stock-prediction-model-1

April 3, 2024

1 Day 22 - Stock Price prediction using GRU

1.0.1 In this notebook we are going to learn , How can we attempt to predict future stock behavior? (Predicting the closing price stock price of GOOGLE inc using GRU)

[ ]:

```

```

Importing the dependencies

[ ]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, GRU, Dropout
```

Loading the Data

[ ]:

```
data = pd.read_csv('/content/drive/MyDrive/stock_price/
 ↪GOOGL_2006-01-01_to_2018-01-01.csv')
```

[ ]:

```
data.head()
```

[ ]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	211.47	218.05	209.32	217.83	13137450	GOOGL
1	2006-01-04	222.17	224.70	220.09	222.84	15292353	GOOGL
2	2006-01-05	223.22	226.00	220.97	225.85	10815661	GOOGL
3	2006-01-06	228.66	235.49	226.85	233.06	17759521	GOOGL
4	2006-01-09	233.44	236.94	230.70	233.68	12795837	GOOGL

Normalising the Closing price of the stocks on the time periods,We Just going the predict the Closing price of the stocks , so lets clean the data as well.

[ ]:

```
data = data.sort_values('Date')
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1, 1))
```

[ ]:

```
scaled_data[:10]
```

[ ]:
```
array([[0.09305195],
       [0.09829122],
       [0.10143897],
       [0.10897892],
       [0.10962729],
       [0.11112273],
       [0.11210575],
       [0.1079227 ],
       [0.10929265],
       [0.10974232]])
```

[ ]:
```
training_data_len = int(len(scaled_data) * 0.8)
train_data = scaled_data[0:training_data_len, :]
```

[ ]:
```
train_data[:10]
```

[ ]:
```
array([[0.09305195],
       [0.09829122],
       [0.10143897],
       [0.10897892],
       [0.10962729],
       [0.11112273],
       [0.11210575],
       [0.1079227 ],
       [0.10929265],
       [0.10974232]])
```

The following loop iterates over the train_data array:

a. **for i in range(60, len(train_data))** This loop starts from index 60 because it seems to be creating sequences of 60 data points (which is a common approach in sequence modeling tasks).

b. **X_train.append(train_data[i-60:i, 0])** For each iteration, it appends a sequence of 60 data points to X_train. The sequence starts from index i-60 and ends at index i-1. The 0 index selects the first (and only) feature in the sequence.

c. **y_train.append(train_data[i, 0])** It appends the target value (which is the next data point after the 60-point sequence) to y_train.

[ ]:
```
X_train = []
y_train = []
for i in range(60, len(train_data)):
    X_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

```
[ ]: test_data = scaled_data[training_data_len-60:, :]
     X_test = []
     y_test = data['Close'][training_data_len:].values
     for i in range(60, len(test_data)):
         X_test.append(test_data[i-60:i, 0])
     X_test = np.array(X_test)
     X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

1.1 Defining the model

```
[ ]: X_train.shape
```

```
[ ]: (2355, 60, 1)
```

```
[ ]: model = Sequential()
     model.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1],␣
      ↪1)))
     model.add(Dropout(0.2))
     model.add(GRU(units=50, return_sequences=True))
     model.add(Dropout(0.2))
     model.add(GRU(units=50))
     model.add(Dropout(0.2))
     model.add(Dense(units=1))
     model.summary()
```

```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 60, 50)            7950

 dropout (Dropout)           (None, 60, 50)            0

 gru_1 (GRU)                 (None, 60, 50)            15300

 dropout_1 (Dropout)         (None, 60, 50)            0

 gru_2 (GRU)                 (None, 50)                15300

 dropout_2 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 38601 (150.79 KB)
Trainable params: 38601 (150.79 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
[ ]: model.compile(optimizer='adam', loss='mean_squared_error')

     model.fit(X_train, y_train, epochs=50, batch_size=32)

     scores = model.evaluate(X_test, y_test)
     print(f'Test loss: {scores}')
```

```
Epoch 1/50
74/74 [==============================] - 14s 94ms/step - loss: 0.0048
Epoch 2/50
74/74 [==============================] - 7s 93ms/step - loss: 9.1946e-04
Epoch 3/50
74/74 [==============================] - 7s 101ms/step - loss: 9.2688e-04
Epoch 4/50
74/74 [==============================] - 7s 92ms/step - loss: 8.1369e-04
Epoch 5/50
74/74 [==============================] - 8s 106ms/step - loss: 7.3345e-04
Epoch 6/50
74/74 [==============================] - 7s 89ms/step - loss: 6.6950e-04
Epoch 7/50
74/74 [==============================] - 8s 106ms/step - loss: 6.9005e-04
Epoch 8/50
74/74 [==============================] - 6s 86ms/step - loss: 6.4221e-04
Epoch 9/50
74/74 [==============================] - 8s 105ms/step - loss: 5.9840e-04
Epoch 10/50
74/74 [==============================] - 6s 86ms/step - loss: 5.9372e-04
Epoch 11/50
74/74 [==============================] - 8s 107ms/step - loss: 5.9476e-04
Epoch 12/50
74/74 [==============================] - 7s 101ms/step - loss: 5.2801e-04
Epoch 13/50
74/74 [==============================] - 8s 107ms/step - loss: 5.6004e-04
Epoch 14/50
74/74 [==============================] - 6s 86ms/step - loss: 5.4078e-04
Epoch 15/50
74/74 [==============================] - 8s 108ms/step - loss: 5.4221e-04
Epoch 16/50
74/74 [==============================] - 6s 87ms/step - loss: 4.9416e-04
Epoch 17/50
74/74 [==============================] - 8s 107ms/step - loss: 4.2772e-04
Epoch 18/50
74/74 [==============================] - 6s 87ms/step - loss: 4.3772e-04
Epoch 19/50
74/74 [==============================] - 8s 108ms/step - loss: 4.0347e-04
Epoch 20/50
```

```
74/74 [==============================] - 6s 86ms/step - loss: 5.0774e-04
Epoch 21/50
74/74 [==============================] - 8s 107ms/step - loss: 3.4780e-04
Epoch 22/50
74/74 [==============================] - 6s 86ms/step - loss: 3.6389e-04
Epoch 23/50
74/74 [==============================] - 8s 109ms/step - loss: 3.6551e-04
Epoch 24/50
74/74 [==============================] - 6s 86ms/step - loss: 3.8745e-04
Epoch 25/50
74/74 [==============================] - 8s 110ms/step - loss: 3.9199e-04
Epoch 26/50
74/74 [==============================] - 7s 88ms/step - loss: 3.3643e-04
Epoch 27/50
74/74 [==============================] - 8s 106ms/step - loss: 3.5228e-04
Epoch 28/50
74/74 [==============================] - 6s 86ms/step - loss: 3.3757e-04
Epoch 29/50
74/74 [==============================] - 8s 106ms/step - loss: 3.3859e-04
Epoch 30/50
74/74 [==============================] - 7s 88ms/step - loss: 3.3068e-04
Epoch 31/50
74/74 [==============================] - 8s 110ms/step - loss: 3.3078e-04
Epoch 32/50
74/74 [==============================] - 7s 94ms/step - loss: 3.1685e-04
Epoch 33/50
74/74 [==============================] - 7s 100ms/step - loss: 2.8476e-04
Epoch 34/50
74/74 [==============================] - 7s 95ms/step - loss: 2.8798e-04
Epoch 35/50
74/74 [==============================] - 7s 101ms/step - loss: 2.7241e-04
Epoch 36/50
74/74 [==============================] - 7s 100ms/step - loss: 3.0730e-04
Epoch 37/50
74/74 [==============================] - 7s 95ms/step - loss: 2.6979e-04
Epoch 38/50
74/74 [==============================] - 7s 96ms/step - loss: 2.8512e-04
Epoch 39/50
74/74 [==============================] - 7s 95ms/step - loss: 2.9476e-04
Epoch 40/50
74/74 [==============================] - 8s 102ms/step - loss: 2.8685e-04
Epoch 41/50
74/74 [==============================] - 7s 90ms/step - loss: 2.4154e-04
Epoch 42/50
74/74 [==============================] - 8s 104ms/step - loss: 2.5447e-04
Epoch 43/50
74/74 [==============================] - 7s 89ms/step - loss: 2.9979e-04
Epoch 44/50
```

```
74/74 [==============================] - 8s 103ms/step - loss: 2.6718e-04
Epoch 45/50
74/74 [==============================] - 7s 90ms/step - loss: 3.0039e-04
Epoch 46/50
74/74 [==============================] - 7s 100ms/step - loss: 2.5940e-04
Epoch 47/50
74/74 [==============================] - 6s 87ms/step - loss: 2.6613e-04
Epoch 48/50
74/74 [==============================] - 8s 108ms/step - loss: 3.1226e-04
Epoch 49/50
74/74 [==============================] - 6s 86ms/step - loss: 2.3899e-04
Epoch 50/50
74/74 [==============================] - 8s 109ms/step - loss: 2.3882e-04
19/19 [==============================] - 2s 29ms/step - loss: 696394.6250
Test loss: 696394.625
```

[ ]:
```python
# Make predictions
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
```

```
19/19 [==============================] - 1s 67ms/step
```

[ ]:
```python
```