Neural network-based time series forecasting model

Aim:

To write a python program for implementing a neural network-based time series forecasting model.

Algorithm:

1. The code imports the necessary libraries and loads the "Air Passengers" dataset from a GitHub source.
2. It converts the 'Month' column to datetime format, sets it as the index, and visualizes the raw time series data.
3. The dataset is normalized using a MinMaxScaler and transformed into a supervised learning dataset by creating look-back windows.
4. The processed data is split into training and testing sets, with inputs reshaped into three-dimensional arrays suitable for LSTM.
5. An LSTM-based model is constructed using Keras, incorporating an LSTM layer with 50 units and a Dense output layer, compiled with the Adam optimizer and mean squared error loss.
6. The model is trained on the training data over a specified number of epochs with a defined batch size, using the test set for validation.
7. Predictions are made on the test set, and both predictions and actual values are inverse-transformed to their original scale.
8. Finally, the actual versus predicted values are plotted to evaluate the forecasting performance of the model.

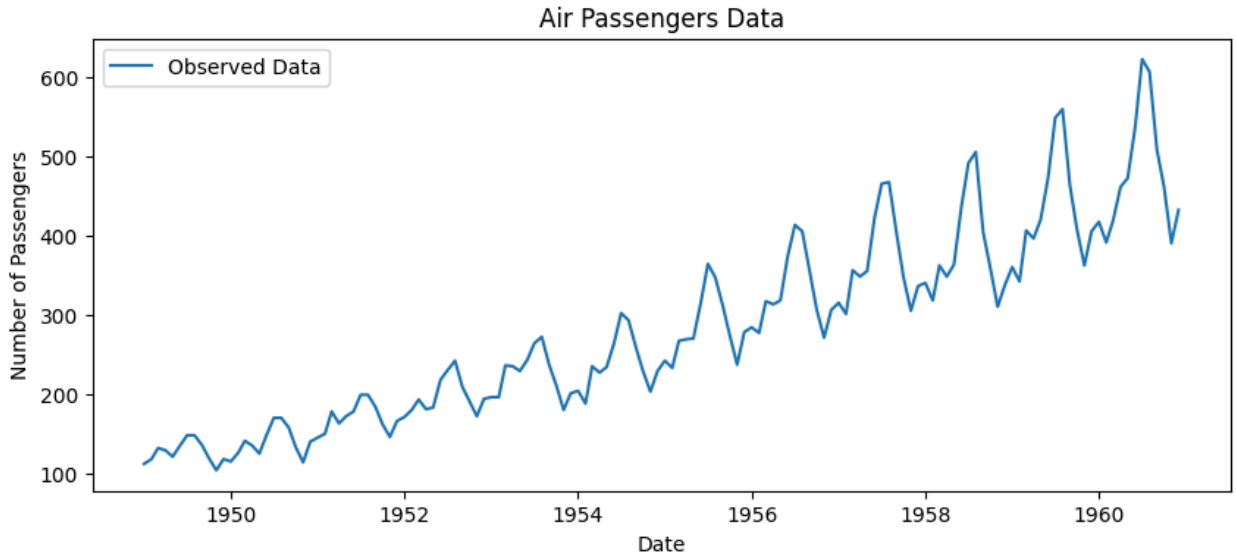
Program Code:

```
# Import necessary libraries
import warnings
warnings.filterwarnings("ignore")
```

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

# For neural networks using TensorFlow/Keras
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler

# Load the Air Passengers dataset from GitHub
data_url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv'
df = pd.read_csv(data_url)

# Convert the 'Month' column to datetime and set it as the index
df['Month'] = pd.to_datetime(df['Month'])
df.set_index('Month', inplace=True)
df.columns = ['Passengers']  # Rename for clarity

# Display the first few rows
print("Dataset head:")
print(df.head())

# Plot the raw time series
plt.figure(figsize=(10, 4))
plt.plot(df['Passengers'], label='Observed Data')
plt.title('Air Passengers Data')
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.legend()
plt.show()
```

```
Dataset head:
            Passengers
Month
```

1949-01-01 112
1949-02-01 118
1949-03-01 132
1949-04-01 129
1949-05-01 121



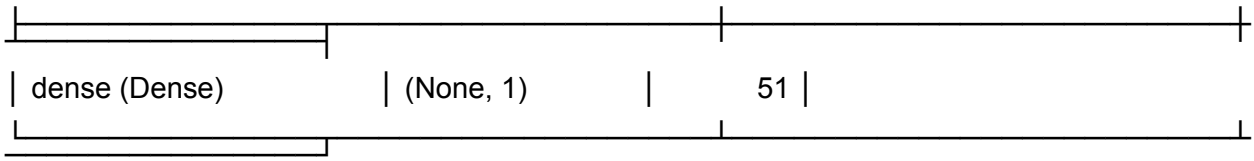

```
# Normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(df.values)

# Function to create a supervised learning dataset from time series
def create_dataset(dataset, look_back=1):
    """
    This function transforms a time series into a supervised learning format.
    Each X sample contains `look_back` consecutive time steps and the y is the next time step.
    """
    X, y = [], []
    for i in range(len(dataset) - look_back):
        X.append(dataset[i:(i + look_back), 0])
        y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(y)
```

```
# Set look_back period (number of past observations to use for predicting the next value)
look_back = 12  # using the previous 12 months to predict the next month
X, y = create_dataset(data_scaled, look_back)

# Reshape input to [samples, time_steps, features] for LSTM (features=1)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Split data into training and test sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

```
Training samples: 105
Testing samples: 27
```

```
# Build the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Display the model summary
model.summary()
```

```
Model: "sequential"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ (None, 50)             │        10,400 │
```

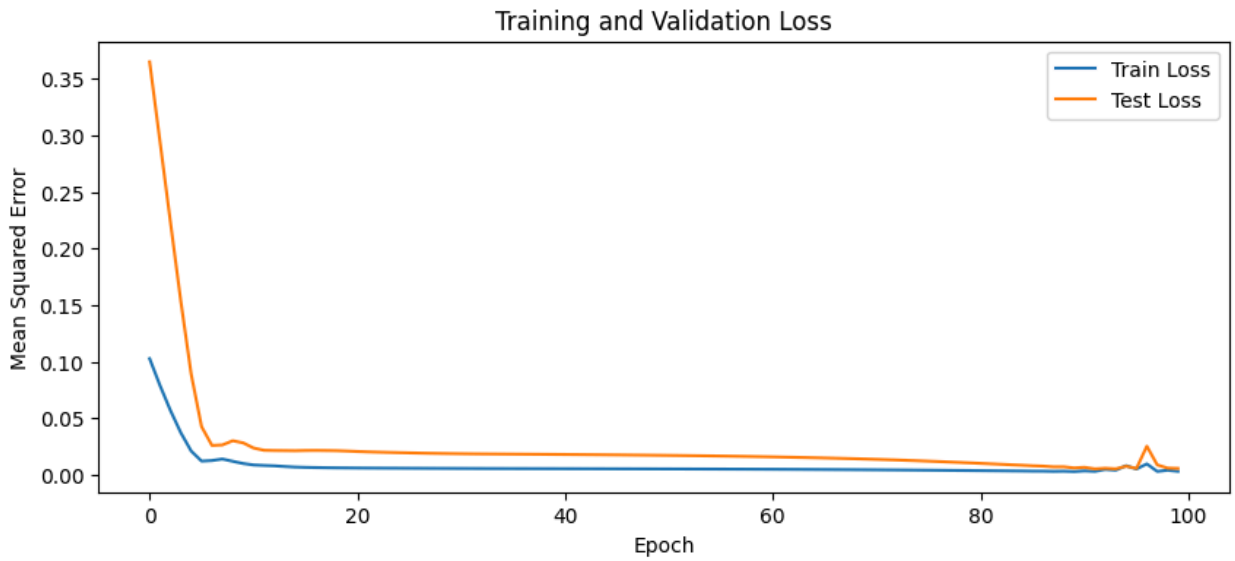


Total params: 10,451 (40.82 KB)

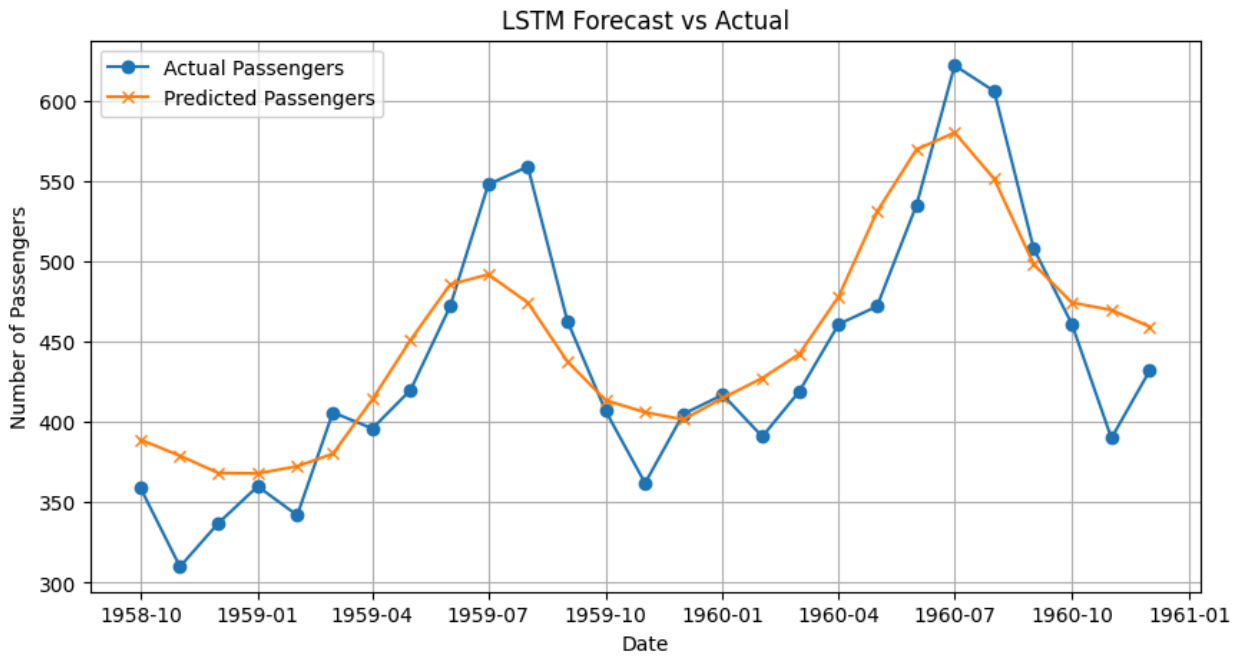
Trainable params: 10,451 (40.82 KB)

Non-trainable params: 0 (0.00 B)

```
# Train the model
epochs = 100
batch_size = 16

history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
                    validation_data=(X_test, y_test), verbose=1, shuffle=False)

# Plot training history
plt.figure(figsize=(10, 4))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
```




```
# Predict on test data
y_pred = model.predict(X_test)

# Inverse-transform predictions and true values to original scale
y_pred_inv = scaler.inverse_transform(y_pred.reshape(-1, 1))
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))

# Create a time index for test set plotting (starting after training period)
test_index = df.index[look_back + train_size: len(df)]

# Plot actual vs predicted values
plt.figure(figsize=(10, 5))
plt.plot(test_index, y_test_inv, label='Actual Passengers', marker='o')
plt.plot(test_index, y_pred_inv, label='Predicted Passengers', marker='x')
plt.title('LSTM Forecast vs Actual')
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.legend()
plt.grid(True)
plt.show()
```



RESULTS:

The program has been created and implemented successfully for creating a Neural network-based time series forecasting model.