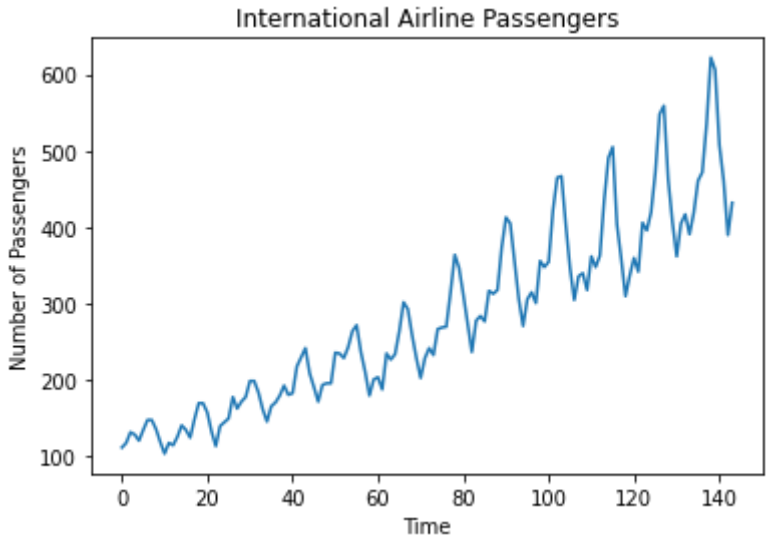
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


dataset = pd.read_csv('./datasets_11956_16450_international-airline-passengers.csv')
dataset = dataset.drop(dataset.index[[144]])
data = dataset.iloc[:, 1:2].values
# print(data)
print(data.shape) # (144, 1)
```

```
(144, 1)
```

```
plt.plot(data)
plt.xlabel("Time")
plt.ylabel("Number of Passengers")
plt.title("International Airline Passengers")
plt.show()
```




```
data.astype("float32")
```

```
array([[112.],
       [118.],
       [132.],
       [129.],
       [121.],
       [135.],
       [148.],
       [148.],
       [136.],
       [119.],
       [104.],
       [118.],
       [115.],
       [126.],
       [141.],
       [135.],
       [125.],
       [149.],
       [170.],
       [170.],
       [158.],
       [133.],
       [114.],
       [140.],
       [145.],
       [150.],
       [178.],
       [163.],
       [172.],
       [178.],
       [199.],
       [199.],
       [184.],
       [162.],
       [146.],
       [166.],
       [171.],
       [180.],
       [193.],
       [181.],
       [183.],
       [218.],
       [230.],
       [242.],
       [209.],
       [191.],
       [172.],
       [194.],
       [196.],
       [196.],
       [236.],
       [235.],
       [229.],
       [243.],
       [264.],
       [272.],
       [237.],
       [211.],
       [180.],
       [201.],
       [204.],
```

```
       [188.],
       [235.],
       [227.],
       [234.],
       [264.],
       [302.],
       [293.],
       [259.],
       [229.],
       [203.],
       [229.],
       [242.],
       [233.],
       [267.],
       [269.],
       [270.],
       [315.],
       [364.],
       [347.],
       [312.],
       [274.],
       [237.],
       [278.],
       [284.],
       [277.],
       [317.],
       [313.],
       [318.],
       [374.],
       [413.],
       [405.],
       [355.],
       [306.],
       [271.],
       [306.],
       [315.],
       [301.],
       [356.],
       [348.],
       [355.],
       [422.],
       [465.],
       [467.],
       [404.],
       [347.],
       [305.],
       [336.],
       [340.],
       [318.],
       [362.],
       [348.],
       [363.],
       [435.],
       [491.],
       [505.],
       [404.],
       [359.],
       [310.],
       [337.],
       [360.],
       [342.],
       [406.],
```

```
       [396.],
       [420.],
       [472.],
       [548.],
       [559.],
       [463.],
       [407.]])
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
data_scaled = scaler.fit_transform(data)
data_scaled
```

```
array([[0.01544402],
       [0.02702703],
       [0.05405405],
       [0.04826255],
       [0.03281853],
       [0.05984556],
       [0.08494208],
       [0.08494208],
       [0.06177606],
       [0.02895753],
       [0.        ],
       [0.02702703],
       [0.02123552],
       [0.04247104],
       [0.07142857],
       [0.05984556],
       [0.04054054],
       [0.08687259],
       [0.12741313],
       [0.12741313],
       [0.1042471 ],
       [0.05598456],
       [0.01930502],
       [0.06949807],
       [0.07915058],
       [0.08880309],
       [0.14285714],
       [0.11389961],
       [0.13127413],
       [0.14285714],
       [0.18339768],
       [0.18339768],
       [0.15444015],
       [0.11196911],
       [0.08108108],
       [0.11969112],
       [0.12934363],
       [0.14671815],
       [0.17181467],
       [0.14864865],
       [0.15250965],
       [0.22007722],
       [0.24324324],
       [0.26640927],
       [0.2027027 ],
       [0.16795367],
       [0.13127413],
       [0.17374517],
       [0.17760618],
       [0.17760618],
       [0.25482625],
       [0.25289575],
       [0.24131274],
       [0.26833977],
       [0.30888031],
       [0.32432432],
       [0.25675676],
       [0.20656371],
       [0.14671815],
       [0.18725869],
       [0.19305019],
```

```
       [0.16216216],
       [0.25289575],
       [0.23745174],
       [0.25096525],
       [0.30888031],
       [0.38223938],
       [0.36486486],
       [0.2992278 ],
       [0.24131274],
       [0.19111969],
       [0.24131274],
       [0.26640927],
       [0.24903475],
       [0.31467181],
       [0.31853282],
       [0.32046332],
       [0.40733591],
       [0.5019305 ],
       [0.46911197],
       [0.4015444 ],
       [0.32818533],
       [0.25675676],
       [0.33590734],
       [0.34749035],
       [0.33397683],
       [0.41119691],
       [0.4034749 ],
       [0.41312741],
       [0.52123552],
       [0.5965251 ],
       [0.58108108],
       [0.48455598],
       [0.38996139],
       [0.32239382],
       [0.38996139],
       [0.40733591],
       [0.38030888],
       [0.48648649],
```

```
time_steps=10
train_data_size = int(len(data_scaled)*0.70)
# train_data_size = 100
test_data_size = len(data_scaled) - train_data_size
print("Train data size is {}".format(train_data_size))
print("Test data size is {}".format(test_data_size))
```

```
Train data size is 100
Test data size is 44
```

```
       [0.4980695 ],
```

```
train = data_scaled[0:train_data_size,:]
test = data_scaled[train_data_size:len(data_scaled),:]
print("Train data size is {}".format(len(train)))
print("Test data size is {}".format(len(test)))
```

```
Train data size is 100
Test data size is 44
```

```
       [0.49420849],
```

```
x_train = []
y_train = []
```

```
y_train = []

# for i in range(time_steps, len(train)):
#   a = train[i- time_steps: i, 0]
#   x_train.append(a)
#   y_train.append(train[i, 0])
for i in range(len(train)-time_steps-1):
    a = train[i:(i+time_steps),0]
    x_train.append(a)
    y_train.append(train[i + time_steps,0])
trainX = np.array(x_train)
trainY = np.array(y_train)
trainX.shape
```

```
(89, 10)
```

```
           [0.96911197],
```

```
x_test = []
y_test = []

# for i in range(len(train)-time_steps-1, len(train)):
#     a = train[i: i+time_steps, 0]
#     x_test.append(a)
#     y_test.append(train[i + time_steps, 0])

'''
# temp = len(train) - 1 - time_steps
# print(test.shape)
# print(train.shape)
# test = np.concatenate((train[temp:, 0], test))

# for i in range(len(train)-time_steps-1):
#     a = train[i-time_steps: i, 0]
#     x_test.append(a)
#     y_test.append(test[i + time_steps,0])
# temp = len(x_train) - 1 - time_steps
# test = np.concatenate(x_train[temp:, 0], test)
# for i in range(time_steps, len(test)):
#     a = test[i-time_steps: i, 0]
#     x_test.append(a)
#     y_test.append(test[i, 0])

'''

for i in range(len(test)-time_steps-1):
    a = test[i:(i+time_steps),0]
    x_test.append(a)
    y_test.append(test[i + time_steps,0])

testX = np.array(x_test)
# testX = np.reshape(testX, (len(testX), 1))
testY = np.array(y_test)
testX.shape
```

```
(33, 10)
```

```
trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1],1))
testX = np.reshape(testX, (testX.shape[0],testX.shape[1],1))
# Print and check shapes
print("Shape of trainX is {}".format(trainX.shape))
print("Shape of testX is {}".format(testX.shape))
# print(testX)
```

```
Shape of trainX is (89, 10, 1)
Shape of testX is (33, 10, 1)
```

```
from keras.layers import Dense, SimpleRNN, Dropout
from keras.metrics import mean_squared_error
from keras.models import Sequential


model = Sequential()
# Add the first layer and Dropout regularization
model.add(SimpleRNN(units=100,activation='tanh',return_sequences=True,
                    input_shape=(trainX.shape[1],1)))
model.add(Dropout(0.20))
# Second layer and Dropout regularization
model.add(SimpleRNN(units = 100, activation='tanh',return_sequences=True))
model.add(Dropout(0.20))
# Third layer and Dropout regularization
model.add(SimpleRNN(units = 70, activation='tanh', return_sequences= True))
model.add(Dropout(0.20))
# Fourth layer and Dropout regularization
model.add(SimpleRNN(units = 50))
model.add(Dropout(0.20))
# Add final or output layer
model.add(Dense(units=1))

# Compile our RNN model
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
# Fitting the RNN to the training set
model.fit(trainX, trainY, epochs = 200, batch_size=32)
```

```
Epoch 1/200
3/3 [==============================] - 0s 14ms/step - loss: 0.6214
Epoch 2/200
3/3 [==============================] - 0s 9ms/step - loss: 0.4257
Epoch 3/200
3/3 [==============================] - 0s 9ms/step - loss: 0.2849
Epoch 4/200
3/3 [==============================] - 0s 8ms/step - loss: 0.2331
Epoch 5/200
3/3 [==============================] - 0s 8ms/step - loss: 0.3060
Epoch 6/200
3/3 [==============================] - 0s 8ms/step - loss: 0.2179
Epoch 7/200
3/3 [==============================] - 0s 8ms/step - loss: 0.1847
Epoch 8/200
3/3 [==============================] - 0s 8ms/step - loss: 0.2207
Epoch 9/200
3/3 [==============================] - 0s 8ms/step - loss: 0.1928
Epoch 10/200
3/3 [==============================] - 0s 9ms/step - loss: 0.1830
Epoch 11/200
3/3 [==============================] - 0s 9ms/step - loss: 0.1391
Epoch 12/200
3/3 [==============================] - 0s 8ms/step - loss: 0.1567
Epoch 13/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0943
Epoch 14/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0922
Epoch 15/200
3/3 [==============================] - 0s 9ms/step - loss: 0.1097
Epoch 16/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0865
Epoch 17/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0670
Epoch 18/200
3/3 [==============================] - 0s 8ms/step - loss: 0.1005
Epoch 19/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0746
Epoch 20/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0674
Epoch 21/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0958
Epoch 22/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0819
Epoch 23/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0810
Epoch 24/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0545
Epoch 25/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0565
Epoch 26/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0597
Epoch 27/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0534
Epoch 28/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0634
Epoch 29/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0488
Epoch 30/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0665
Epoch 31/200
```

```
3/3 [==============================] - 0s 11ms/step - loss: 0.0673
Epoch 32/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0575
Epoch 33/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0628
Epoch 34/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0544
Epoch 35/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0489
Epoch 36/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0469
Epoch 37/200
3/3 [==============================] - 0s 11ms/step - loss: 0.0480
Epoch 38/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0449
Epoch 39/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0391
Epoch 40/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0464
Epoch 41/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0452
Epoch 42/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0355
Epoch 43/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0350
Epoch 44/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0390
Epoch 45/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0362
Epoch 46/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0329
Epoch 47/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0275
Epoch 48/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0455
Epoch 49/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0327
Epoch 50/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0362
Epoch 51/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0282
Epoch 52/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0319
Epoch 53/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0379
Epoch 54/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0399
Epoch 55/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0334
Epoch 56/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0326
Epoch 57/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0234
Epoch 58/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0243
Epoch 59/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0370
Epoch 60/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0301
Epoch 61/200
3/3 [==============================] - 0s 11ms/step - loss: 0.0196
Epoch 62/200
```

```
Epoch 62/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0191
Epoch 63/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0235
Epoch 64/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0304
Epoch 65/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0188
Epoch 66/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0324
Epoch 67/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0185
Epoch 68/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0205
Epoch 69/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0207
Epoch 70/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0231
Epoch 71/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0251
Epoch 72/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0158
Epoch 73/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0206
Epoch 74/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0156
Epoch 75/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0193
Epoch 76/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0155
Epoch 77/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0205
Epoch 78/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0195
Epoch 79/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0210
Epoch 80/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0245
Epoch 81/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0177
Epoch 82/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0164
Epoch 83/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0159
Epoch 84/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0100
Epoch 85/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0159
Epoch 86/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0107
Epoch 87/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0133
Epoch 88/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0118
Epoch 89/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0163
Epoch 90/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0135
Epoch 91/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0130
Epoch 92/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0125
```

```
Epoch 93/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0149
Epoch 94/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0127
Epoch 95/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0103
Epoch 96/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0104
Epoch 97/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0131
Epoch 98/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0093
Epoch 99/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0159
Epoch 100/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0093
Epoch 101/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0107
Epoch 102/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0131
Epoch 103/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0091
Epoch 104/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0117
Epoch 105/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0098
Epoch 106/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0096
Epoch 107/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0128
Epoch 108/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0110
Epoch 109/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0126
Epoch 110/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0139
Epoch 111/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0131
Epoch 112/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0112
Epoch 113/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0116
Epoch 114/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0087
Epoch 115/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0103
Epoch 116/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0101
Epoch 117/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0071
Epoch 118/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0089
Epoch 119/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0093
Epoch 120/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0078
Epoch 121/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0126
Epoch 122/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0096
Epoch 123/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0120
```

```
Epoch 124/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0103
Epoch 125/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0109
Epoch 126/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0084
Epoch 127/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0091
Epoch 128/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0120
Epoch 129/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0114
Epoch 130/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0120
Epoch 131/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0106
Epoch 132/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0098
Epoch 133/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0118
Epoch 134/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0114
Epoch 135/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0091
Epoch 136/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0077
Epoch 137/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0098
Epoch 138/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0095
Epoch 139/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0075
Epoch 140/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0091
Epoch 141/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0115
Epoch 142/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0107
Epoch 143/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0071
Epoch 144/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0085
Epoch 145/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0089
Epoch 146/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0082
Epoch 147/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0076
Epoch 148/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0089
Epoch 149/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0101
Epoch 150/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0071
Epoch 151/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0069
Epoch 152/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0076
Epoch 153/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0088
Epoch 154/200
```

```
3/3 [==============================] - 0s 9ms/step - loss: 0.0070
Epoch 155/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0063
Epoch 156/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0057
Epoch 157/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0081
Epoch 158/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0065
Epoch 159/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0081
Epoch 160/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0099
Epoch 161/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0059
Epoch 162/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0073
Epoch 163/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0078
Epoch 164/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0100
Epoch 165/200
3/3 [==============================] - 0s 11ms/step - loss: 0.0067
Epoch 166/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0070
Epoch 167/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0072
Epoch 168/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0068
Epoch 169/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0069
Epoch 170/200
3/3 [==============================] - 0s 12ms/step - loss: 0.0081
Epoch 171/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0083
Epoch 172/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0066
Epoch 173/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0065
Epoch 174/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0080
Epoch 175/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0085
Epoch 176/200
3/3 [==============================] - 0s 10ms/step - loss: 0.0075
Epoch 177/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0081
Epoch 178/200
3/3 [==============================] - 0s 11ms/step - loss: 0.0087
Epoch 179/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0078
Epoch 180/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0084
Epoch 181/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0070
Epoch 182/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0075
Epoch 183/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0058
Epoch 184/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0066
Epoch 185/200
```

```
3/3 [==============================] - 0s 8ms/step - loss: 0.0062
Epoch 186/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0060
Epoch 187/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0091
Epoch 188/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0065
Epoch 189/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0088
Epoch 190/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0063
Epoch 191/200
3/3 [==============================] - 0s 8ms/step - loss: 0.0073
Epoch 192/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0066
Epoch 193/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0058
Epoch 194/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0069
Epoch 195/200
3/3 [==============================] - 0s 9ms/step - loss: 0.0036
```

```
trainPrediction = model.predict(trainX)
# testX = testX.astype(np.int)
testPrediction = model.predict(testX)

# we scaled datas between 0 and 1 but now we're at the end of the project.
# So we should inverse transform datas.

trainPrediction = scaler.inverse_transform(trainPrediction)
trainY = scaler.inverse_transform([trainY])
testPrediction = scaler.inverse_transform(testPrediction)
testY = scaler.inverse_transform([testY])


# import tensorflow as tf
# import math
# sess = tf.compat.v1.Session()
# with sess.as_default():
#     trainScore = math.sqrt(mean_squared_error(np.array(trainY[0]), np.array(trainPredict
#     testScore = math.sqrt(mean_squared_error(np.array(testY[0]), np.array(testPredictior
# print("Train Score is %.2lf RMSE"%(trainScore))
# print("Test Score is %.2lf RMSE"%(testScore))


trainPredictPlot = np.empty_like(dataset)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[time_steps:len(trainPrediction)+time_steps, :] = trainPrediction

testPredictPlot = np.empty_like(dataset)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(trainPrediction)+(time_steps*2)+1:len(dataset)-1, :] = testPrediction

plt.plot(scaler.inverse_transform(data_scaled),label = 'True Values', color='blue')
plt.plot(trainPredictPlot,label='Train Prediction', color='green')
plt.plot(testPredictPlot,label = 'Test Prediction', color='red')
plt.xlabel("Time")
plt.ylabel("Number of Passengers")
```

```
plt.title("International Airline Passengers")
plt.legend()
plt.show()
```