EX:No.9 DATE:12/04/25	**Develop neural network-based time series forecasting model.**

AIM:

To develop a neural network-based time series forecasting model using LSTM on AAPL stock data.

ALGORITHM:

- Import AAPL stock data and necessary libraries (Pandas, Keras, etc.).
- Preprocess the data: sort by date, normalize, and create sequences.
- Split the dataset into training and testing sets.
- Reshape the data to fit LSTM input format: [samples, time_steps, features].
- Build the LSTM model with layers and compile it.
- Train the model using the training set with validation split.
- Predict on test data, inverse-transform predictions, and evaluate performance.

CODE:

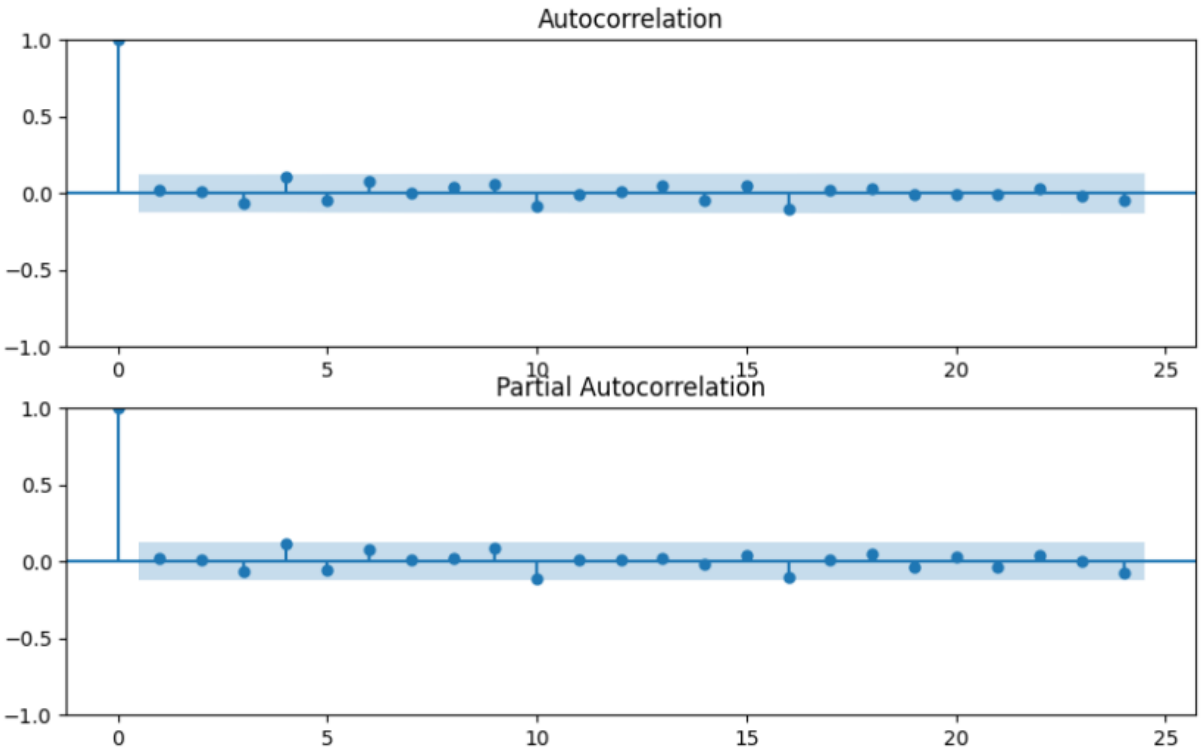
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping


# 1. Load the dataset
data = pd.read_csv('/content/AAPL.csv')
data['Date'] = pd.to_datetime(data['Date'])
data = data.sort_values('Date')
```

```
data.set_index('Date', inplace=True)

# 2. Preprocess - use only 'Close' prices
close_data = data['Close'].values.reshape(-1, 1)
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(close_data)

# 3. Create sequences for LSTM
def create_sequences(data, window_size):
    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i + window_size])
        y.append(data[i + window_size])
    return np.array(X), np.array(y)

window_size = 60
X, y = create_sequences(scaled_data, window_size)

# 4. Split into train and test sets
split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# 5. Build the LSTM model
model = Sequential()
model.add(LSTM(units=64, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=32))
model.add(Dropout(0.2))
```

```
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')

# 6. Train the model
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=50, batch_size=16,
                    validation_split=0.1, callbacks=[early_stop], verbose=1)

# 7. Predict and inverse scale
predicted = model.predict(X_test)
predicted_prices = scaler.inverse_transform(predicted)
actual_prices = scaler.inverse_transform(y_test)

# 8. Evaluation Metrics
mae = mean_absolute_error(actual_prices, predicted_prices)
mse = mean_squared_error(actual_prices, predicted_prices)
rmse = np.sqrt(mse)

print(f"\n LSTM Forecasting Metrics:")
print(f"MAE: {mae:.4f}")
print(f"MSE: {mse:.4f}")
print(f"RMSE: {rmse:.4f}")

# 9. Plot predictions vs actual
plt.figure(figsize=(12, 6))
plt.plot(actual_prices, label='Actual Prices')
plt.plot(predicted_prices, label='Predicted Prices')
```

```
plt.title('AAPL Stock Price Prediction using LSTM')
plt.xlabel('Days')
plt.ylabel('Price')
plt.legend()
plt.show()

# 10.Plot training history
plt.figure(figsize=(10, 4))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

OUTPUT:



RESULT:

Thus the program has been completed and verified successfully.