**Assignment no 5**

**Title**: Write a program to predict stock market trends using an artificial neural network.

**Objective**: The aim of this practical assignment is to develop an Artificial Neural Network (ANN) model to predict stock market trends, specifically to identify whether the stock price will rise or fall the following day based on historical price data and derived features.

**Theory**:

**Introduction**

Stock market trend prediction is a challenging task due to highly volatile and nonlinear price movements. Machine learning approaches like Artificial Neural Networks have shown promise by capturing complex patterns in the data. This project implements a feedforward ANN for binary classification of stock price movement (up or down) using feature engineering and historical trading data.

Artificial Neral Network:

An artificial neural network (ANN) is a computational model inspired by the human brain's structure, used in machine learning to recognize patterns and make predictions. It consists of interconnected nodes, or "neurons," organized in layers (input, hidden, and output) that process information and learn from data through a process called backpropagation. ANNs are used in AI for tasks like image recognition, natural language processing, and forecasting.

**How it works**

- Structure: ANNs are built with layers of artificial neurons.
    - Input Layer: Receives raw data from the outside world.
    - Hidden Layer(s): One or more layers where most of the processing occurs. Each layer transforms the output of the previous layer and passes it to the next.
    - Output Layer: Provides the final result of the network's processing.
- Connections: Each connection between neurons has a "weight," which determines the strength of the signal passing through it.
- Learning: The network "learns" by adjusting these weights through a process called training.
    - The network is fed examples (data) and compares its output to the correct answer.
    - Using an algorithm like gradient descent, it backpropagates the error (the difference between the predicted and actual output) through the network to adjust the weights.
    - The goal is to minimize this error, making the network's predictions more accurate over time.

**Applications**

- Pattern Recognition: Identifying patterns in medical images (like X-rays) or other complex datasets.
- Forecasting: Making predictions in business, finance, and other scientific fields.
- Natural Language Processing: Powering applications like AI chatbots and translation services.
- Image anad Speech Recognition: Enabling technologies to recognize objects in photos or understand spoken commands.

**Limitations**

- Data Dependency: They require large datasets for effective training.
- Computational Cost: Training complex networks can be computationally expensive and time-consuming.
- Overfitting: A model may become too complex and "overfit" the training data, performing poorly on new, unseen data.

**Methodology**

- Data Collection: Historical stock data containing daily open, high, low, close prices and volume was used.
- Feature Engineering: Features such as difference between open and close prices, high and low prices, along with normalized price and volume data were created to enhance predictive capability.
- Target Variable: The target was binary, set to 1 if the next day's close price is greater than the current day's close price, otherwise 0.
- Data Preprocessing: Features were scaled using MinMaxScaler to normalize values for better ANN performance.
- Model Design: A simple multilayer feedforward neural network was built with two hidden layers using ReLU activation and a sigmoid output layer for binary classification.
- Training: The model was trained using binary cross-entropy loss and the Adam optimizer.

**Results**

The ANN model successfully learned to predict next-day price movement with an accuracy around the mid to high 70% range on the validation set after 50 epochs. Training and validation accuracy and loss curves showed reasonable convergence without severe overfitting, indicating effective learning from features.

**Conclusion**

This exercise demonstrates the applicability of Artificial Neural Networks in financial time series prediction. While the model can predict general market trends with moderate accuracy, further improvements could be achieved by incorporating more complex architectures like LSTMs for sequence modeling, additional technical indicators, or integration of macroeconomic factors. Nonetheless, this approach highlights the potential of AI in aiding trading decisions through data-driven methods.

**Code:**

```
import numpy as np
import pandas as pd
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Step 1: Load stock data
ticker = 'AAPL'
df = yf.download(ticker, start='2020-01-01', end='2023-01-01')
df['Target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)

# Step 2: Feature selection and scaling
features = df[['Open', 'High', 'Low', 'Close', 'Volume']]
scaler = MinMaxScaler()
X = scaler.fit_transform(features)
y = df['Target'].values

# Step 3: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Step 4: Build ANN model
model = Sequential()
model.add(Dense(64, input_dim=5, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)

# Step 5: Evaluate model
y_pred = (model.predict(X_test) > 0.5).astype("int32")
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
```

**Output:**

Accuracy: 0.59
Confusion Matrix:
[[51 29]
 [34 38]]