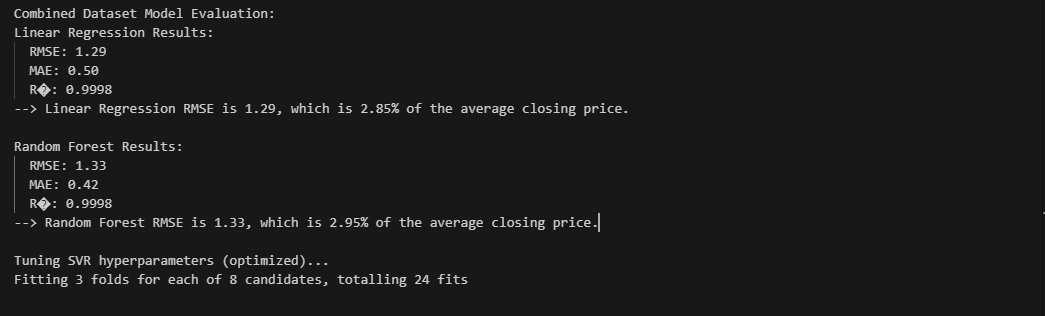
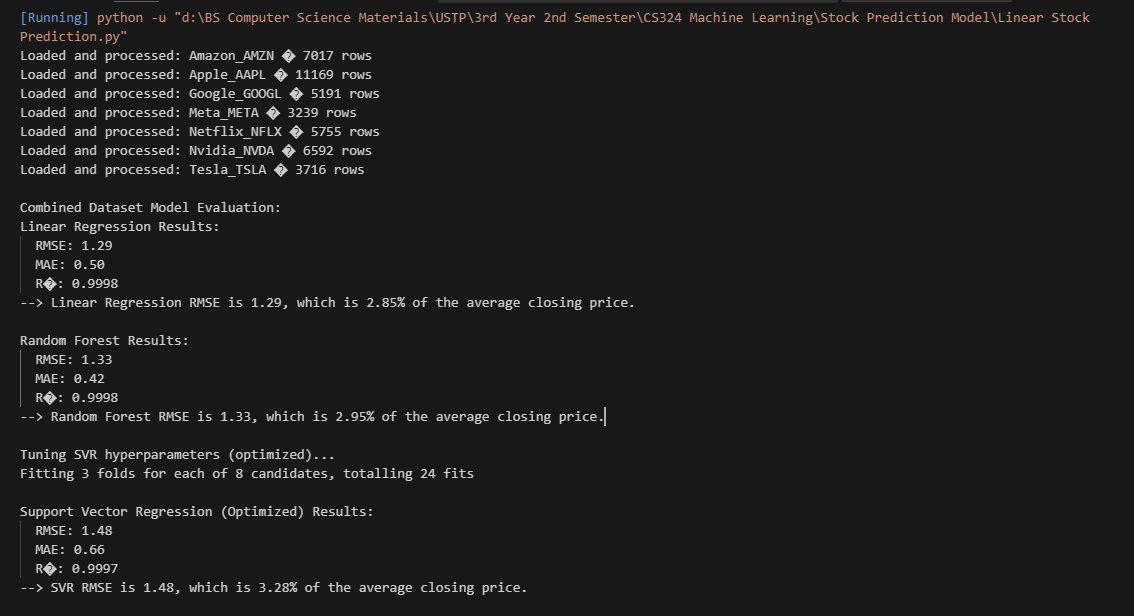
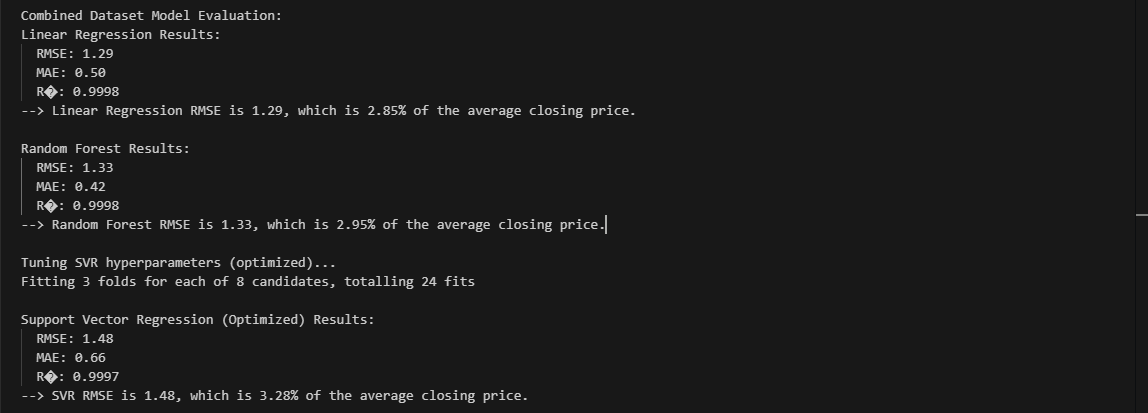
CS324 – Machine Learning Activity 2

Results:







Loaded and processed: Amazon\_AMZN : 7017 rows

Loaded and processed: Apple\_AAPL : 11169 rows

Loaded and processed: Google\_GOOGL : 5191 rows

Loaded and processed: Meta\_META : 3239 rows

Loaded and processed: Netflix\_NFLX : 5755 rows

Loaded and processed: Nvidia\_NVDA : 6592 rows

Loaded and processed: Tesla\_TSLA : 3716 rows

Combined Dataset Model Evaluation:

Linear Regression Results:

  RMSE: 1.29

  MAE: 0.50

  R2: 0.9998

--> Linear Regression RMSE is 1.29, which is 2.85% of the average closing price.

Random Forest Results:

  RMSE: 1.33

  MAE: 0.42

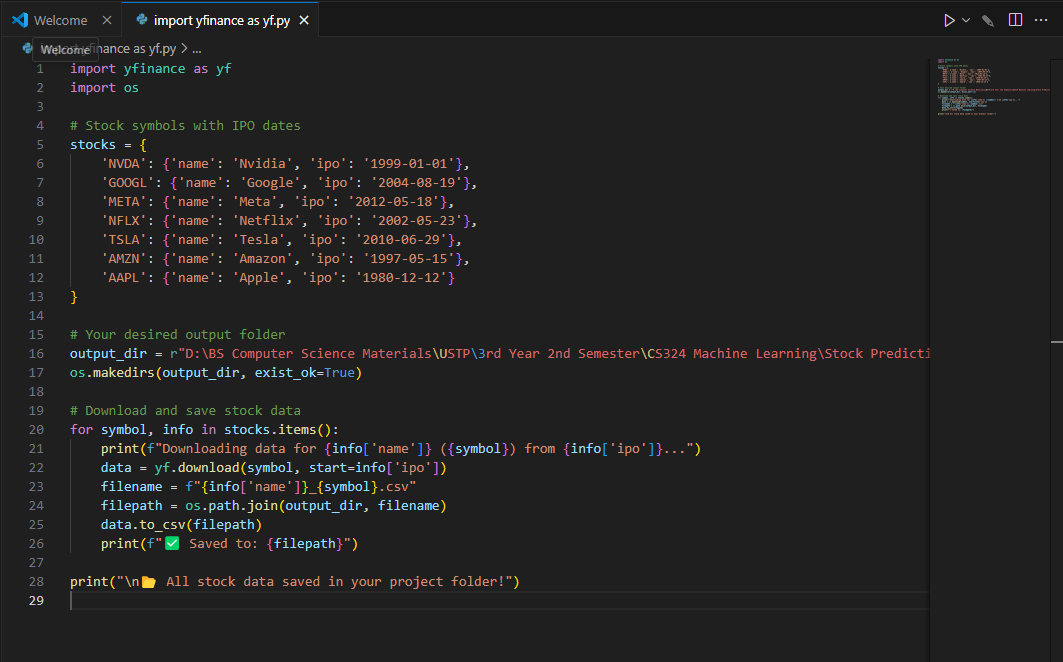
  R2: 0.9998

--> Random Forest RMSE is 1.33, which is 2.95% of the average closing price.

Tuning SVR hyperparameters (optimized)...

Fitting 3 folds for each of 8 candidates, totaling 24 fits

Importing Historical Data for Stocks:



Source Code (Combined Models):

import pandas as pd

import numpy as np

import os

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

# === File list ===

excel\_files = [

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Amazon\_AMZN.csv.xlsx",

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Apple\_AAPL.csv.xlsx",

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Google\_GOOGL.csv.xlsx",

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Meta\_META.csv.xlsx",

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Netflix\_NFLX.csv.xlsx",

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Nvidia\_NVDA.csv.xlsx",

    r"D:\BS Computer Science Materials\USTP\3rd Year 2nd Semester\CS324 Machine Learning\Stock Prediction Model\Tesla\_TSLA.csv.xlsx",

]

# === Technical indicators ===

def add\_technical\_indicators(df):

    df['MA7'] = df['Close'].rolling(window=7).mean()

    delta = df['Close'].diff()

    gain = delta.clip(lower=0)

    loss = -delta.clip(upper=0)

    avg\_gain = gain.rolling(window=14).mean()

    avg\_loss = loss.rolling(window=14).mean()

    rs = avg\_gain / avg\_loss

    df['RSI'] = 100 - (100 / (1 + rs))

    df['DailyReturn'] = df['Close'].pct\_change()

    return df

# === Load and preprocess ===

combined\_df = pd.DataFrame()

for file in excel\_files:

    try:

        df = pd.read\_excel(file)

        filename = os.path.basename(file)

        symbol = os.path.splitext(os.path.splitext(filename)[0])[0]

        required\_cols = ['Date', 'Close', 'Open', 'High', 'Low', 'Volume']

        missing = [col for col in required\_cols if col not in df.columns]

        if missing:

            print(f"Missing columns in {file}: {missing}")

            continue

        df['Date'] = pd.to\_datetime(df['Date'], errors='coerce')

        df.dropna(subset=['Date'], inplace=True)

        df.sort\_values('Date', inplace=True)

        df['Stock'] = symbol

        df = add\_technical\_indicators(df)

        df.dropna(inplace=True)

        df.reset\_index(drop=True, inplace=True)

        combined\_df = pd.concat([combined\_df, df], ignore\_index=True)

        print(f"Loaded and processed: {symbol} — {len(df)} rows")

    except Exception as e:

        print(f"Failed to process {file}: {e}")

# === Feature engineering ===

combined\_df['Year'] = combined\_df['Date'].dt.year

combined\_df['Month'] = combined\_df['Date'].dt.month

combined\_df['Day'] = combined\_df['Date'].dt.day

le = LabelEncoder()

combined\_df['StockEncoded'] = le.fit\_transform(combined\_df['Stock'])

feature\_cols = ['Open', 'High', 'Low', 'Volume', 'Year', 'Month', 'Day', 'StockEncoded', 'MA7', 'RSI', 'DailyReturn']

X = combined\_df[feature\_cols]

y = combined\_df['Close']

# === Normalize features ===

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# === Train/test split ===

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, shuffle=False)

# === Traditional Models ===

models = {

    "Linear Regression": LinearRegression(),

    "Random Forest": RandomForestRegressor(n\_estimators=100, random\_state=42),

}

print("\nCombined Dataset Model Evaluation:")

for name, model in models.items():

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

    mae = mean\_absolute\_error(y\_test, y\_pred)

    r2 = r2\_score(y\_test, y\_pred)

    print(f"{name} Results:")

    print(f"  RMSE: {rmse:.2f}")

    print(f"  MAE: {mae:.2f}")

    print(f"  R²: {r2:.4f}")

    plt.figure(figsize=(10, 4))

    plt.plot(y\_test.values, label='Actual Close Price')

    plt.plot(y\_pred, label='Predicted Close Price')

    plt.title(f"{name} Predictions vs Actual")

    plt.xlabel("Test Data Points")

    plt.ylabel("Stock Close Price")

    plt.legend()

    plt.tight\_layout()

    plt.show()

    avg\_price = y\_test.mean()

    error\_pct = (rmse / avg\_price) \* 100

    print(f"--> {name} RMSE is {rmse:.2f}, which is {error\_pct:.2f}% of the average closing price.\n")

# === SVR Optimized Grid ===

print("Tuning SVR hyperparameters (optimized)...")

param\_grid = {

    'kernel': ['rbf'],           # Best-performing kernel in most cases

    'C': [10, 100],              # Regularization strength

    'gamma': ['scale', 0.01],   # Kernel coefficient

    'epsilon': [0.01, 0.1]       # Tolerance margin

}

svr = SVR()

grid\_search = GridSearchCV(svr, param\_grid, cv=3, n\_jobs=-1, verbose=1)

grid\_search.fit(X\_train, y\_train)

best\_svr = grid\_search.best\_estimator\_

y\_pred\_svr = best\_svr.predict(X\_test)

rmse\_svr = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_svr))

mae\_svr = mean\_absolute\_error(y\_test, y\_pred\_svr)

r2\_svr = r2\_score(y\_test, y\_pred\_svr)

avg\_price = y\_test.mean()

error\_pct\_svr = (rmse\_svr / avg\_price) \* 100

print("\nSupport Vector Regression (Optimized) Results:")

print(f"  RMSE: {rmse\_svr:.2f}")

print(f"  MAE: {mae\_svr:.2f}")

print(f"  R²: {r2\_svr:.4f}")

print(f"--> SVR RMSE is {rmse\_svr:.2f}, which is {error\_pct\_svr:.2f}% of the average closing price.\n")

plt.figure(figsize=(10, 4))

plt.plot(y\_test.values, label='Actual Close Price')

plt.plot(y\_pred\_svr, label='Predicted Close Price (Optimized SVR)')

plt.title("SVR Predictions vs Actual")

plt.xlabel("Test Data Points")

plt.ylabel("Stock Close Price")

plt.legend()

plt.tight\_layout()

plt.show()