```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import kagglehub


# Download dataset using kagglehub
path = kagglehub.dataset_download("akram24/google-stock-price-test")
dataset_path = f"{path}/Google_Stock_Price_Test.csv"


# Load dataset
data = pd.read_csv(dataset_path, usecols=[1]).dropna().values.astype(float)


# Normalize data
scaler = MinMaxScaler()
data = scaler.fit_transform(data)


# Prepare time series data
def create_dataset(dataset, time_step=10):
    X, y = [], []
    for i in range(len(dataset) - time_step):
        X.append(dataset[i:i + time_step, 0])
        y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(y)


# Ensure dataset has enough samples
time_step = min(10, len(data) - 1)
if len(data) > time step:
```

```
    X, y = create_dataset(data, time_step)
    X = X.reshape(X.shape[0], X.shape[1], 1)

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Define RNN model
    model = Sequential([
        SimpleRNN(50, activation='relu', return_sequences=True, input_shape=(time_step, 1)),
        SimpleRNN(50, activation='relu'),
        Dense(1)
    ])

    # Compile and train model
    model.compile(optimizer='adam', loss='mse')
    model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

    # Evaluate model
    print(f"Test Loss: {model.evaluate(X_test, y_test)}")
else:
    print("Error: Not enough data points to create sequences. Consider using a smaller time step.")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a la
  super().__init__(**kwargs)
Epoch 1/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - loss: 0.4295 - val_loss: 0.1907
Epoch 2/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step - loss: 0.3778 - val_loss: 0.1483
Epoch 3/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - loss: 0.3255 - val_loss: 0.1099
Epoch 4/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - loss: 0.2773 - val_loss: 0.0755
Epoch 5/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - loss: 0.2342 - val_loss: 0.0470
Epoch 6/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 106ms/step - loss: 0.1952 - val_loss: 0.0247
Epoch 7/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - loss: 0.1594 - val_loss: 0.0099
Epoch 8/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - loss: 0.1278 - val_loss: 0.0021
Epoch 9/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 161ms/step - loss: 0.1021 - val_loss: 0.0021
Epoch 10/20
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - loss: 0.0815 - val_loss: 0.0118
Epoch 11/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - loss: 0.0668 - val_loss: 0.0330
Epoch 12/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - loss: 0.0614 - val_loss: 0.0642
Epoch 13/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - loss: 0.0659 - val_loss: 0.0972
Epoch 14/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - loss: 0.0765 - val_loss: 0.1195
Epoch 15/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 0.0854 - val_loss: 0.1254
Epoch 16/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - loss: 0.0872 - val_loss: 0.1182
Epoch 17/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - loss: 0.0827 - val_loss: 0.1013
Epoch 18/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step - loss: 0.0739 - val_loss: 0.0822
Epoch 19/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step - loss: 0.0652 - val_loss: 0.0634
Epoch 20/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - loss: 0.0587 - val_loss: 0.0478
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 0.0478
Test Loss: 0.047767698764801025
```

```
Start coding or ge_nerate with AI.
```