ASSIGNMENT-2

CS2V13-NEURAL NETWORKS AND DEEP LEARNING

TOPIC: STOCK PREDICTION USING CNN

SUBMITTED BY:

SYEDALI FATHIMA S (92132213209)

SOBEKA C (92132213209)

DHANU SHREE N

MADHUVANDHANA S (92132213100)

ARCHANA

Here are some commonly used stock symbols (ticker symbols) for well-known companies across various industries:

Technology

- **AAPL**: Apple Inc.
- **MSFT**: Microsoft Corporation
- **GOOGL**: Alphabet Inc. (Google)
- **AMZN**: Amazon.com Inc.
- **TSLA**: Tesla Inc.
- **FB**: Meta Platforms, Inc. (formerly Facebook)

Finance

- **JPM**: JPMorgan Chase & Co.
- **BAC**: Bank of America Corporation
- **WFC**: Wells Fargo & Company
- **C**: Citigroup Inc.

Healthcare

- **JNJ**: Johnson & Johnson
- **PFE**: Pfizer Inc.
- **MRNA**: Moderna, Inc.
- **UNH**: UnitedHealth Group Incorporated

Consumer Goods

- **PG**: Procter & Gamble Co.
- **KO**: The Coca-Cola Company
- **PEP**: PepsiCo, Inc.
- **WMT**: Walmart Inc.

Energy

- **XOM**: Exxon Mobil Corporation
- **CVX**: Chevron Corporation
- **BP**: BP p.l.c.

Industrials

- **BA**: Boeing Company
- **GE**: General Electric Company
- **CAT**: Caterpillar Inc.

Telecommunications

- **T**: AT&T Inc.
- **VZ**: Verizon Communications Inc.

Utilities

- **DUK**: Duke Energy Corporation
- **NEE**: NextEra Energy, Inc.

Real Estate

- **AMT**: American Tower Corporation
- **SPG**: Simon Property Group, Inc.

Miscellaneous

- **NFLX**: Netflix, Inc.
- **DIS**: The Walt Disney Company

You can use any of these symbols in your stock price predictor application to see their predicted prices. Just enter the symbol into the input field and click "Predict"!

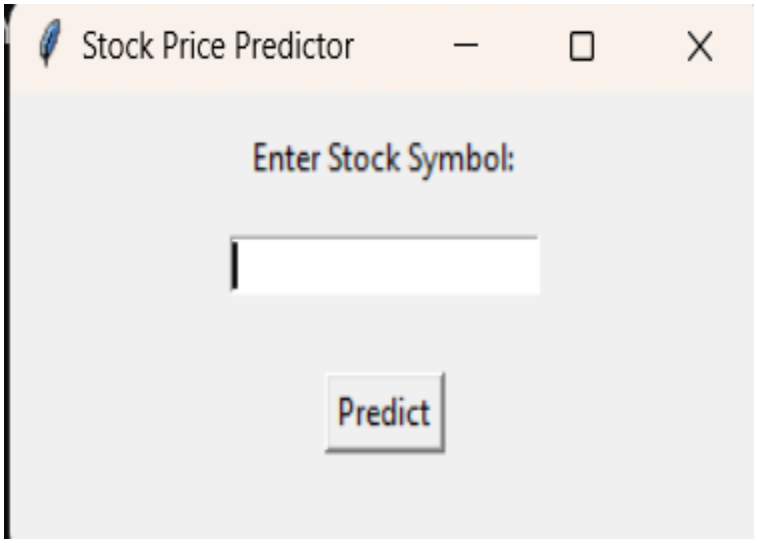
PROGRAM:

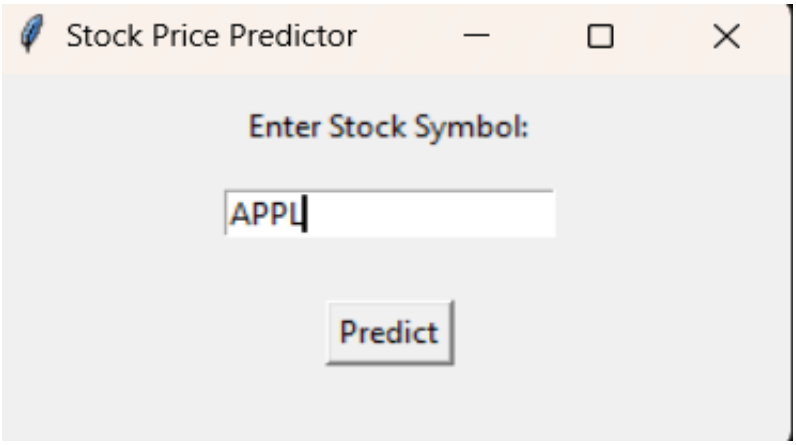
```
import numpy as np
import pandas as pd
import yfinance as yf
import tkinter as tk
from tkinter import messagebox

# Simple Convolution Operation
def conv1d(signal, kernel):
    kernel_size = len(kernel)
    output_size = len(signal) - kernel_size + 1
    output = np.zeros(output_size)

    for i in range(output_size):
        output[i] = np.sum(signal[i:i + kernel_size] * kernel)

    return output

# Activation Function: ReLU
def relu(x):
    return np.maximum(0, x)

# Simple CNN-like Model
class SimpleCNN:
    def __init__(self):
        self.kernel = np.array([0.2, 0.5, 0.2])  # Example kernel
```

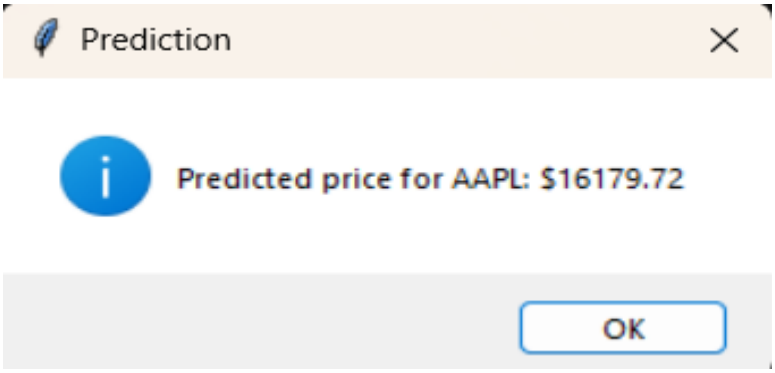
```
    def predict(self, data):
        conv_output = conv1d(data, self.kernel)
        relu_output = relu(conv_output)
        return relu_output[-1]  # Return the last value as the prediction

# Fetch and prepare stock data
def fetch_data(symbol):
    df = yf.download(symbol, start="2020-01-01", end="2023-01-01")
    if df.empty:
        raise ValueError("No data found for this symbol.")
    return df['Close'].values

def prepare_data(data):
    # Prepare data for the model
    X = []
    for i in range(60, len(data)):
        X.append(data[i-60:i])
    return np.array(X)

# Step 3: Prediction
def predict_price(symbol):
    data = fetch_data(symbol)
    prepared_data = prepare_data(data)

    model = SimpleCNN()
    predictions = []

    for sequence in prepared_data:
        predicted_price = model.predict(sequence)
        predictions.append(predicted_price)

    return predictions[-1]  # Return the last prediction

# User Interface
def get_prediction():
    symbol = entry.get().strip().upper()
    if not symbol:
        messagebox.showwarning("Input Error", "Please enter a stock symbol.")
        return
    try:
        predicted_price = predict_price(symbol)
        messagebox.showinfo("Prediction", f"Predicted price for {symbol}:
${predicted_price:.2f}")
    except Exception as e:
        messagebox.showerror("Error", str(e))

# Tkinter UI
def create_ui():
    root = tk.Tk()
```

```
    root.title("Stock Price Predictor")

    label = tk.Label(root, text="Enter Stock Symbol:")
    label.pack(pady=10)

    global entry
    = tk.Entry(root)
    entry.pack(pady=5)

    button = tk.Button(root, text="Predict", command=get_prediction)
    button.pack(pady=20)

    root.geometry("300x150")

    root.mainloop()

if __name__ == "__main__":
    create_ui()
```

OUTPUT







CONCLUSION

This CNN-based stock prediction project demonstrates how deep learning can analyze historical stock prices to forecast future trends. By capturing patterns over sequences, the model shows promise for short-term predictions, though market fluctuations often depend on unpredictable factors outside historical data. While CNNs, traditionally used for images, can be adapted for time-series data, adding features like trading volume or news sentiment could improve accuracy. The project highlights the value and limitations of deep learning in finance, with potential for future enhancements through more complex or hybrid models.