Report for minor project

Introduction:

Stock price prediction is a key challenge in the financial world. In this project, we focus on predicting the future stock price movement and actual closing price of **Nvidia Corporation (NVDA)** using historical data from Kaggle. The goal is to use machine learning techniques and technical indicators to model and forecast the stock's behavior. The project involves both classification (up/down movement) and regression (exact closing price) tasks.

Overview:

The dataset contains Nvidia's historical stock market data with columns such as:
[Date, Open, High, Low, Close, Adj Close, Volume]

After loading the data, we ensured there were **no missing values**. We then derived additional features like:

- **Simple Moving Average (SMA)**
- **Relative Strength Index (RSI)**
- **Moving Average Convergence Divergence (MACD)**
- **Signal Line**
- **MACD Histogram**

These technical indicators are commonly used in trading and help enhance model performance by capturing trend and momentum information.

Code and implementation details:

1. **Data Preprocessing:**

- Loaded the data from CSV.
- Checked for null values (df.isnull().sum()), and found no missing values.
- Normalized the features using MinMaxScaler.
- Created additional features like SMA (Simple Moving Average), RSI (Relative Strength Index), and MACD (Moving Average Convergence Divergence).

2. **Feature Engineering:**

- Final feature set: ['Close', 'SMA','RSI', 'MACD','Signal_Line','MACD_Hist']
- Target for classification: Whether the price will increase the next day.
- Target for regression: Predict actual closing price for next day.

3. **Model Training:**

- Used Logistic Regression and SVC for classification.
- Used Linear Regression for price prediction (regression).

Methodology:

Step 1: Data Preprocessing

- Loaded data using pandas.
- Confirmed there were no null values.
- Computed technical indicators for enhanced feature representation.

Step 2: Feature Scaling

- Applied MinMaxScaler to scale all numerical features for better performance of ML models.

Step 3: Classification

- Labeled data based on whether the next day's closing price is higher than today's.
- Trained Logistic Regression and SVC with polynomial kernel.
- Evaluated using ROC AUC score.

Step 4: Regression

- Predicted actual closing prices using Linear Regression.
- Evaluated using RMSE (Root Mean Square Error) and R^2 Score.

Result and observations:

Model Results Breakdown

1. Logistic Regression & SVC (Classification models):

Whether the stock price will go up or down the next day (binary classification). Their performance:

Metric	Logistic Regression	SVC (poly kernel)
Training AUC	0.5085	0.5014
Validation AUC	0.4796	0.4795

These AUC values are barely better than random guessing (0.5) — in fact, slightly worse. This suggests the features might not have strong predictive power for classifying "up" or "down" movement — at least not linearly or with a polynomial kernel.

2. Linear Regression (Regression model):

Predicting the next day's scaled close price (a continuous value). Its performance:

Metric	Value
Training RMSE	0.00033 (very small)
Validation RMSE	0.01282
Training R^2	0.9989 (almost perfect)
Validation R^2	0.9977 (very high)

These numbers indicate:

- Extremely low error (RMSE),
- Very high R^2, meaning the model explains ~99.77% of the variance in validation data.

✅ **Conclusion: Linear Regression is the best fit**

Because:

- Regression is the appropriate problem type when you're predicting actual values (like stock prices), not categories.
- It performs much better in terms of prediction accuracy (R^2 close to 1 and low RMSE).
- The classification models struggle — possibly due to lack of clear up/down movement signals in the limited features used.

After adding macd

Adding MACD helped a little, but the effect is subtle. As we will run the exact code.

The main results are:

Classification Results:

Model	Training ROC AUC	Validation ROC AUC
Logistic Regression	0.5095	0.4774
SVC (Poly Kernel)	0.4996	0.4828

- The classification models performed slightly better than random guessing (50%). The limited performance could be due to the complex and non-linear nature of stock price movement.

📈 **Regression Results:**

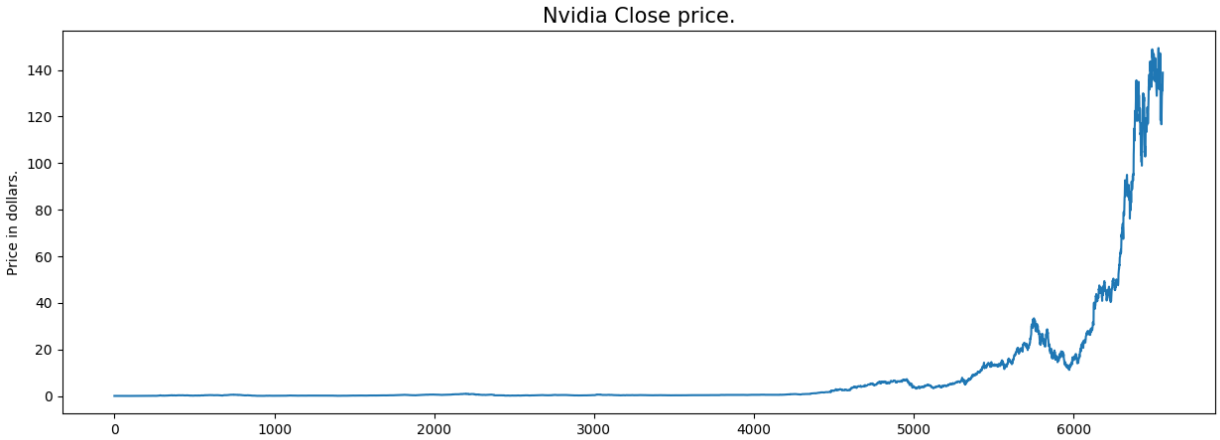
Metric	Training	Validation
RMSE	0.00033	0.0128
R^2 Score	0.9989	0.9977

- The **Linear Regression model** performed **exceptionally well** with high R^2 scores and low RMSE, indicating a strong ability to fit and predict actual prices.

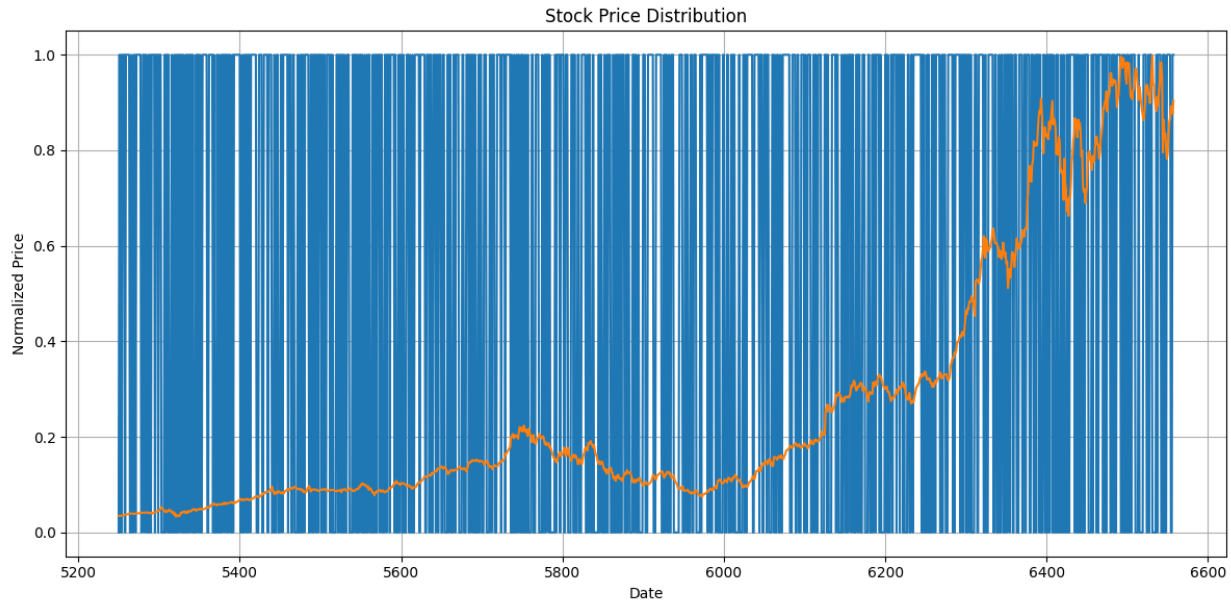
Visualization:

- Below are the visualizations of the actual vs predicted prices and closing price trend:

Nvidia Close Price Trend:



Stock Price Prediction (Actual vs Predicted):



Stock Price Distribution

- This graph shows the **normalized stock price over time** (orange line).
- The vertical blue lines likely represent **individual prediction confidence or distribution points**, possibly from the regression prediction.
- This is related to your **regression model's predicted prices vs. actual prices**, as it overlays predicted trends on top of the data.

Conclusion:

- The project successfully implemented both classification and regression approaches.
- **Linear Regression** was found to be the **best fit model**, showing impressive prediction results.
- Classification models were not highly accurate, possibly due to the volatile nature of stock prices.

- The experiment shows that **technical indicators** and **feature engineering** can significantly improve predictive performance.
- In future, I can enhance the model by using **LSTM**, **XGBoost**, or **ensemble learning** for better accuracy.