Gold Price Prediction]

Name:danial baye
Id:1401105
A.name:Ml_Assignment

Gold Price Prediction Project: Documentation Report

Table of Contents

1. Introduction

Purpose of the Document

This document provides a comprehensive and detailed description of the Gold Price Prediction project, outlining the problem statement, data sources, model selection, and deployment approach. It is intended to serve as a guide for anyone interested in understanding the methodology behind the project.

Project Overview

The primary goal of this project is to develop a machine learning model capable of predicting daily gold closing prices based on historical data and market indicators. The model is trained using historical gold price data and various predictive features. The trained model is deployed as a web API using **FastAPI**, enabling users to make predictions in real-time.

High-Level Results

The best-performing model achieves a **Mean Absolute Error (MAE)** of X and an **R-squared (R2) score** of Y. However, the model's accuracy is influenced by external factors such as global market trends and geopolitical events.

2. Problem Definition & Data Acquisition

Problem Statement

Gold prices fluctuate due to various economic and geopolitical factors. This project aims to build a predictive model to estimate gold prices based on historical trends and market data.

Motivation

Accurate gold price prediction is crucial for investors, traders, and financial analysts. A reliable predictive model can help stakeholders make informed decisions and optimize their investment strategies.

Data Source

- **Dataset Name**: `goldstock v1.csv`
- **Number of Records**: Training Data: 1129, Test Data: 1015
- **Features Included**: Date, Volume, Open, High, Low, Closing Price, etc.
- **Data Source URL**: [URL Placeholder]

License/Terms of Use

The dataset used in this project is publicly available and does not have any restrictions on its usage.

---

3. Data Understanding & Exploration (EDA)

Loading and Inspecting Data

```
import pandas as pd
df = pd.read_csv('goldstock v1.csv')
df.head()
```

- `df.info()` provides insights into feature data types and missing values.
- `df.describe()` helps understand the statistical distribution of numerical features.

Handling Missing Values

- Identified missing values using `df.isnull().sum()`.
- Missing values were handled using mean imputation for numerical data and mode imputation for categorical data.

Exploratory Data Analysis (EDA)

Distribution Analysis

- Histograms and KDE plots were used to analyze feature distributions.

Outlier Detection

- Boxplots were used to identify and handle outliers.

Correlation Analysis

- A **correlation heatmap** was generated to identify relationships between numerical variables, especially the correlation between the target variable (`Close`) and input features.

4. Data Preprocessing

Handling Missing Values

- Rows with missing values were either removed or imputed depending on their significance.

Feature Engineering

- Extracted temporal features such as `year`, `month`, `day`, `weekday`, and `is_weekend` from the date column.

Encoding Categorical Data

- Categorical features were one-hot encoded where necessary.

Feature Scaling

- **StandardScaler** was applied to numerical features to normalize the data distribution and improve model performance.

Train-Test Split

- The dataset was split into **80% training** and **20% testing** using `train_test_split()` from `sklearn`.

---

5. Model Implementation and Training

Algorithm Selection

- **Random Forest Regressor** was chosen due to its robustness in handling nonlinear data and reducing overfitting through ensemble learning.

Hyperparameter Tuning

- GridSearchCV was used to optimize hyperparameters such as `n_estimators`, `max_depth`, and `min_samples_split`.

Model Training

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

6. Model Evaluation and Analysis

Evaluation Metrics

- **Mean Absolute Error (MAE)**: Measures the average absolute difference between actual and predicted values.
- **R2 Score**: Measures how well the model explains variance in the target variable.

Results

- **MAE**: X
- **R2 Score**: Y

Baseline Comparison

- The model's performance was compared against a **dummy regressor** to assess its predictive power beyond naive predictions.

7. Model Deployment

Deployment Framework

- The trained model was deployed using **FastAPI**, allowing for real-time predictions via a REST API.

API Structure

- **Endpoint**: `/predict` (Accepts POST requests with input features and returns predicted gold prices.)

How to Use the API

- Start API using `uvicorn app:app --reload`.
- Send a POST request with feature values in JSON format to get predictions.

---

8. Conclusion

Summary

- Successfully developed and deployed a gold price prediction model.
- Implemented robust preprocessing techniques to improve accuracy.

Limitations

- The model does not account for macroeconomic trends that significantly impact gold prices.

Future Work

- Improve feature selection by incorporating **macroeconomic indicators**.
- Explore **deep learning** approaches for better performance.

9. Required Technology Stack

- **Programming Language**: Python
- **Libraries Used**:
    - Data Processing: `pandas`, `numpy`
    - Machine Learning: `scikit-learn`
    - Data Visualization: `matplotlib`, `seaborn`
    - API Development: `FastAPI`, `Uvicorn`
    - Model Persistence: `joblib`

10. References

- Scikit-learn Documentation
- FastAPI Documentation
- [Dataset Source: URL Placeholder]
