**Task:**

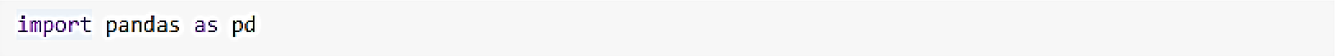
Task: Time Series Forecasting with RNN

**Objective:**

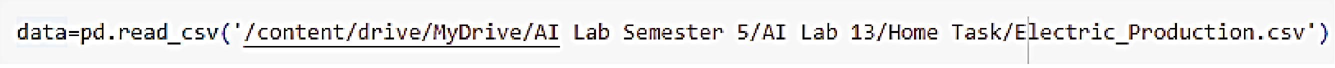
Build an RNN model for time series forecasting using a dataset (https://www.kaggle.com/datasets/shenba/time-series-datasets) with sequential data. The goal is to predict future values in the time series based on historical patterns.

**Include these Basic Steps:**

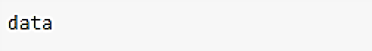
1. Dataset Selection
2. Data Preprocessing
3. Train-Test Split
4. RNN Model Architecture
5. Model Training
6. Validation
7. Hyperparameter Tuning
8. Performance Evaluation
9. Visualization
10. Future Predictions



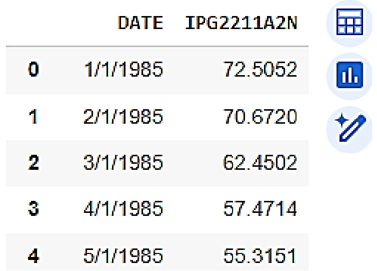
```
import pandas as pd
```



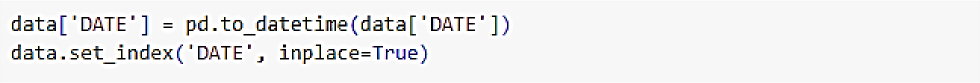
```
data=pd.read_csv('/content/drive/MyDrive/AI Lab Semester 5/AI Lab 13/Home Task/Electric_Production.csv')
```



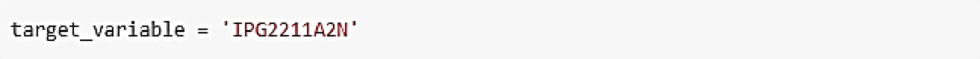
```
data
```



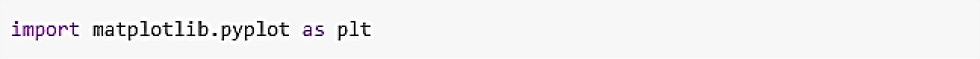
|   | DATE | IPG2211A2N |
|---|---|---|
| **0** | 1/1/1985 | 72.5052 |
| **1** | 2/1/1985 | 70.6720 |
| **2** | 3/1/1985 | 62.4502 |
| **3** | 4/1/1985 | 57.4714 |
| **4** | 5/1/1985 | 55.3151 |



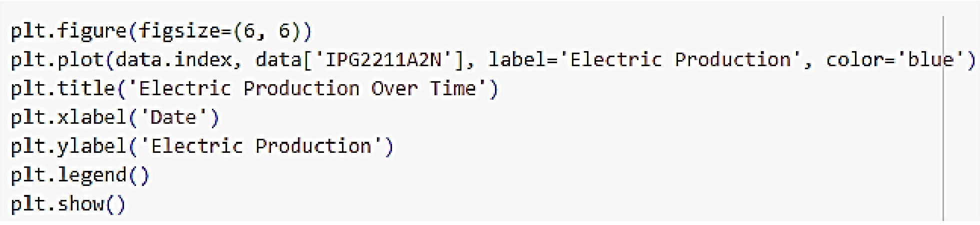
```
data['DATE'] = pd.to_datetime(data['DATE'])
data.set_index('DATE', inplace=True)
```



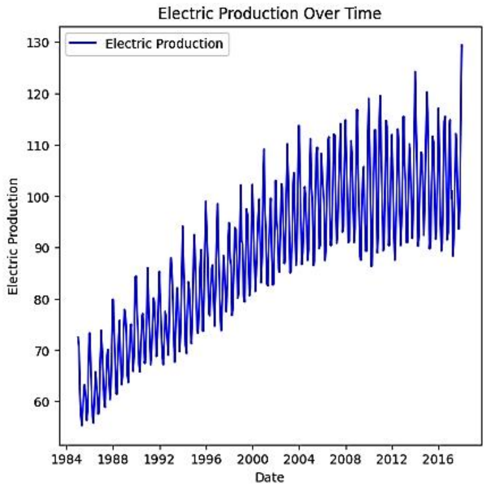
```
target_variable = 'IPG2211A2N'
```

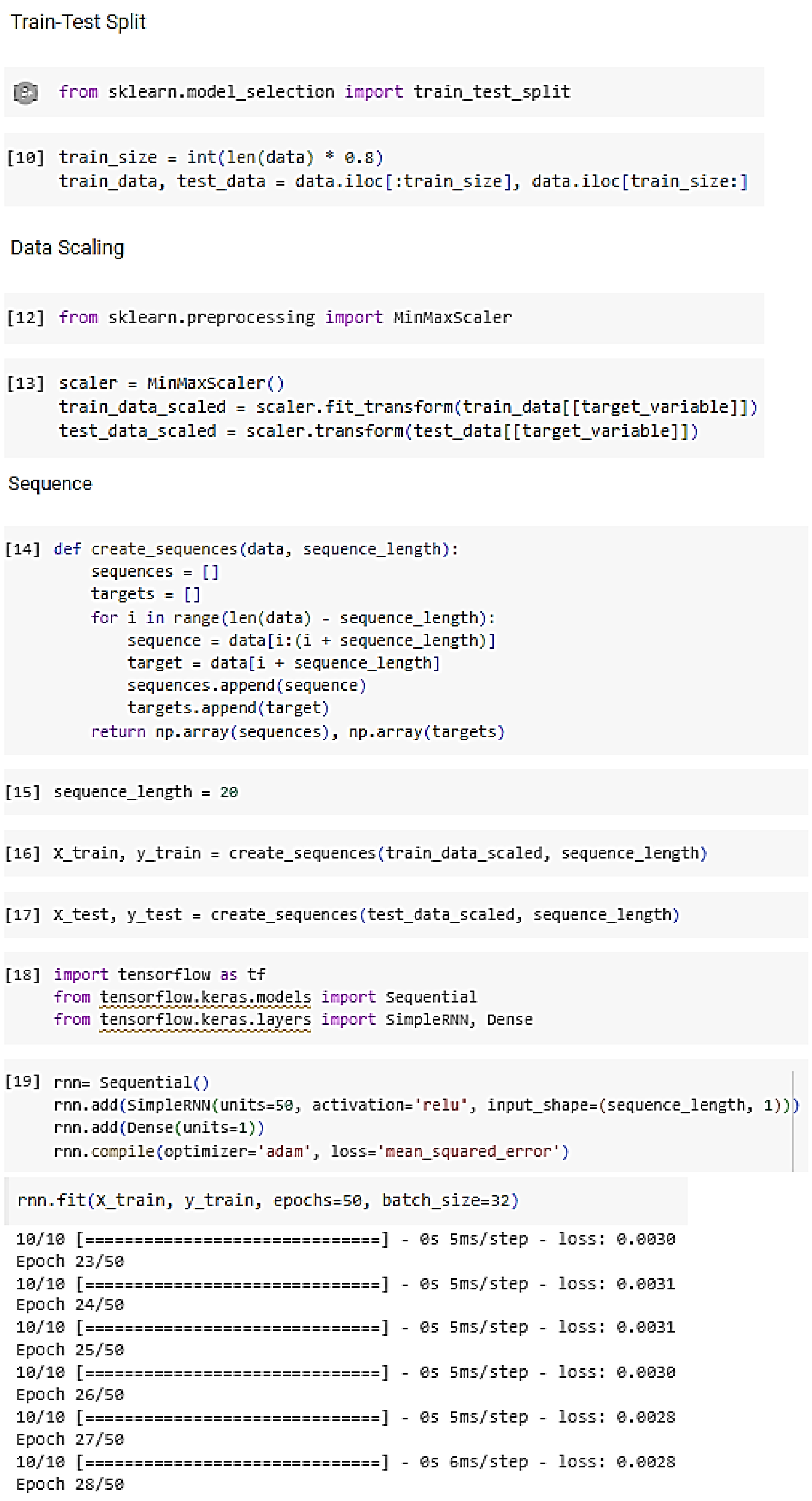


```
import matplotlib.pyplot as plt
```



```
plt.figure(figsize=(6, 6))
plt.plot(data.index, data['IPG2211A2N'], label='Electric Production', color='blue')
plt.title('Electric Production Over Time')
plt.xlabel('Date')
plt.ylabel('Electric Production')
plt.legend()
plt.show()
```





### Train-Test Split

```
     from sklearn.model_selection import train_test_split
```

```
[10] train_size = int(len(data) * 0.8)
     train_data, test_data = data.iloc[:train_size], data.iloc[train_size:]
```

### Data Scaling

```
[12] from sklearn.preprocessing import MinMaxScaler
```

```
[13] scaler = MinMaxScaler()
     train_data_scaled = scaler.fit_transform(train_data[[target_variable]])
     test_data_scaled = scaler.transform(test_data[[target_variable]])
```

### Sequence

```
[14] def create_sequences(data, sequence_length):
         sequences = []
         targets = []
         for i in range(len(data) - sequence_length):
             sequence = data[i:(i + sequence_length)]
             target = data[i + sequence_length]
             sequences.append(sequence)
             targets.append(target)
         return np.array(sequences), np.array(targets)
```

```
[15] sequence_length = 20
```

```
[16] X_train, y_train = create_sequences(train_data_scaled, sequence_length)
```

```
[17] X_test, y_test = create_sequences(test_data_scaled, sequence_length)
```

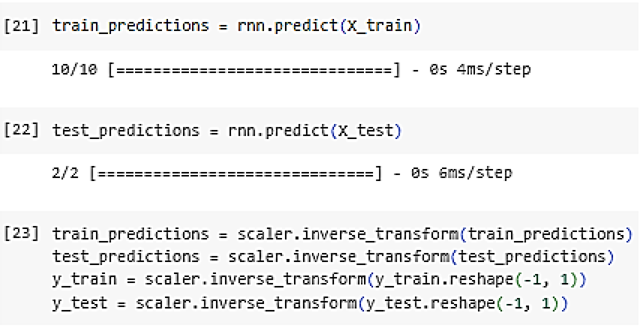
```
[18] import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import SimpleRNN, Dense
```

```
[19] rnn= Sequential()
     rnn.add(SimpleRNN(units=50, activation='relu', input_shape=(sequence_length, 1)))
     rnn.add(Dense(units=1))
     rnn.compile(optimizer='adam', loss='mean_squared_error')
```

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

```
10/10 [==============================] - 0s 5ms/step - loss: 0.0030
Epoch 23/50
10/10 [==============================] - 0s 5ms/step - loss: 0.0031
Epoch 24/50
10/10 [==============================] - 0s 5ms/step - loss: 0.0031
Epoch 25/50
10/10 [==============================] - 0s 5ms/step - loss: 0.0030
Epoch 26/50
10/10 [==============================] - 0s 5ms/step - loss: 0.0028
Epoch 27/50
10/10 [==============================] - 0s 6ms/step - loss: 0.0028
Epoch 28/50
```

Validation

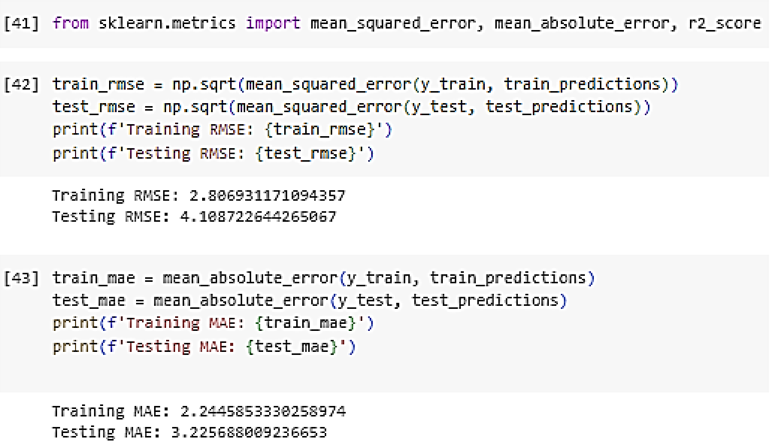


```
[21] train_predictions = rnn.predict(X_train)

     10/10 [==============================] - 0s 4ms/step
```

```
[22] test_predictions = rnn.predict(X_test)

     2/2 [==============================] - 0s 6ms/step
```

```
[23] train_predictions = scaler.inverse_transform(train_predictions)
     test_predictions = scaler.inverse_transform(test_predictions)
     y_train = scaler.inverse_transform(y_train.reshape(-1, 1))
     y_test = scaler.inverse_transform(y_test.reshape(-1, 1))
```

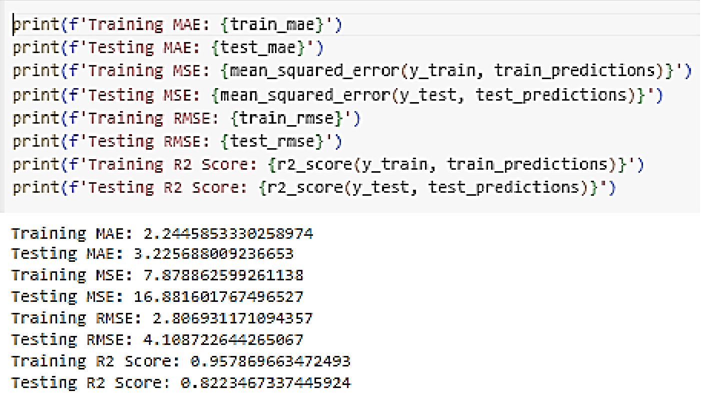
Evaluation



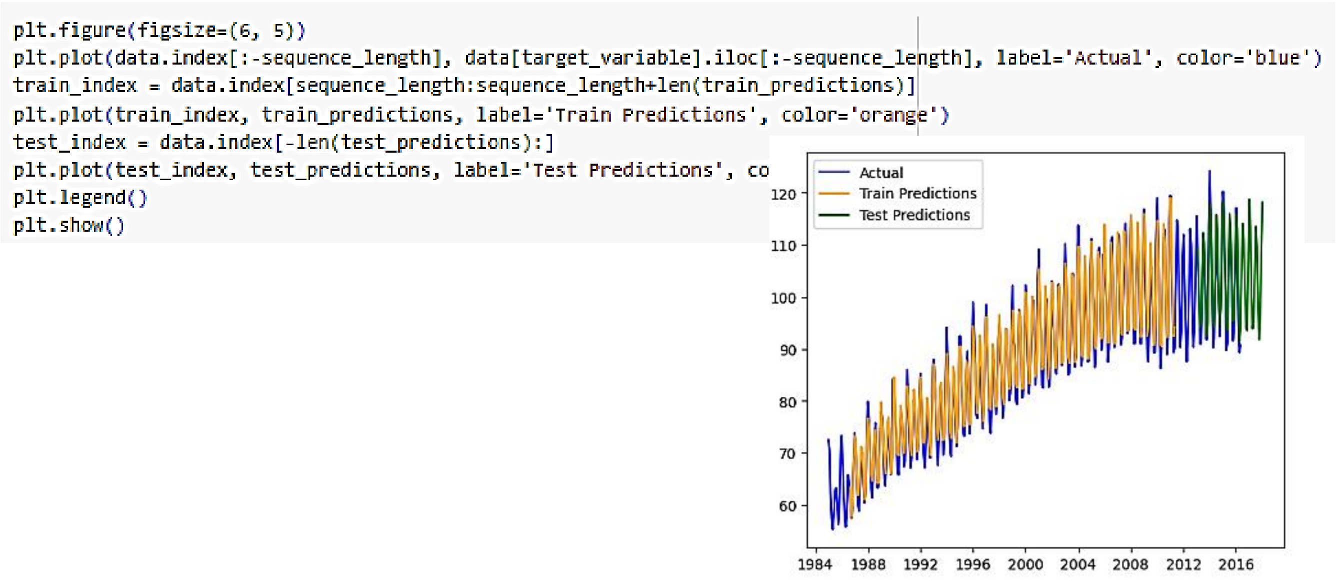
```
[41] from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
[42] train_rmse = np.sqrt(mean_squared_error(y_train, train_predictions))
     test_rmse = np.sqrt(mean_squared_error(y_test, test_predictions))
     print(f'Training RMSE: {train_rmse}')
     print(f'Testing RMSE: {test_rmse}')

     Training RMSE: 2.806931171094357
     Testing RMSE: 4.108722644265067
```

```
[43] train_mae = mean_absolute_error(y_train, train_predictions)
     test_mae = mean_absolute_error(y_test, test_predictions)
     print(f'Training MAE: {train_mae}')
     print(f'Testing MAE: {test_mae}')


     Training MAE: 2.2445853330258974
     Testing MAE: 3.225688009236653
```

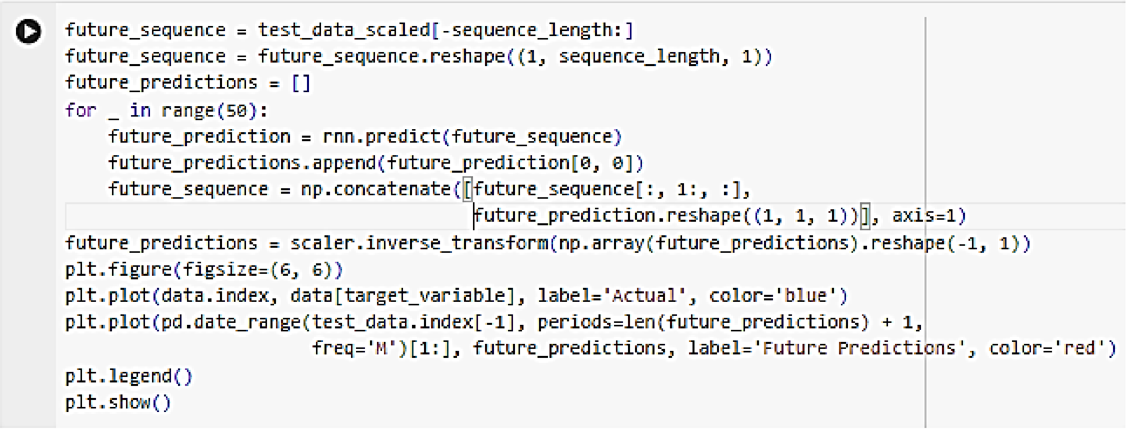


```
print(f'Training MAE: {train_mae}')
print(f'Testing MAE: {test_mae}')
print(f'Training MSE: {mean_squared_error(y_train, train_predictions)}')
print(f'Testing MSE: {mean_squared_error(y_test, test_predictions)}')
print(f'Training RMSE: {train_rmse}')
print(f'Testing RMSE: {test_rmse}')
print(f'Training R2 Score: {r2_score(y_train, train_predictions)}')
print(f'Testing R2 Score: {r2_score(y_test, test_predictions)}')

Training MAE: 2.2445853330258974
Testing MAE: 3.225688009236653
Training MSE: 7.878862599261138
Testing MSE: 16.881601767496527
Training RMSE: 2.806931171094357
Testing RMSE: 4.108722644265067
Training R2 Score: 0.957869663472493
Testing R2 Score: 0.8223467337445924
```

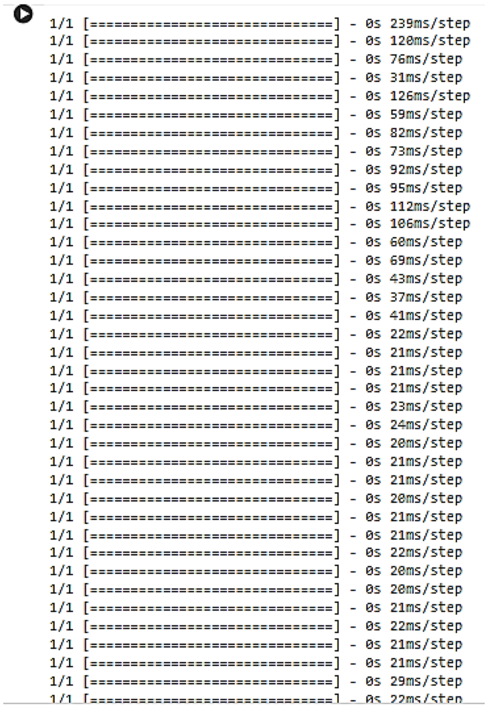


```
plt.figure(figsize=(6, 5))
plt.plot(data.index[:-sequence_length], data[target_variable].iloc[:-sequence_length], label='Actual', color='blue')
train_index = data.index[sequence_length:sequence_length+len(train_predictions)]
plt.plot(train_index, train_predictions, label='Train Predictions', color='orange')
test_index = data.index[-len(test_predictions):]
plt.plot(test_index, test_predictions, label='Test Predictions', co
plt.legend()
plt.show()
```

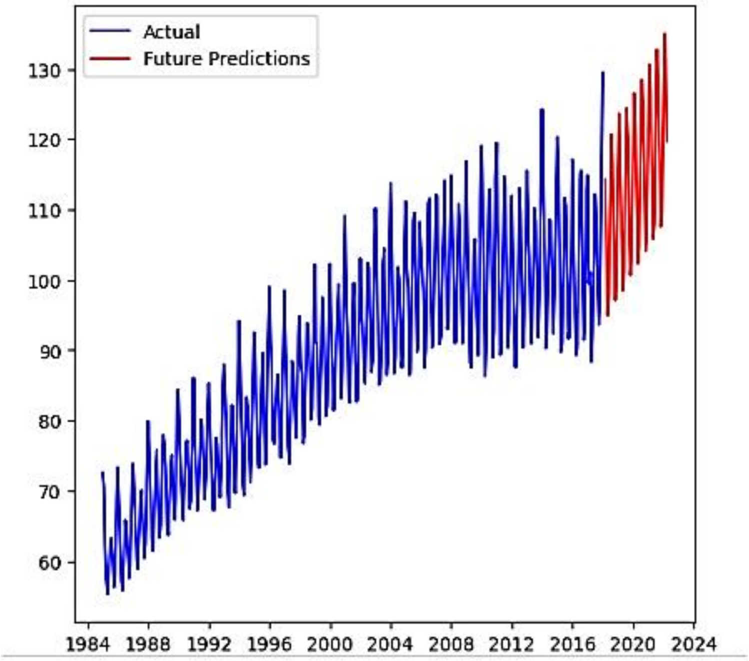
Prediction



```
future_sequence = test_data_scaled[-sequence_length:]
future_sequence = future_sequence.reshape((1, sequence_length, 1))
future_predictions = []
for _ in range(50):
    future_prediction = rnn.predict(future_sequence)
    future_predictions.append(future_prediction[0, 0])
    future_sequence = np.concatenate([future_sequence[:, 1:, :],
                                      future_prediction.reshape((1, 1, 1))], axis=1)
future_predictions = scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))
plt.figure(figsize=(6, 6))
plt.plot(data.index, data[target_variable], label='Actual', color='blue')
plt.plot(pd.date_range(test_data.index[-1], periods=len(future_predictions) + 1,
                       freq='M')[1:], future_predictions, label='Future Predictions', color='red')
plt.legend()
plt.show()
```



```
1/1 [==============================] - 0s 239ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 76ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 73ms/step
1/1 [==============================] - 0s 92ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 22ms/step
```


Actual
Future Predictions