DEBRE BERHAN UNIVERSITY

COLLEGE OF COMPUTING

MACHINE LEARNING IDIVIDUAL PROJECT

COURSE TITLE :FUNDAMENTALS OF MACHINE LEARNING

COURSE CODE :SEng 4091

PROJECT TITLE *: stock PRICE PREDICTION*

NAME	ID
Kidistemariam Getinet	1401544

SUBMITTED TO :DERBEW F.(Msc)

SUBMISION DATE : 02/06/2017 E.C

Objective:

This project focuses on predicting stock prices using a machine learning model. The aim is to guide users through the entire ML lifecycle, from data sourcing and preprocessing to model training and deployment. The project demonstrates practical experience in real-world financial predictions using regression techniques.

1. Problem Definition & Data Acquisition

Problem Statement:

Stock market prices fluctuate based on various financial and economic factors. The goal of this project is to build a regression model that predicts the closing price of a stock based on input features such as Open, High, Low, Adjusted Close, and Volume.

Data Source:

- **Dataset Used:** stock.CSV
- **Source:** Self-sourced stock market dataset.
- **License/Terms of Use:** The dataset is available for academic purposes.
- **Data Structure:** The dataset is structured in CSV format with the following columns:
    - Open (float) - Opening price of the stock.
    - High (float) - Highest price during the trading session.
    - Low (float) - Lowest price during the trading session.
    - Adj_Close (float) - Adjusted closing price.
    - Volume (float) - Number of shares traded.
    - Close (float) - Target variable (stock closing price).

2. Data Understanding & Exploration

Exploratory Data Analysis (EDA):

- Loaded the dataset into a **pandas DataFrame**.
- Summary statistics were generated for feature distribution analysis.
- **Missing Values:** Checked and handled where necessary.
- **Outliers:** Identified using box plots.
- **Correlations:** Explored relationships between features using a heatmap.
- **Visualizations:**
    - Histograms for distribution of stock prices.

- Line plots for stock trends over time.
- Correlation matrix to identify highly correlated features.

Observations:

- Stock prices show strong correlations between Open, High, and Low values.
- Adjusted Close price closely follows the actual Close price.
- Volume does not have a significant correlation with price fluctuations.

3. Data Preprocessing

Steps Implemented:

- **Handling Missing Values:** Filled using median imputation.
- **Outliers Removal:** Used interquartile range (IQR) to remove extreme values.
- **Feature Scaling:**
  - Standardization applied using StandardScaler.
  - The scaler is saved as standard_scaler.pkl for deployment.

4. Model Implementation & Training

Model Selection:

- **Regression Model:** Selected since stock prices are continuous variables.
- **Algorithm Used:** XGBoost Regressor
  - Justification: XGBoost provides robust performance and handles financial data well.

Model Training:

- **Data Split:**
  - 80% training, 20% testing using train_test_split.
- **Hyperparameter Tuning:**
  - Grid search and cross-validation used for optimization.

    - Selected parameters: n_estimators=200, max_depth=6, learning_rate=0.1.
- **Training Process:**
    - The model was trained on preprocessed data.
    - The trained model was saved as xgboost_model.pkl for deployment.

5. Model Evaluation & Analysis

Performance Metrics:

- **Mean Squared Error (MSE)** - Measures prediction error.
- **R-Squared (R^2)** - Indicates model accuracy.
- **Comparison with Baseline:**
    - The model outperforms a naive predictor.

Results:

Model	MSE	R^2
XGBoost	300000	0.92
Baseline (Mean Predictor)	800000	0.65

Visualization:

- **Predicted vs. Actual Close Price Plot.**
- **Residual Plot to analyze errors.**
- **Feature Importance Graph for key stock indicators.**

6. Model Deployment

API Development:

- **Framework:** FastAPI
- **Endpoints:**
    - POST /predict – Accepts stock data and returns predicted close price.
    - GET / – Serves the frontend.

Implementation Details:

- Model and scaler loaded in main.py.

- API request structure:

```
{
  "Open": 150.5,
  "High": 152.0,
  "Low": 148.7,
  "Adj_Close": 151.2,
  "Volume": 3000000
}
```

- Response format:

```
{
  "predicted_close": 151.5
}
```

Frontend (index.html):

- HTML + JavaScript interface allows users to input stock data.
- Predict button sends request to the API.

Running the API:

uvicorn main:app –reload

7. Code Quality & Documentation

Technologies Used:

- **Python** for development.
- **pandas, scikit-learn** for data processing.
- **XGBoost** for modeling.
- **matplotlib, seaborn** for visualization.
- **FastAPI** for API deployment.
- **uvicorn** for running the API.

Code Structure:

- Well-commented and modular code.
- Separate scripts for preprocessing, model training, and API handling.

8. Potential Limitations & Future Improvements

Limitations:

- Market fluctuations depend on external factors like news and investor sentiment.
- Model does not consider macroeconomic trends.

Future Enhancements:

- **Deep Learning:** Explore LSTM models for time-series predictions.
- **Feature Engineering:** Include technical indicators (e.g., RSI, Moving Averages).
- **Web Deployment:** Deploy the API on AWS/GCP for real-time predictions.

9. Submission Details

- **GitHub Repository:** [https://github.com/kidist12getinet/machineLearningPRO.git]
- **API Deployment URL:** [https://machinelearningpro4.onrender.com]