```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras import layers
from keras.models import Sequential
```

```
dtrain=pd.read_csv('trainset.csv')
dtrain.columns
dtrain.head()
dtrainset=dtrain.iloc[:,1:2].values
```

```
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(dtrainset)
training_set_scaled.shape
```

```
(1259, 1)
```

```
X_train_array = []
y_train_array = []
for i in range(60, 1259):
    X_train_array.append(training_set_scaled[i-60:i,0])
    y_train_array.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train_array), np.array(y_train_array)
X_train1 = X_train.reshape((X_train.shape[0], X_train.shape[1],1))
X_train.shape
```

```
(1199, 60)
```

```
X_train_array = []
y_train_array = []
for i in range(60, 1259):
    X_train_array.append(training_set_scaled[i-60:i,0])
    y_train_array.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train_array), np.array(y_train_array)
X_train1 = X_train.reshape((X_train.shape[0], X_train.shape[1],1))
X_train.shape
```

```
(1199, 60)
```

```
model = Sequential([layers.SimpleRNN(42,input_shape=(60,1)),layers.Dense(1)])
model.compile(optimizer='adam',loss='mse')
model.summary()
model.fit(X_train1,y_train,epochs=20, batch_size=32)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argum
  super().__init__(**kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ simple_rnn (SimpleRNN)               │ (None, 42)                  │           1,848 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 1)                   │              43 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 1,891 (7.39 KB)
 Trainable params: 1,891 (7.39 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0725
Epoch 2/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0011
Epoch 3/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 6.8447e-04
Epoch 4/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 6.1171e-04
Epoch 5/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 5.4241e-04
Epoch 6/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 4.2521e-04
Epoch 7/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 4.6907e-04
Epoch 8/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 4.8632e-04
Epoch 9/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 3.7656e-04
Epoch 10/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 3.8287e-04
Epoch 11/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 3.8290e-04
Epoch 12/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 4.2719e-04
Epoch 13/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.3009e-04
Epoch 14/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.6482e-04
Epoch 15/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.5131e-04
Epoch 16/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.3443e-04
Epoch 17/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.7001e-04
Epoch 18/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 3.4820e-04
Epoch 19/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 3.3990e-04
Epoch 20/20
38/38 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 3.1563e-04
<keras.src.callbacks.history.History at 0x7b18c1a8a980>
```

```
dataset_test = pd.read_csv('testset.csv')
test_set = dataset_test.iloc[:,1:2].values
test_set.shape
```

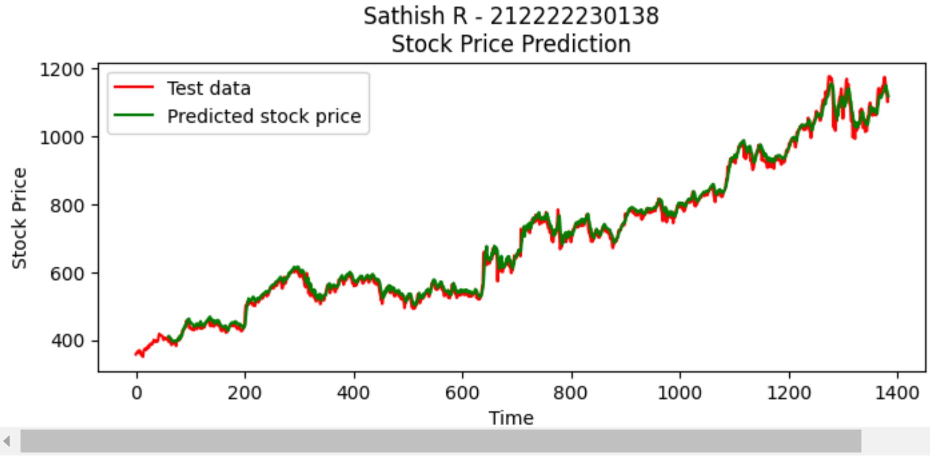
```
(125, 1)
```

```
dataset_total = pd.concat((dtrain['Open'],dataset_test['Open']),axis=0)
inputs = dataset_total.values
inputs = inputs.reshape(-1,1)
inputs_scaled=sc.transform(inputs)
X_test = []
y_test = []
for i in range(60,1384):
    X_test.append(inputs_scaled[i-60:i,0])
    y_test.append(inputs_scaled[i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0], X_test.shape[1],1))
X_test.shape
```

```
(1324, 60, 1)
```

```
predicted_stock_price_scaled = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price_scaled)
plt.figure(figsize=(8,3))
plt.plot(np.arange(0,1384),inputs, color='red', label = 'Test data')
plt.plot(np.arange(60,1384),predicted_stock_price, color='green',label = 'Predicted stock price')
plt.title('Sathish R - 212222230138\nStock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

```
42/42 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step
```


Sathish R - 212222230138
Stock Price Prediction
Test data
Predicted stock price
Stock Price
Time