Architecture Design

Investment Prediction

Written By	Tejas Pandav
Document Version	1.1
Last Revised Date	07/06/2024

DOCUMENT CONTROL

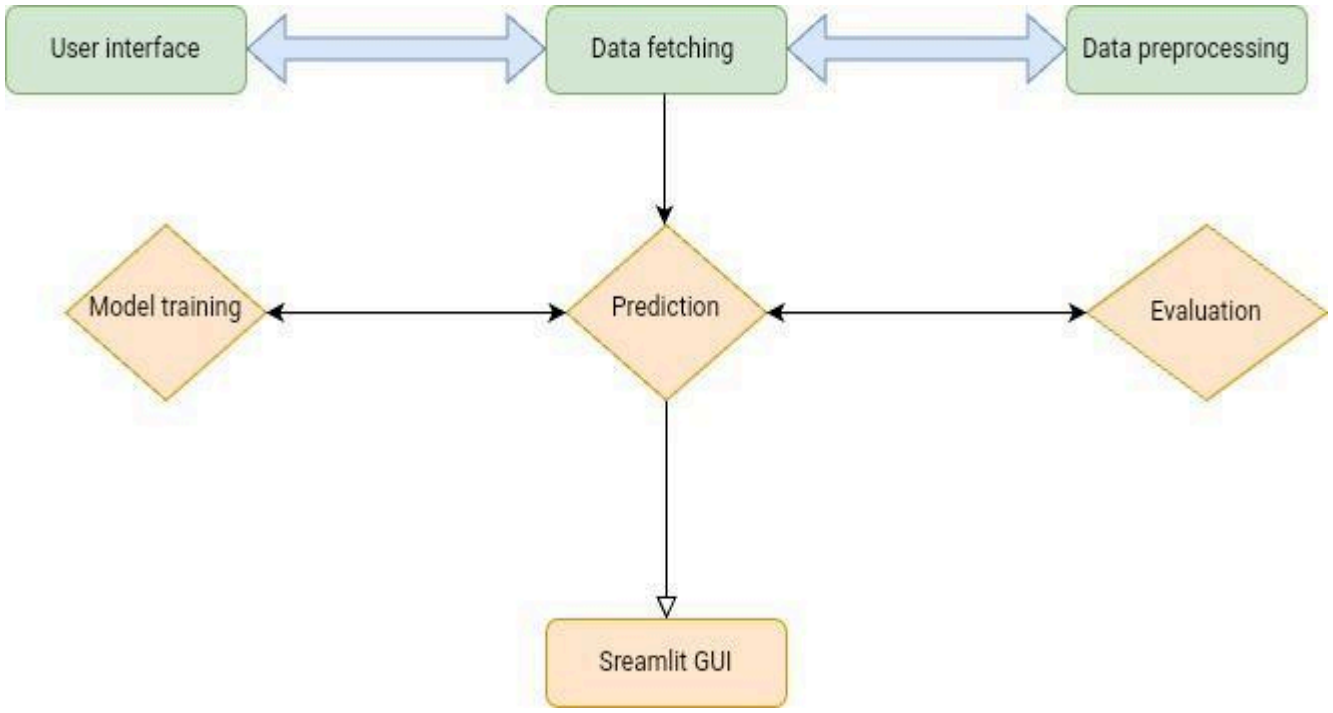
Change Record:

Date	Version	Description	Author
28th May 2024	1.0	Introduction, Architecture and deployment	Tejas Pandav
29th May 2024	1.1	Final Version of Complete Architecture	Tejas Pandav

Architecture Design for Stock Price Prediction Application :

1. Overview

This document describes the architecture design for the Stock Price Prediction application. The architecture is designed to be modular, scalable, and maintainable, with clear separation of concerns among different components. The architecture includes data ingestion, feature engineering, model training, prediction, and a user interface.



2. Architecture Components:

The architecture consists of the following key components:

1. Data Ingestion
2. Feature Engineering
3. Model Training
4. Prediction
5. User Interface
6. Error Handling and Logging
7. Deployment

2.1 Data Ingestion

Overview

This component is responsible for fetching historical stock data from Yahoo Finance. It ensures that the data is collected accurately and efficiently.

Technologies

- Yahoo Finance API (via `yfinance` library)
- Pandas for data manipulation

Process

1. User inputs the stock symbol, start date, and end date.
2. The application calls the Yahoo Finance API to fetch historical stock data.
3. Data is stored in a Pandas DataFrame for further processing.

Components

- `fetch_stock_data(symbol, start_date, end_date)`

2.2 Feature Engineering

Overview

This component processes the raw stock data to generate additional features that are useful for the machine learning model.

Technologies

- Pandas for data manipulation

Process

1. Calculate the Simple Moving Average (SMA) and Relative Strength Index (RSI) for the stock prices.
2. Add these features to the DataFrame.

Components

- `create_features(data)`
- `calculate_RSI(close_prices, window=14)`

2.3 Model Training

Overview

This component trains a machine learning model using the prepared features and historical stock prices.

Technologies

- XGBoost for regression modeling
- Scikit-learn for preprocessing and model evaluation

Process

1. Prepare the feature matrix (X) and target vector (y) from the enhanced DataFrame.
2. Train the XGBoost model using the training data.
3. Save the trained model for future predictions.

Components

- `prepare_data(data)`
- `train_model(X_train, y_train)`
- `save_model(model, filename)`

2.4 Prediction

Overview

This component makes future stock price predictions using the trained model.

Technologies

- XGBoost for making predictions

Process

1. Generate future dates for prediction.
2. Prepare the data for these dates using the same feature engineering techniques.
3. Use the trained model to predict future prices.

Components

- `predict_future_prices(model, data)`
- `predict_stock_price(model, X)`

2.5 User Interface

Overview

This component provides an interactive web interface for users to input data, view actual stock prices, see predictions, and get stock information.

Technologies

- Streamlit for building the web interface

Process

1. User navigates through different pages (Actual Value, Prediction, Ticker Information).
2. The application fetches data, processes it, and displays results based on user inputs.

Components

- Streamlit application with pages for actual values, predictions, and ticker information

2.6 Error Handling and Logging

Overview

This component handles errors gracefully and logs important events for debugging and monitoring purposes.

Technologies

- Python's logging module

Process

1. Capture and log errors during data fetching, model training, and predictions.
2. Display user-friendly error messages on the interface.

Components

- Custom error handling functions
- Logging configuration

2.7 Deployment

Overview

This component handles the deployment of the application on a web server, ensuring it is accessible to users.

Technologies

- Streamlit sharing for quick deployment
- Docker for containerization (optional)
- Cloud services (e.g., AWS, GCP, Azure) for scalable deployment

Process

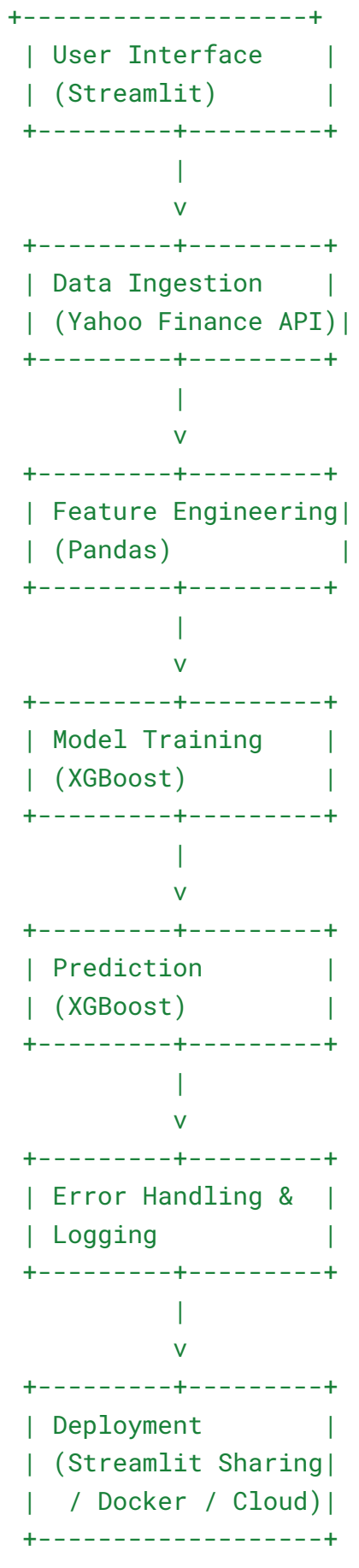
1. Package the application code and dependencies.
2. Deploy the application on a web server or cloud service.
3. Ensure the application is accessible via a web URL.

Components

- Dockerfile (optional)
- Deployment scripts

3. Architecture Diagram

The following diagram provides a visual representation of the architecture:



4. Conclusion

This architecture design ensures a modular and scalable approach to building the Stock Price Prediction application. Each component has a clear responsibility, and the use of well-established technologies and frameworks ensures robustness and maintainability. The architecture is designed to handle real-time data processing, efficient model training, accurate predictions, and an intuitive user interface.