Hedge Fund Application Project

Course: Big-Data Sys Engr Using Scala SEC 01 Fall 2024
Professor: Robin Hillyard
Team-3: Shuyan Bian, Xingxing Xiao, Caowang Sun

GitHub Link: https://github.com/starsbro/ScalaFinalProject.git

Table of contents

1. Project title with team members (and team number);
2. Use cases to be satisfied for hypothetical customers--see separate item on this under *Projects*;
3. Methodology (how/what do you propose to do);
4. Data sources, enumerating the magnitude (primarily the # of rows) of the data that you will process;
5. Milestones/sprints (approximately one week to ten days between each);
6. What will you program in Scala (and will there be any code *not* in Scala?) and where is your code repository?
7. Acceptance criteria (how will I know that you have achieved what you set out to do? -- see separate item);
8. Goals of the project (what do *you* expect to achieve and accomplish).

8. Goals of the Project

Real-Time Analysis: Create an application capable of providing accurate real-time trading recommendations based on market data.

→**Create DSL, Get Real-Time Data, and predict by Spark MLlib**

Scala and Akka Proficiency: Gain practical experience with real-time data processing and actor-based concurrency in Scala.

→ **Yes, we use Scala and Akka to fetch real-time data.**

2. Use cases

Hypothetical customers: financial professionals, investors, and fund managers, Data engineers, students

User will input the company stock symbol for prediction and model will provide the future stock prediction of the mentioned company

For example: User choose “MMM”, “APPL”, “GOOGL”, “TSLA” etc to get real-time price and the future stock prediction.

3. Methodology

How/what do you propose to do？

Create DSL for special buy/sell stock/option

Data cleaning and parsing

Spark and Spark MLlib for prediction

Hedge Fund application Package

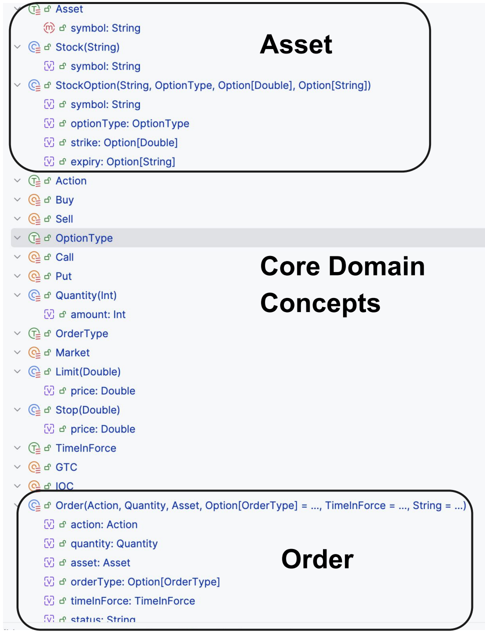
Dsl: Create main Domain Concepts and DslDemo

ApiClient: Try different Api (free) and Demo

MockExchange: Based on DSL, Demo mock exchange

Prediction: Use history and current stock/option price to predict and evaluate.

DSL(Domain-Specific Language)

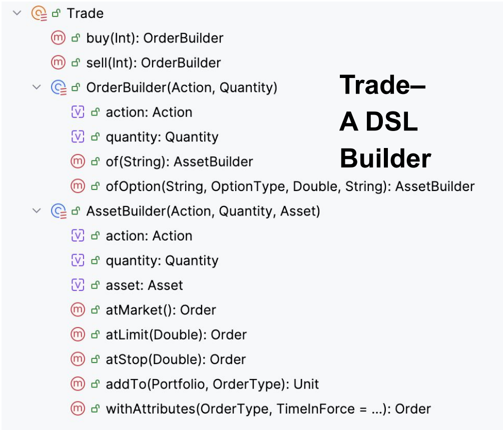


```
Asset
  symbol: String
Stock(String)
  symbol: String
StockOption(String, OptionType, Option[Double], Option[String])
  symbol: String
  optionType: OptionType
  strike: Option[Double]
  expiry: Option[String]
Action
Buy
Sell
OptionType
Call
Put
Quantity(Int)
  amount: Int
OrderType
Market
Limit(Double)
  price: Double
Stop(Double)
  price: Double
TimeInForce
GTC
IOC
Order(Action, Quantity, Asset, Option[OrderType] = ..., TimeInForce = ..., String = ...)
  action: Action
  quantity: Quantity
  asset: Asset
  orderType: Option[OrderType]
  timeInForce: TimeInForce
  status: String
```

Asset

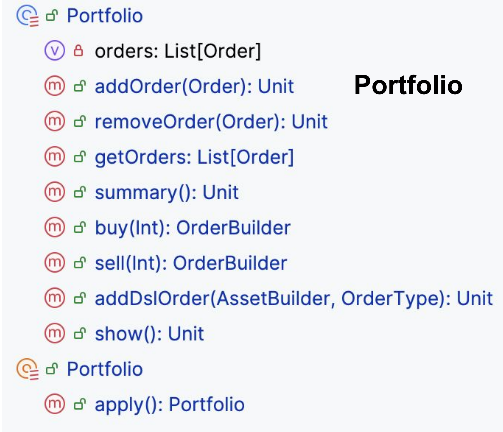
Core Domain Concepts

Order



```
Trade
  buy(Int): OrderBuilder
  sell(Int): OrderBuilder
  OrderBuilder(Action, Quantity)
    action: Action
    quantity: Quantity
    of(String): AssetBuilder
    ofOption(String, OptionType, Double, String): AssetBuilder
  AssetBuilder(Action, Quantity, Asset)
    action: Action
    quantity: Quantity
    asset: Asset
    atMarket(): Order
    atLimit(Double): Order
    atStop(Double): Order
    addTo(Portfolio, OrderType): Unit
    withAttributes(OrderType, TimeInForce = ...): Order
```

Trade– A DSL Builder



```
Portfolio
  orders: List[Order]
  addOrder(Order): Unit
  removeOrder(Order): Unit
  getOrders: List[Order]
  summary(): Unit
  buy(Int): OrderBuilder
  sell(Int): OrderBuilder
  addDslOrder(AssetBuilder, OrderType): Unit
  show(): Unit
Portfolio
  apply(): Portfolio
```

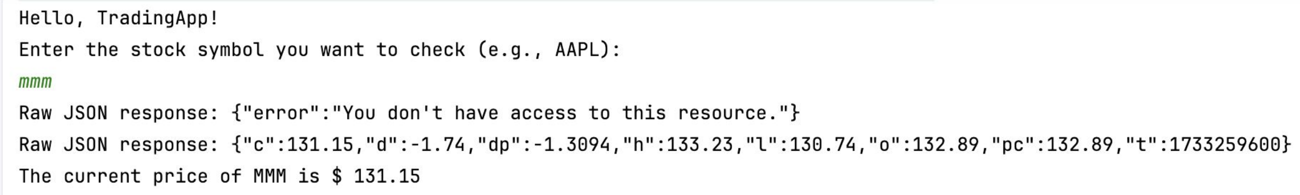
Portfolio

Define the Core Domain Concepts
Define Trade
Define Portfolio
DslDemo (Implement the DSL)
ApiDemo (Integrated with APIs)

DSL(Domain-Specific Language) Demo Result

```scala
private val buyOrder2 = portfolio.buy(50).of("GOOG").atLimit(1500.0)  // Limit order for 50 GOOG shares at $1500
```

```
Portfolio Details:

Buy 50 of Stock(GOOG) as Limit(1500.0)

Sell 100 of Stock(AAPL) as Stop(140.0)

Sell 300 of StockOption(MMM,Call,Some(132.0),Some(2024-12-31)) as Market

Buy 10 of StockOption(AAPL,Call,Some(150.0),Some(2024-12-15)) as Market

Buy 20 of StockOption(MMM,Call,Some(132.0),Some(2024-12-31)) as Limit(145.0)
```



```
Hello, TradingApp!
Enter the stock symbol you want to check (e.g., AAPL):
mmm
Raw JSON response: {"error":"You don't have access to this resource."}
Raw JSON response: {"c":131.15,"d":-1.74,"dp":-1.3094,"h":133.23,"l":130.74,"o":132.89,"pc":132.89,"t":1733259600}
The current price of MMM is $ 131.15
```

Use Finnhub API to fetch real-time stock price

Real-Time Market Data Analysis: Provide ongoing analysis of stock and option prices, detecting opportunities and potential risks in real time.

We choose AlphaVantage, YahooFinance, and Finnhub API to get real-time data, but at last just Finnhub work. The result as below:

Raw JSON response:
{"c":131.15,"d":-1.74,"dp":-1.3094,"h":133.23,"l":130.74,"o":132.89,"pc":132.89,"t":1733259600}

“c” // Current price;   “d” // Price change

“dp” //Percentage change; “H” //High price of the day

“l” // low price of the day;   “o” //Opening price of the day

“pc” // Previous close price, “t” //Timestamp

4.Data sources

API: fetch real-time data

Main Data Source (from Kaggle): Simulate stock and option price data streams. Each data point will contain the following fields: symbol, price, timestamp, and assetType.

https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks/data

The data source we used was taken from Kaggle which has about **1886753** rows for 500 Companies over the period of ten years

Historical Data Source (backup): Incorporate open financial datasets (like Yahoo Finance) for backtesting if time permits. –No use this data source.

5. Milestones/sprints

Week 1: Project setup, mock data generator, and basic data ingestion. –Done

Week 2: Implementation of moving averages and volatility calculations.–Done

Week 3: Initial version of the recommendation engine (buy, sell, hold logic). –Not yet

Week 4: Backtesting functionality and basic UI or reporting tools, Testing, refinement, and documentation. –Partial completed

6. What will you program in Scala ?

(and will there be any code not in Scala?) and where is your code repository?

Programming Language: Primarily in Scala, using libraries like Akka for data streaming, Breeze for calculations, and Spray JSON for serialization.

Additional Code: If necessary, some auxiliary tools may be written in shell scripts (e.g., for data preprocessing).

→ **At last, just use scala, no other language.**

Repo URL:
https://github.com/starsbro/ScalaFinalProject.git

7. Acceptance Criteria

Functionality: The application must be able to receive mock data in real time, process it, and output trading recommendations.

→Due to time constraints, We haven’t completed the UI yet, and currently using IDE to input and output some mock data.

Performance: The application should handle at least 1 million records in real time without significant lag.

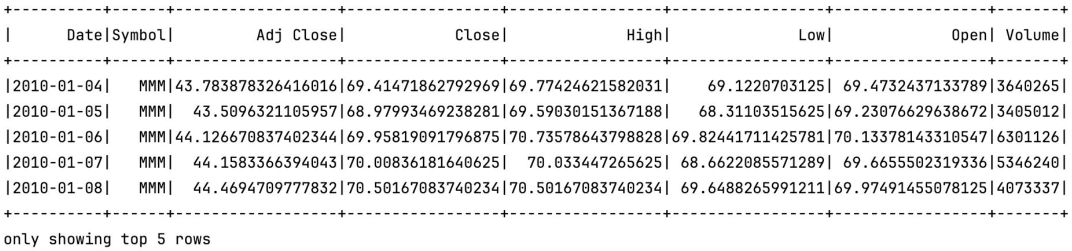
→ Not meet, total number of rows in the dataset: 1886753 > 1 million, but show significant lag.

Stock Price Prediction

To use the dataset from Kaggle for stock price prediction using **Apache Spark MLlib** in **Scala**, we'll focus on predicting the **Closing Price** (Close) based on the other available features like **Open**, **High**, **Low**, and **Volume**.

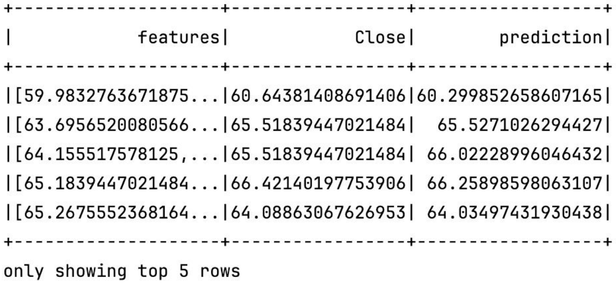
Split data into training (80%) and test (20%) sets. Or you can try other ratio.

Input DataSet Head



```
+----------+------+------------------+-----------------+-----------------+-----------------+-----------------+-------+
|      Date|Symbol|         Adj Close|            Close|             High|              Low|             Open| Volume|
+----------+------+------------------+-----------------+-----------------+-----------------+-----------------+-------+
|2010-01-04|   MMM|43.783878326416016|69.41471862792969|69.77424621582031|    69.1220703125| 69.4732437133789|3640265|
|2010-01-05|   MMM|  43.5096321105957|68.97993469238281|69.59030151367188|   68.31103515625|69.23076629638672|3405012|
|2010-01-06|   MMM|44.126670837402344|69.95819091796875|70.73578643798828|69.82441711425781|70.13378143310547|6301126|
|2010-01-07|   MMM|  44.1583366394043|70.00836181640625|  70.033447265625| 68.6622085571289| 69.6655502319336|5346240|
|2010-01-08|   MMM|  44.4694709777832|70.50167083740234|70.50167083740234| 69.6488265991211|69.97491455078125|4073337|
+----------+------+------------------+-----------------+-----------------+-----------------+-----------------+-------+
only showing top 5 rows
```

Output Predicting closing Price

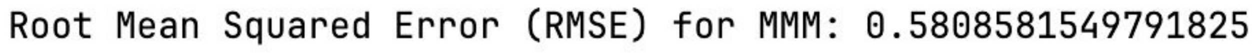


```
+--------------------+-----------------+------------------+
|            features|            Close|        prediction|
+--------------------+-----------------+------------------+
|[59.9832763671875...|60.64381408691406|60.299852658607165|
|[63.6956520080566...|65.51839447021484|  65.5271026294427|
|[64.155517578125,...|65.51839447021484| 66.02228996046432|
|[65.1839447021484...|66.42140197753906| 66.25898598063107|
|[65.2675552368164...|64.08863067626953| 64.03497431930438|
+--------------------+-----------------+------------------+
only showing top 5 rows
```

We used in Apache Spark MLlib to combine multiple feature columns into a single vector column for machine learning tasks, and get closing price. It looks not bad. How to evaluate this result?

Evaluate Prediction

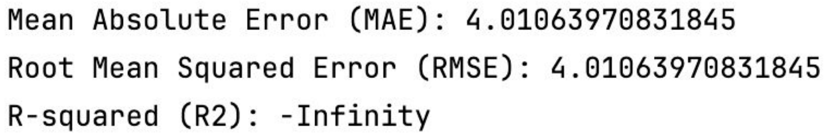
To evaluate prediction results using metrics like **MAE (Mean Absolute Error)**, **RMSE (Root Mean Squared Error)**, and **R² (R-squared)** in Spark, we use the RegressionEvaluator class from the org.apache.spark.ml.evaluation package.



```
Root Mean Squared Error (RMSE) for MMM: 0.5808581549791825
```

MMM real-time price is 131.15, RMSE = 0.58, it looks good prediction for closing price.

After this, we predict the closing price of next day according to the historical price.



```
Mean Absolute Error (MAE): 4.01063970831845
Root Mean Squared Error (RMSE): 4.01063970831845
R-squared (R2): -Infinity
```

MMM real-time price is 131.15, RMSE = 4.01, it looks not too good, but not bad. prediction for the closing price of next day.

Evaluate Prediction

Predict the closing price for the next 7 days

MMM real-time price is 131.15, it looks not bad for closing price.

Predicted Closing prices for the next 7 days: 132.40014262955754, 132.8355244896731, 132.90667166428312, 132.91829805692532, 132.92019796379597, 132.92050843379542, 132.9205591687141.

We also predicted and evaluated other stocks (such as GOOG, AAPL, TSLA etc.). It is not always good, so we planned future task.

Future Task

1. Combine real-time data and history data to predict future price, and use DSL to create order to trade.
2. Optimizing the prediction using other model (Decision Tree Regressor, Random Forest Regressor).
3. Combining the result from all prediction, and getting more optimized prediction result.

Q & A

Thank you