**LAB2: TỔNG QUAN 7 BƯỚC THỰC HIỆN**

1. **Business Understanding & Data Collection**
2. **Data Description & Preprocessing**
3. **Data Analysis**
4. **Data Visualization (Advanced + XAI)**
5. **Chatbot Integration (Gemini & so sánh chatbox)**
6. **Model Building & Prediction**
7. **Application Development (Flask Dashboard + Chatbot)**

**Deadline 9/6/2025, nộp báo cáo Latex, code trên nộp google drive mỗi nhóm folder LAB2**

**1. BUSINESS UNDERSTANDING & DATA COLLECTION**

**1.1. Mục tiêu Bước 1**

* **Làm rõ** phạm vi: Xây một hệ thống end-to-end cho dự án (Stock/Weather/Healthcare) với:
  1. **Multi-modal data** (giá, volume, tin tức, social media, macro).
  2. **XAI methods** (ICFTS, DAVOTS) để giải thích kết quả.
  3. **Dashboard Flask** hiển thị prediction và các visual XAI.
  4. **Chatbot Gemini + Chatbot thứ hai** để người dùng tương tác, đặt câu hỏi về dữ liệu, mô hình và giải thích.
* **Output Bước 1**:
  1. Báo cáo ngắn (Markdown hoặc Word) mô tả:
     + Dữ liệu cần thu thập (ví dụ: Yahoo Finance, News API, X Platform, FRED).
     + Tần suất thu thập (giả sử “hàng ngày” tức 02:00 AM).
     + Tiêu chuẩn chất lượng (completeness, validity, consistency).
     + Các câu hỏi chính dự định giải đáp (ví dụ: “Yếu tố nào ảnh hưởng nhiều nhất đến giá TSLA?”, “Causal link giữa tariff exemptions và giá cổ phiếu?”).
  2. Sơ đồ luồng chung (Flowchart) gồm:
     + Nút “Data Sources” → “Preprocessing Pipeline” → “Database” → “Analysis & Visualization” → “Model Training & Prediction” → “Flask Dashboard + Chatbot”.

**1.2. Thu thập dữ liệu (Data Collection)**

* **Nguồn dữ liệu** (tối thiểu 3 nguồn):
  1. **Yahoo Finance** (qua thư viện yfinance) → lịch sử giá & volume.
  2. **News API** (ví dụ: NewsAPI.org) → tin tức (article title, description) phục vụ **text embedding**.
  3. **X Platform (Twitter API)** → tweet liên quan đến ticker (sentiment).
  4. (Option) **FRED API** → chỉ số kinh tế (lãi suất, CPI).
* **Lưu ý**:
  1. Mỗi nguồn gắn metadata (Thời gian thu thập, volume bản ghi, schema).
  2. Đặt script Python để tự động tải dữ liệu hàng ngày (theo cấu hình config.yml).

**Output**

* Folder data/raw/source\_X/ chứa file CSV:

data/

└─ raw/

├─ yahoo\_finance/

│ ├─ raw\_2025-06-01.csv

│ └─ raw\_2025-06-02.csv

├─ news\_api/

│ ├─ raw\_2025-06-01.json

│ └─ raw\_2025-06-02.json

├─ x\_platform/

│ ├─ raw\_2025-06-01.json

│ └─ raw\_2025-06-02.json

└─ fred/

├─ raw\_2025-06-01.csv

└─ raw\_2025-06-02.csv

* Tập con “metadata.txt” ghi sơ bộ schema & record count cho mỗi file.

**2. DATA DESCRIPTION & PREPROCESSING**

**2.1. Mục tiêu Bước 2**

* **Mô tả dữ liệu** trước và sau khi xử lý.
* **Tiền xử lý** multi-modal:
  1. **Price & Volume**: z-score normalization.
  2. **Text (News & Tweet)**: FinBERT embedding (768-dim).
  3. **Macro (FRED)**: Min-Max normalization.
  4. **Missing value**: KNN imputation (k = 5).
* **Output Bước 2**:
  1. Notebook lab2\_step2\_preprocessing.ipynb (mã Python cụ thể):
     + Thống kê missing & invalid.
     + Demo FinBERT embedding cho news & tweet.
     + Lưu file data/processed/multi\_modal\_2025-06-02.pkl ( có key: price\_df, embedding\_df, macro\_df ).
  2. Folder data/processed/daily/ chứa:
     + processed\_price\_2025-06-02.csv,
     + processed\_news\_emb\_2025-06-02.npy,
     + processed\_tweet\_emb\_2025-06-02.npy,
     + processed\_macro\_2025-06-02.csv.

**3. DATA ANALYSIS**

**3.1. Mục tiêu Bước 3**

* **Phân tích sơ bộ** multi-modal:
  1. **Correlation matrix** giữa các stock (Pearson).
  2. **Time-series decomposition** (trend, seasonal, residual) cho một asset mẫu (ví dụ TSLA).
  3. **Sentiment analysis**:
     + Tính tần suất classification (positive/neutral/negative) mỗi ngày.
     + Rolling average sentiment (7-day, 14-day) → correlation với returns.
  4. **Network Analysis**: tạo dynamic graph (Graph tại mỗi ngày) dựa trên correlation > 0.5.
* **Output Bước 3**:
  1. Notebook lab2\_step3\_analysis.ipynb:
     + Hiển thị correlation heatmap (stocks).
     + Biểu đồ decomposition TSLA (mẫu).
     + Bar chart sentiment distribution per day.
     + Vẽ 2 snapshot dynamic graph (ngày Volatility cao vs. Volatility thấp).

**4. DATA VISUALIZATION (ADVANCED + XAI)**

**4.1. Visualization Cơ Bản (≥ 5 biểu đồ)**

Trong Lab 1 dùng: Waffle, Area, Histogram, Bar, Pie, Scatter, Word Cloud, Choropleth. Lab 2 yêu cầu:

1. **Line Plot (Plotly interactive)**:
   * Price vs. Volume vs. Sentiment cùng chart (secondary y-axis).
2. **Area Plot + Waffle Chart**:
   * Tỷ lệ volume theo sector/industry (nếu Healthcare: ratio bệnh...).
3. **Histogram + Box Plot**:
   * Distribution of returns (positive vs. negative).
   * Box plot so sánh returns cho TSLA, AAPL, JPM.
4. **Scatter Plot**:
   * Price vs. Sentiment Rolling Avg, trendline + confidence interval.
5. **Word Cloud**:
   * Top-50 từ xuất hiện nhiều nhất trong news headlines, màu sắc tương ứng sentiment.
6. **Choropleth Map** (chỉ với Weather hoặc Healthcare có data geo):
   * Mức độ nhiệt độ hoặc chỉ số sức khỏe theo vùng/state.

**4.2. Visualization XAI (DAVOTS & ICFTS)**

**4.2.1. DAVOTS Heatmap**

* **Cách thực hiện**:
  1. Dùng **Captum** (PyTorch) hoặc tự viết hàm Integrated Gradients / SmoothGrad trên multi-modal model.
  2. Cho từng timestep t (ví dụ t = 𝑡₀−30 → 𝑡₀), tính attribution score per feature (Price, Volume, Sentiment, Macro).
  3. Kết quả tạo ma trận A ∈ [T × F], F = 4–5.
  4. Vẽ heatmap (Plotly) trục X = timestep, trục Y = features, color scale = normalized importance.
* **Output**:
  1. Notebook lab2\_step4\_xai\_visualization.ipynb lệnh tính attribution & lưu heatmap:

from captum.attr import IntegratedGradients

# … chuẩn bị input\_tensor (T × F), model, v.v.

ig = IntegratedGradients(model)

attributions = ig.attribute(input\_tensor, target=target\_asset) # shape (T,F)

# Vẽ heatmap

import plotly.express as px

fig = px.imshow(attributions.detach().cpu().numpy().T,

x=list(range(T)), y=feature\_names,

labels=dict(x="Time Step", y="Feature"),

color\_continuous\_scale="Viridis")

fig.update\_layout(title="DAVOTS Heatmap – Feature Importance Over Time")

fig.show()

* 1. Xuất file ảnh hoặc embed interactive HTML.
  2. **Giải thích**: Mô tả ý nghĩa color scale, ví dụ “tại timestep t = 7, Sentiment chiếm ~35% importance, Price ~22.6%, …”.

**4.2.2. ICFTS Causal Plot**

* **Cách thực hiện**:
  1. Sử dụng **pc** causal discovery (package causal-learn hoặc pgmpy).
  2. Chọn biến “x” (ví dụ Sentiment) và output “y” (Price).
  3. Thực hiện do-calculus: tạo bộ data counterfactual (Sentiment ± 5%, ±10%).
  4. Dự đoán ŷ\_do for each scenario (gọi model với input đã thay đổi).
  5. Tính Δy = ŷ\_do − ŷ\_baseline.
  6. Vẽ causal plot (Matplotlib hoặc Plotly) trục X = time steps, trục Y = Δy(t).
* **Output**:
  1. Trong notebook trên, lưu biểu đồ:

fig = px.line(x=list(range(len(delta\_y))), y=delta\_y,

labels=dict(x="Time Step", y="Δ Price (USD)"),

title="ICFTS Causal Plot – Sentiment → Price")

fig.add\_scatter(x=[7], y=[delta\_y[7]], mode="markers+text",

text=["Event: Tariff Exempt"], textposition="top center")

fig.show()

* 1. **Giải thích**: “Nếu Sentiment tăng 5% tại t = 7, Price dự kiến tăng +2.5 USD so với baseline, tương ứng với sự kiện exempt tariff ngày 11/04/2025.”

**4.3. Tổng hợp Output Visual**

1. **Dashboard Demo** hiển thị tất cả visual:
   * Price/Vol/Sentiment line chart (Plotly).
   * DAVOTS heatmap.
   * ICFTS causal plot.
   * Biểu đồ distribution (Histogram, Box).
   * Word Cloud.
2. **Folder lưu ảnh**:

visuals/

├─ line\_price\_sentiment.html

├─ davots\_heatmap.html

├─ icfts\_causal\_plot.html

├─ histogram\_returns.png

└─ wordcloud\_news.png

**5. CHATBOT INTEGRATION (GEMINI & CHATBOT THỨ HAI)**

**5.1. Mục tiêu Bước 5**

* Triển khai song song **Gemini Chatbot** và **Chatbot thứ hai** (Ví dụ IBM Watson Assistant hoặc ChatGPT API) để so sánh khả năng:
  1. **Trả lời** về quy trình tiền xử lý, mô tả dữ liệu (ví dụ “Triển khai FinBERT embedding thế nào?”).
  2. **Giải thích** visual XAI (ví dụ “Giải thích DAVOTS heatmap tại timestep 10?”).
  3. **Causal reasoning** (ví dụ “Sentiment tăng 5% → Price thay đổi ra sao?”).
* **Output Bước 5**:
  1. Code sample tích hợp API Gemini.
  2. Code sample tích hợp API Watson/Rasa/ChatGPT.
  3. Demo web (Flask) hiển thị 2 Chatbot, user có thể chọn “Gemini” hoặc “Alternate” (giao diện radio button).
  4. Bảng 10 câu hỏi mẫu (provided in report) và kết quả trả lời thu được.
  5. Nhận xét so sánh (accuracy, fluency, response time).

**5.2. Hướng dẫn tích hợp Gemini Chatbot**

**5.2.1. Chuẩn bị tài khoản & API Key**

* **Đăng ký Google Cloud Project** → Kích hoạt API “Gemini Chat” (nếu có).
* Tạo Service Account và tải file JSON credentials gemini\_credentials.json.

**5.2.2. Cài đặt thư viện (Python)**

pip install google-cloud-gemini

**5.2.3. Mã ví dụ gọi Gemini**

# file: chatbots/gemini\_client.py

import os

from google.cloud import gemini\_v1

def init\_gemini\_client(cred\_path="gemini\_credentials.json"):

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = cred\_path

client = gemini\_v1.chat.ChatServiceClient()

return client

def ask\_gemini(question, model="gemini-pro-1.0"):

client = init\_gemini\_client()

chat\_req = gemini\_v1.ChatCompletionRequest(

model=model,

prompts=[gemini\_v1.ChatPromptContent(text=question)]

)

response = client.chat\_completion(request=chat\_req)

return response.completion

**5.2.4. Ví dụ gọi trong Flask**

# trong app.py (Flask)

from flask import Flask, render\_template, request, jsonify

from chatbots.gemini\_client import ask\_gemini

app = Flask(\_\_name\_\_)

@app.route("/ask\_gemini", methods=["POST"])

def ask\_gemini\_route():

data = request.json

question = data.get("question", "")

answer = ask\_gemini(question)

return jsonify({"answer": answer})

# Phần HTML:

# <input type="text" id="gemini\_input" placeholder="Ask Gemini...">

# <button onclick="sendGemini()">Send</button>

# <div id="gemini\_output"></div>

# JS (embedded trong template):

// function sendGemini() {

// const q = document.getElementById("gemini\_input").value;

// fetch("/ask\_gemini", {

// method: "POST",

// headers: {'Content-Type': 'application/json'},

// body: JSON.stringify({question: q})

// })

// .then(res => res.json())

// .then(data => {

// document.getElementById("gemini\_output").innerText = data.answer;

// });

// }

**5.3. Hướng dẫn tích hợp Chatbot thứ hai (Ví dụ: IBM Watson Assistant)**

**5.3.1. Chuẩn bị Watson Assistant**

1. Đăng ký IBM Cloud → Tạo instance Watson Assistant → Lưu API Key & URL.
2. Xây dialog flow (intents, entities) tập trung trả lời:
   * Câu hỏi về dataset (ví dụ “Khi nào thì dùng KNN imputation?”)
   * Câu hỏi về XAI (ví dụ “Dòng nào trong DAVOTS heatmap thể hiện gì?”)
   * Câu hỏi technical (ví dụ “Làm sao để tính dynamic graph?”).
3. Tạo assistant\_credentials.json chứa { "apikey": "...", "url": "..." , "assistant\_id": "..." }.

**5.3.2. Cài đặt thư viện**

pip install ibm\_watson

**5.3.3. Mã ví dụ gọi Watson**

# file: chatbots/watson\_client.py

import json

from ibm\_watson import AssistantV2

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

def init\_watson():

creds = json.load(open("assistant\_credentials.json"))

auth = IAMAuthenticator(creds["apikey"])

assistant = AssistantV2(

version="2025-06-01",

authenticator=auth

)

assistant.set\_service\_url(creds["url"])

return assistant, creds["assistant\_id"]

def ask\_watson(question):

assistant, assistant\_id = init\_watson()

session = assistant.create\_session(assistant\_id=assistant\_id).get\_result()["session\_id"]

res = assistant.message(

assistant\_id=assistant\_id,

session\_id=session,

input={"message\_type":"text","text":question}

).get\_result()

# Lấy response text đầu tiên

if res["output"]["generic"]:

return res["output"]["generic"][0]["text"]

return "No response"

**5.3.4. Ví dụ gọi cùng Flask**

# Trong app.py tiếp tục:

from chatbots.watson\_client import ask\_watson

@app.route("/ask\_watson", methods=["POST"])

def ask\_watson\_route():

data = request.json

question = data.get("question", "")

answer = ask\_watson(question)

return jsonify({"answer": answer})

**5.4. Giao diện Chatbot Trong Flask Dashboard**

* **Template HTML (ví dụ templates/index.html)**:

html

CopyEdit

<!DOCTYPE html>

<html>

<head>

<title>Lab2 Dashboard with Chatbots</title>

<script src="https://cdn.plot.ly/plotly-2.0.0.min.js"></script>

</head>

<body>

<h1>Lab 2 – Dashboard</h1>

<!-- Đây sẽ là div cho các biểu đồ Plotly -->

<div id="price\_chart" style="width:600px;height:400px;"></div>

<div id="davots\_heatmap" style="width:600px;height:400px;"></div>

<div id="icfts\_plot" style="width:600px;height:400px;"></div>

<hr>

<h2>Chat With Models</h2>

<form id="chat\_form">

<label><input type="radio" name="bot" value="gemini" checked> Gemini</label>

<label><input type="radio" name="bot" value="watson"> Watson</label><br>

<input type="text" id="question\_input" size="80" placeholder="Type your question...">

<button type="button" onclick="sendQuestion()">Ask</button>

</form>

<div id="chat\_response" style="white-space: pre-wrap; border: 1px solid #ccc; padding:10px; margin-top:10px;"></div>

<script>

function sendQuestion() {

const question = document.getElementById("question\_input").value;

const bot = document.querySelector('input[name="bot"]:checked').value;

const endpoint = (bot === "gemini") ? "/ask\_gemini" : "/ask\_watson";

fetch(endpoint, {

method: "POST",

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({question: question})

})

.then(res => res.json())

.then(data => {

document.getElementById("chat\_response").innerText = data.answer;

});

}

// Hàm khởi tạo Plotly graphs (sẽ được load data via Flask route hoặc embed)

// Ví dụ:

document.addEventListener("DOMContentLoaded", function(){

// Load JSON data cho biểu đồ, giả định Flask trả về data qua /get\_visuals endpoint

fetch("/get\_visuals")

.then(res => res.json())

.then(data => {

Plotly.newPlot("price\_chart", data.price\_data.data, data.price\_data.layout);

Plotly.newPlot("davots\_heatmap", data.davots\_data.data, data.davots\_data.layout);

Plotly.newPlot("icfts\_plot", data.icfts\_data.data, data.icfts\_data.layout);

});

});

</script>

</body>

</html>

* **Flask Route cho “/get\_visuals”**:

# Trong app.py

import json

import pandas as pd

from flask import jsonify

@app.route("/get\_visuals")

def get\_visuals():

# 1) Price chart: đọc từ file JSON hoặc CSV đã chuẩn bị

price\_df = pd.read\_csv("data/processed/latest/latest\_source\_A.csv")

# Giả lập data cho Plotly:

price\_data = {

"data": [

{"x": price\_df["Date"].tolist(), "y": price\_df["ClosePrice"].tolist(),

"type": "scatter", "name": "ClosePrice"},

# Có thể thêm Volume & Sentiment...

],

"layout": {"title": "Price Time-Series (Latest)"}

}

# 2) DAVOTS heatmap (đã lưu sẵn JSON hoặc có thể load numpy rồi convert)

davots\_matrix = ... # load từ file npy hoặc JSON

davots\_data = {

"data": [

{"z": davots\_matrix, "x": feature\_names, "y": list(range(len(davots\_matrix))),

"type": "heatmap", "colorscale": "Viridis"}

],

"layout": {"title": "DAVOTS Heatmap (Latest)"}

}

# 3) ICFTS plot: tương tự

icfts\_data = {

"data": [

{"x": list(range(len(delta\_y))), "y": delta\_y, "type": "line", "name": "Δ Price"}

],

"layout": {"title": "ICFTS Causal Plot (Latest)"}

}

return jsonify({

"price\_data": price\_data,

"davots\_data": davots\_data,

"icfts\_data": icfts\_data

})

**6. MODEL BUILDING & PREDICTION**

**6.1. Mục tiêu Bước 6**

* **Triển khai model T-GNN++ Transformer**, kết hợp:
  1. **Dynamic graph pruning** (Pearson ≥ 0.5).
  2. **Multi-modal embedding** (1D CNN cho price, FinBERT cho text, MLP cho macro).
  3. **T-GNN module** (GAT + adaptive temporal attention).
  4. **Transformer module** (8 heads, 4 layers).
  5. **Fusion layer & Prediction** (linear).
  6. **Loss = MSE + 0.1·CausalPenalty** (ICFTS).
* **So sánh với baseline**:
  1. SARIMAX, Transformer (thuần), T-GNN (không Transformer), TFT (Temporal Fusion Transformer).
  2. XGBoost (features engineered).
* **Đánh giá**:
  1. MSE, Directional Accuracy, ECE, Inference Time.
  2. **Faithfulness Score (0.87)** & **Reliability Score (0.92)** (sample).
  3. Ablation study: w/o dynamic graph, w/o adaptive temporal attention, w/o causal penalty, w/o XAI modules.
  4. Per-asset performance (TSLA, AAPL, …).
  5. Performance across volatility: Low (Jan 2025) vs. High (Apr 2025).
* **Output Bước 6**:
  1. Notebook lab2\_step6\_model.ipynb (hoặc Python scripts):
     + Mã tạo dynamic graph, multi-modal embedding.
     + Định nghĩa T-GNN++ class (PyTorch).
     + Quá trình train/validate/test, lưu model best checkpoint.
     + Tính XAI metrics (Faithfulness, Reliability).
     + Lưu all results (CSV hoặc pickle).
  2. File results/ chứa:
     + performance\_summary.csv (MSE, accuracy, ECE mỗi model).
     + ablation\_results.csv.
     + per\_asset\_performance.csv.
  3. Tóm tắt đoạn code training convergence:
     + Lưu số epoch, MSE training/validation.

**7. APPLICATION DEVELOPMENT (FLASK DASHBOARD + CHATBOT)**

**7.1. Mục tiêu Bước 7**

* **Xây dựng Flask Dashboard**:
  1. **Hiển thị real-time visualization**: price, DAVOTS, ICFTS.
  2. **Cho phép user tương tác**:
     + Chọn asset (dropdown).
     + Chọn forecasting horizon (5 min, 1 hr, 1 day).
     + Chọn chế độ hiển thị XAI (DAVOTS / ICFTS).
  3. **Nhúng Chatbot**: Gemini & Chatbot thứ hai (Watson/ChatGPT).
  4. **Chức năng prediction on-demand**:
     + User click “Predict Next” → gọi model phục hồi (model checkpoint) → trả về prediction, overlay lên chart.
* **Output Bước 7**:
  1. Folder app/ gồm:
     + app.py (Flask main).
     + templates/index.html, templates/layout.html.
     + static/ chứa CSS, JS (Plotly.js).
     + chatbots/gemini\_client.py, chatbots/watson\_client.py.
     + model/ chứa checkpoint (ví dụ tgnnpp\_best.pth).
  2. **Dockerfile** (tùy chọn) để build Docker image cho app.
  3. Hướng dẫn chạy local (README.md):

# Cài đặt

pip install -r requirements.txt

export FLASK\_APP=app.py

flask run # mặc định lên http://127.0.0.1:5000

* 1. (Option) **Heroku/Streamlit Cloud** link demo (nếu có thể deploy).

**II. SƯỜN VIẾT BÁO CÁO LATEX**

Dưới đây là **sườn LaTeX** (template) cho báo cáo Lab 2, đặt tên file DAP391m\_Lab2\_Report\_GroupX.tex.

\documentclass[10pt, conference]{IEEEtran}

\usepackage{amsmath,amsfonts,amssymb}

\usepackage{graphicx}

\usepackage{hyperref}

\usepackage{booktabs}

\usepackage{multirow}

\usepackage{caption}

\usepackage{subcaption}

\title{%

Lab 2 – Xây dựng Pipeline, XAI, Dashboard Flask và So sánh Chatbot

}

\author{%

\IEEEauthorblockN{Nhóm XYZ\\

Trường Đại học ABC, Khoa CNTT}

\IEEEauthorblockA{Email: \{student1, student2, student3\}@abc.edu.vn}

}

\begin{document}

\maketitle

\begin{abstract}

Intraday financial forecasting faces challenges về tính minh bạch và tính giải thích theo MiFID II.

Lab 2 xây dựng pipeline tự động thực hiện thu thập, tiền xử lý, phân tích dữ liệu từ nhiều nguồn

(Yahoo Finance, News API, X Platform, FRED), tích hợp XAI (ICFTS, DAVOTS), phát triển Dashboard

Flask hiển thị real-time visualization và nhúng Chatbot Gemini cùng Chatbot thay thế để so sánh

khả năng trả lời. Kết quả: pipeline chạy ổn định 5 ngày (MSE ~0.030, directional accuracy ~82%),

XAI metrics đạt Faithfulness = 0.87, Reliability = 0.92, Chatbot Gemini cho độ chính xác ~85%

cho các câu hỏi dữ liệu, thời gian phản hồi ~0.6 s.

\end{abstract}

\IEEEpeerreviewmaketitle

\section{Introduction}

\label{sec:introduction}

\begin{itemize}

\item Bối cảnh tổng quan (Niên vụ HFT, yêu cầu MiFID II, explainability).

\item Giới thiệu Lab 2: kế thừa Lab 1 dựa trên 7 bước, bổ sung pipeline tự động, XAI, Dashboard Flask, Chatbot.

\item Đóng góp chính:

\begin{enumerate}

\item Pipeline tự động tải – validate – chuẩn hóa dữ liệu hàng ngày.

\item Tích hợp XAI (ICFTS, DAVOTS) để giải thích model.

\item Dashboard Flask real-time hiển thị XAI visualization.

\item So sánh Chatbot Gemini vs. Watson (hoặc ChatGPT).

\end{enumerate}

\end{itemize}

\section{Methodology}

\label{sec:methodology}

\subsection{Overview of the 7-Step Process}

\begin{itemize}

\item Mô tả ngắn 7 bước:

1. Business Understanding \& Data Collection,

2. Data Description \& Preprocessing,

3. Data Analysis,

4. Data Visualization (Advanced + XAI),

5. Chatbot Integration,

6. Model Building \& Prediction,

7. Application Development (Flask Dashboard + Chatbot).

\item Sơ đồ tổng quát pipeline (Figure \ref{fig:high\_level\_pipeline}).

\end{itemize}

\subsection{Step 1: Business Understanding \& Data Collection}

\begin{itemize}

\item Mục tiêu & câu hỏi kinh doanh (Business Questions).

\item Danh sách “multi-modal” data:

\begin{enumerate}

\item Yahoo Finance (Price \& Volume),

\item News API (News Articles),

\item X Platform (Tweets),

\item FRED (Macro Indicators).

\end{enumerate}

\item Tần suất tải (cronjob hàng ngày 02:00 AM).

\item Dữ liệu đầu ra: folder \texttt{data/raw/\*}, metadata.

\end{itemize}

\subsection{Step 2: Data Description \& Preprocessing}

\begin{itemize}

\item Thống kê sơ bộ (missing, invalid, schema).

\item Tiền xử lý:

\begin{enumerate}

\item Price \& Volume: z-score normalization, KNN imputation (k=5).

\item Text (News, Tweet): FinBERT embedding (768-dim).

\item Macro: Min-Max normalization.

\end{enumerate}

\item Kết quả: multi-modal data sau xử lý, lưu \texttt{data/processed/daily/}.

\item Code minh hoạ (Python/PyTorch): mất ít nhất 50 dòng.

\end{itemize}

\subsection{Step 3: Data Analysis}

\begin{itemize}

\item Correlation matrix giữa stocks (Pearson r).

\item Time-series decomposition (Seasonal, Trend, Residual) cho TSLA (Figure \ref{fig:tsla\_decomp}).

\item Sentiment statistics: distribution positive/neutral/negative, rolling average.

\item Dynamic graph: sử dụng NetworkX, threshold $r \ge 0.5$, vẽ 2 snapshot (low vs. high volatility) (Figure \ref{fig:dynamic\_graph}).

\end{itemize}

\subsection{Step 4: Data Visualization (Advanced) \& XAI}

\label{subsec:visualization\_xai}

\begin{itemize}

\item \textbf{Advanced Visualization:}

\begin{enumerate}

\item Line Plot (Price vs. Volume vs. Sentiment).

\item Area Plot + Waffle Chart (Volume distribution).

\item Histogram \& Box Plot (returns).

\item Scatter Plot (Price vs. Sentiment, kèm trendline).

\item Word Cloud (News headlines).

\item [Nếu có geo] Choropleth map (Weather/Healthcare).

\end{enumerate}

\item \textbf{XAI Visualization:}

\begin{enumerate}

\item DAVOTS Heatmap:

\begin{itemize}

\item Mô tả: dùng Integrated Gradients (Captum) để tính attribution, tạo ma trận $A\_{T\times F}$.

\item Vẽ heatmap bằng Plotly (Figure \ref{fig:davots\_heatmap}).

\item Giải thích: “tại $t=7$, Sentiment importance = 35\%…”.

\end{itemize}

\item ICFTS Causal Plot:

\begin{itemize}

\item Mô tả thuật toán: PC algorithm + do-calculus (Pearl 2009 [15]).

\item Tạo data counterfactual (Sentiment ± 5\%, ± 10\%).

\item Tính Δ$y = \hat y\_{do(x)} - \hat y\_{baseline}$, vẽ plot (Figure \ref{fig:icfts\_plot}).

\item Giải thích: causal link Tariff Exempt → 2.5\% Price spike (11/04/2025).

\end{itemize}

\item Counterfactual Scenario Visualization (Giá thực tế vs. Giá nếu không có negative news) (Figure \ref{fig:counterfactual}).

\end{enumerate}

\item Tất cả code đều có chú thích rõ ràng, lưu file HTML (Plotly) để embed vào Flask.

\end{itemize}

\subsection{Step 5: Chatbot Integration}

\label{subsec:chatbot\_integration}

\begin{itemize}

\item \textbf{Gemini Chatbot:}

\begin{enumerate}

\item Chuẩn bị: tạo Google Cloud Project, JSON credentials.

\item Code gọi: `gemini\_client.py` (Google Cloud Gemini API).

\item Test câu hỏi mẫu (n=10), ví dụ:

\begin{itemize}

\item “Tôi muốn biết KNN imputation hoạt động ra sao?”

\item “DAVOTS heatmap interpret như thế nào?”

\item “ICFTS causal shows gì ngày 11/04/2025?”

\end{itemize}

\end{enumerate}

\item \textbf{Alternate Chatbot (Watson Assistant hoặc ChatGPT API):}

\begin{enumerate}

\item Chuẩn bị: Watson Assistant instance (credentials JSON) hoặc ChatGPT API key.

\item Code gọi: `watson\_client.py` (hoặc `chatgpt\_client.py`).

\item Test sử dụng cùng 10 câu hỏi mẫu để so sánh:

\begin{itemize}

\item Đo “accuracy” (đầy đủ thông tin?), “fluency”, “response time (ms)”.

\end{itemize}

\end{enumerate}

\item \textbf{Kết quả so sánh} (bảng trong report):

| Question | Gemini Answer | AltBot Answer | Accuracy (Gemini) | Accuracy (Alt) | Response Time (Gemini) | Response Time (Alt) |

|---|---|---|---|---|---|---|

| … | … | … | \% | \% | ms | ms |

\item \textbf{Giao diện web:}

\begin{itemize}

\item Radio button chọn “Gemini” hoặc “Alternate”.

\item Text input + “Send” → fetch API to Flask → hiển thị response.

\end{itemize}

\item \textbf{Giải thích cách nhúng Chatbot} vào Flask (xem Section \ref{subsec:chatbot\_html}).

\end{itemize}

\subsection{Step 6: Model Building \& Prediction}

\label{subsec:modeling}

\begin{itemize}

\item \*\*Kiến trúc T-GNN++ Transformer (tham khảo T-GNN++ paper)\*\*

\begin{enumerate}

\item Dynamic graph: nodes = {AAPL,…}, edges = Pearson r ≥ 0.5.

\item Embedding layer:

\begin{itemize}

\item 1D CNN (64 filters, kernel = 3) cho Price.

\item FinBERT embedding (768-dim) cho text.

\item MLP (128→64, ReLU) cho Macro.

\end{itemize}

\item T-GNN module:

\begin{itemize}

\item Graph Attention Networks (GAT):

\(\alpha\_{ij} = \mathrm{softmax}(\mathrm{LeakyReLU}(a^T[W h\_i \parallel W h\_j]))\).

\item Adaptive temporal attention:

\(\alpha\_t = \mathrm{softmax}(\frac{\mathrm{Volatility}(t) \cdot W\_q h\_t}{\sqrt{d\_k}})\).

\end{itemize}

\item Transformer module: 8 heads, 4 layers, positional encoding.

\item Fusion & Prediction:

\(\displaystyle h\_{final} = W\_f [h\_{\text{T-GNN}} \parallel h\_{\text{Transformer}}] + b\_f\).

Prediction: \( \hat y\_t = W\_o h\_{final} + b\_o .\)

\item Loss: \( \mathrm{MSE}(y, \hat y) + 0.1 \times \mathrm{CausalPenalty} .\)

\(\displaystyle \mathrm{CausalPenalty} = \sum\_{i,j} \Bigl| P(y\_i \mid \mathrm{do}(x\_j)) - \hat P(y\_i \mid x\_j)\Bigr|\).

\end{enumerate}

\item \*\*Baseline Models for Comparison\*\*:

\begin{itemize}

\item SARIMAX (statsmodels).

\item Transformer (thuần, không GNN).

\item T-GNN (không Transformer).

\item TFT (Temporal Fusion Transformer).

\item XGBoost (features = MA, vol, sentiment…).

\end{itemize}

\item \*\*Thiết lập Experiments\*\*:

\begin{itemize}

\item Môi trường: PyTorch 2.0.1 trên NVIDIA RTX 3060 (12 GB).

\item Tham số: epochs = 50, batch\_size = 32, optimizer = AdamW(lr=1e-4, wd=1e-2), early stopping = patience 3.

\item Data split: Training 70\% (70k rows), Validation 15\% (15k), Test 15\% (15k), chú trọng Q1–Q2 2025.

\item Metrics: MSE, Directional Accuracy (\% đúng chiều), ECE, Faithfulness Score (Pearson corr), Reliability Score (std dev over 50 sample variants), Inference Time (giây/batch).

\end{itemize}

\item \*\*Ablation Study\*\*:

\begin{enumerate}

\item w/o dynamic graph pruning.

\item w/o adaptive temporal attention.

\item w/o causal penalty.

\item w/o XAI modules (ICFTS + DAVOTS).

\end{enumerate}

\item \*\*Per-Asset Performance\*\*: TSLA, AAPL, JPM, UNH, v.v.

\item \*\*Market Condition Robustness\*\*: Low Vol (Jan 2025) vs. High Vol (Apr 2025).

\item \*\*Output\*\*: `lab2\_step6\_model.ipynb`, `results/\*.csv`, `models/\*.pth`.

\end{itemize}

\subsection{Step 7: Application Development (Flask Dashboard + Chatbot)}

\label{subsec:flask\_dashboard}

### 7.1. Mục tiêu

- \*\*Xây Flask App\*\* hiển thị interactive visualization, cho user tương tác, và nhúng Chatbot.

- \*\*Prediction on-demand\*\*: user bấm “Predict Next” → gọi model → trả kết quả → overlay chart.

### 7.2. Cấu trúc thư mục App

app/  
├─ app.py  
├─ requirements.txt  
├─ Dockerfile (tùy chọn)  
├─ chatbots/  
│ ├─ gemini\_client.py  
│ └─ watson\_client.py  
├─ model/  
│ └─ tgnnpp\_best.pth  
├─ templates/  
│ ├─ layout.html  
│ └─ index.html  
└─ static/  
└─ main.js (có thể tách riêng JS nếu cần)

### 7.3. app.py (Flask)

```python

# app/app.py

from flask import Flask, render\_template, request, jsonify

import pandas as pd

import plotly

import plotly.graph\_objs as go

from chatbots.gemini\_client import ask\_gemini

from chatbots.watson\_client import ask\_watson

import torch

from model.tgnnpp import TGNNTPlusPlus # Giả sử model class đã định nghĩa

app = Flask(\_\_name\_\_)

# 1) Route chính hiển thị dashboard

@app.route("/")

def index():

return render\_template("index.html")

# 2) Route /get\_visuals: trả JSON chứa Plotly charts cho price, davots, icfts

@app.route("/get\_visuals")

def get\_visuals():

# 2.1) Price chart (Load latest data)

price\_df = pd.read\_csv("data/processed/latest/latest\_source\_A.csv")

price\_fig = go.Figure()

price\_fig.add\_trace(go.Scatter(

x=price\_df["Date"], y=price\_df["ClosePrice"], mode="lines", name="Price"

))

price\_fig.update\_layout(title="Latest Price Time-Series", xaxis\_title="Date", yaxis\_title="Price (USD)")

# 2.2) DAVOTS heatmap: giả sử đã lưu matrix vào npy hoặc JSON

import numpy as np

davots\_matrix = np.load("data/xai/davots\_matrix\_latest.npy") # shape (F × T)

feature\_names = ["Price", "Volume", "Sentiment", "Macro"]

davots\_fig = go.Figure(data=go.Heatmap(

z=davots\_matrix,

x=list(range(davots\_matrix.shape[1])),

y=feature\_names,

colorscale="Viridis"

))

davots\_fig.update\_layout(title="DAVOTS Heatmap (Latest)")

# 2.3) ICFTS causal plot: giả sử đã lưu delta\_y vào npy

delta\_y = np.load("data/xai/icfts\_delta\_y\_latest.npy") # shape (T,)

icfts\_fig = go.Figure()

icfts\_fig.add\_trace(go.Scatter(

x=list(range(len(delta\_y))), y=delta\_y, mode="lines", name="Δ Price"

))

icfts\_fig.update\_layout(title="ICFTS Causal Plot (Latest)", xaxis\_title="Time Step", yaxis\_title="Δ Price (USD)")

# Encode Plotly figures sang JSON

price\_json = json.dumps(price\_fig, cls=plotly.utils.PlotlyJSONEncoder)

davots\_json = json.dumps(davots\_fig, cls=plotly.utils.PlotlyJSONEncoder)

icfts\_json = json.dumps(icfts\_fig, cls=plotly.utils.PlotlyJSONEncoder)

return jsonify({

"price\_data": price\_json,

"davots\_data": davots\_json,

"icfts\_data": icfts\_json

})

# 3) Route /ask\_gemini

@app.route("/ask\_gemini", methods=["POST"])

def ask\_gemini\_route():

data = request.json

question = data.get("question", "")

answer = ask\_gemini(question)

return jsonify({"answer": answer})

# 4) Route /ask\_watson

@app.route("/ask\_watson", methods=["POST"])

def ask\_watson\_route():

data = request.json

question = data.get("question", "")

answer = ask\_watson(question)

return jsonify({"answer": answer})

# 5) Route /predict: Prediction on-demand cho asset và horizon

@app.route("/predict", methods=["POST"])

def predict\_route():

data = request.json

asset = data.get("asset", "TSLA")

horizon = data.get("horizon", "5min")

# 5.1) Load model

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

model = TGNNTPlusPlus().to(device)

model.load\_state\_dict(torch.load("model/tgnnpp\_best.pth", map\_location=device))

model.eval()

# 5.2) Chuẩn bị input (lấy latest window data cho asset)

# - Giả sử hàm get\_latest\_window trả về tensor shape (1, T, F)

input\_tensor = get\_latest\_window(asset, horizon).to(device)

with torch.no\_grad():

pred = model(input\_tensor) # shape (1, 1) nếu dự báo 1 bước

# 5.3) Convert pred sang float

pred\_value = float(pred.cpu().numpy()[0][0])

return jsonify({"prediction": pred\_value})

# Hàm hỗ trợ get\_latest\_window (ví dụ)

def get\_latest\_window(asset, horizon):

# Load data từ data/processed/latest/latest\_source\_A.csv

# Lọc asset tương ứng, normalize, convert thành tensor

# Dummy code:

import torch

return torch.randn(1, 30, 5) # 30 timesteps, 5 features

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**7.4. templates/index.html**

html

CopyEdit

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Lab 2 – Dashboard & Chatbots</title>

<script src="https://cdn.plot.ly/plotly-2.0.0.min.js"></script>

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

.chart { margin-bottom: 40px; }

#chat\_response { white-space: pre-wrap; border: 1px solid #ccc; padding: 10px; margin: 10px 0; }

</style>

</head>

<body>

<h1>Lab 2 – Real-Time Dashboard & Chatbots</h1>

<div class="chart" id="price\_chart" style="width:600px;height:400px;"></div>

<div class="chart" id="davots\_heatmap" style="width:600px;height:400px;"></div>

<div class="chart" id="icfts\_plot" style="width:600px;height:400px;"></div>

<hr>

<h2>Ask a Question</h2>

<form id="chat\_form">

<label><input type="radio" name="bot" value="gemini" checked> Gemini</label>

<label><input type="radio" name="bot" value="watson"> Watson</label><br>

<input type="text" id="question\_input" size="80" placeholder="Type your question...">

<button type="button" onclick="sendQuestion()">Ask</button>

</form>

<div id="chat\_response"></div>

<hr>

<h2>Predict Next Step</h2>

<label for="asset\_select">Asset:</label>

<select id="asset\_select">

<option value="TSLA">TSLA</option>

<option value="AAPL">AAPL</option>

<option value="JPM">JPM</option>

</select>

<label for="horizon\_select">Horizon:</label>

<select id="horizon\_select">

<option value="5min">5 Minutes</option>

<option value="1hr">1 Hour</option>

<option value="1day">1 Day</option>

</select>

<button onclick="predictNext()">Predict Next</button>

<div id="pred\_output" style="margin-top:10px;"></div>

<script>

// Lấy và render các biểu đồ Plotly

document.addEventListener("DOMContentLoaded", function() {

fetch("/get\_visuals")

.then(res => res.json())

.then(data => {

const price\_fig = JSON.parse(data.price\_data);

Plotly.newPlot("price\_chart", price\_fig.data, price\_fig.layout);

const davots\_fig = JSON.parse(data.davots\_data);

Plotly.newPlot("davots\_heatmap", davots\_fig.data, davots\_fig.layout);

const icfts\_fig = JSON.parse(data.icfts\_data);

Plotly.newPlot("icfts\_plot", icfts\_fig.data, icfts\_fig.layout);

});

});

function sendQuestion() {

const question = document.getElementById("question\_input").value;

const bot = document.querySelector('input[name="bot"]:checked').value;

const endpoint = (bot === "gemini") ? "/ask\_gemini" : "/ask\_watson";

fetch(endpoint, {

method: "POST",

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({question: question})

})

.then(res => res.json())

.then(data => {

document.getElementById("chat\_response").innerText = data.answer;

});

}

function predictNext() {

const asset = document.getElementById("asset\_select").value;

const horizon = document.getElementById("horizon\_select").value;

fetch("/predict", {

method: "POST",

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({asset: asset, horizon: horizon})

})

.then(res => res.json())

.then(data => {

document.getElementById("pred\_output").innerText =

`Prediction for ${asset} (${horizon}): ${data.prediction.toFixed(4)} USD`;

});

}

</script>

</body>

</html>

**III. CẤU TRÚC BÁO CÁO LATEX CHI TIẾT**

Dưới đây là phần chi tiết hơn cho từng section/subsection trong file lab2\_report.tex.

\documentclass[10pt, conference]{IEEEtran}

\usepackage{amsmath,amsfonts,amssymb}

\usepackage{graphicx}

\usepackage{hyperref}

\usepackage{booktabs}

\usepackage{multirow}

\usepackage{caption}

\usepackage{subcaption}

\usepackage{array}

\usepackage{algorithm}

\usepackage{algorithmic}

\title{%

Lab 2 – Xây dựng Pipeline Hàng Ngày, Tích hợp XAI, Flask Dashboard và So sánh Chatbot

}

\author{%

\IEEEauthorblockN{Nhóm XYZ\\

Trường Đại học ABC, Khoa CNTT}

\IEEEauthorblockA{Email: \{student1, student2, student3\}@abc.edu.vn}

}

\begin{document}

\maketitle

\begin{abstract}

Intraday financial forecasting đòi hỏi sự minh bạch và giải thích cho quyết định giao dịch theo quy định MiFID II.

Trong Lab 2, chúng tôi xây dựng pipeline tự động để thu thập, validate, và chuẩn hóa dữ liệu hàng ngày từ nhiều nguồn (Yahoo Finance, News API, X Platform, FRED), tích hợp giải thích XAI (ICFTS, DAVOTS), phát triển một ứng dụng Flask Dashboard hiển thị real-time visualization, và nhúng hai Chatbot (Gemini và Watson) để so sánh khả năng trả lời câu hỏi. Kết quả: pipeline chạy liên tục 5 ngày thành công, model T-GNN++ đạt MSE = 0.030, directional accuracy = 82\%, Faithfulness = 0.87, Reliability = 0.92. Chatbot Gemini có độ chính xác ~85\% câu hỏi, thời gian phản hồi ~0.6 s, vượt Watson về độ đầy đủ thông tin.

\end{abstract}

\begin{IEEEkeywords}

Financial Forecasting, XAI, T-GNN++, DAVOTS, ICFTS, Flask Dashboard, Chatbot Gemini, Data Pipeline

\end{IEEEkeywords}

\section{Introduction}

\label{sec:introduction}

\IEEEPARstart{I}{ntraday} financial forecasting poses significant challenges due to market volatility, multi-modal data complexity, and the need for regulatory-compliant interpretability (MiFID II) \cite{MiFID}. While Lab 1 laid nền móng về thu thập và xử lý dữ liệu, Lab 2 sẽ triển khai một pipeline tự động, tích hợp Explainable AI (XAI) và phát triển ứng dụng Flask để cung cấp visualizations real-time. Đồng thời, chúng tôi nhúng hai Chatbot—Gemini và Watson—để người dùng hỏi đáp về dữ liệu, mô hình và XAI, rồi so sánh kết quả.

\subsection{Contributions}

\begin{itemize}

\item \*\*Pipeline tự động\*\*: thu thập, validate, và chuẩn hóa dữ liệu hàng ngày với cronjob.

\item \*\*Explainable AI (XAI)\*\*: ICFTS (Iterative Counterfactual Temporal Search) và DAVOTS (Dense-pixel Attribution for Visualizing Over Time Series) \cite{TGNN++}.

\item \*\*Flask Dashboard\*\* hiển thị real-time multi-modal visualization (Plotly) và XAI outputs.

\item \*\*Chatbot Integration\*\*: Gemini Chatbot (Google Cloud) và Watson Assistant so sánh khả năng trả lời 10 câu hỏi mẫu.

\item \*\*Model Performance\*\*: T-GNN++ Transformer đạt MSE = 0.030, directional accuracy = 82\%, Faithfulness = 0.87, Reliability = 0.92.

\end{itemize}

\subsection{Organization}

Phần còn lại của báo cáo được tổ chức như sau:

\begin{itemize}

\item Section \ref{sec:methodology}: tổng quan quy trình 7 bước và chi tiết từng bước.

\item Section \ref{sec:experiments}: setup experiments, baseline, benchmarking, XAI metrics, so sánh Chatbot.

\item Section \ref{sec:discussion}: thảo luận, limitations, future work.

\item Section \ref{sec:conclusion}: kết luận.

\end{itemize}

\section{Methodology}

\label{sec:methodology}

\subsection{Overview of the 7-Step Process}

\label{subsec:overview\_7\_steps}

\begin{figure}[!t]

\centering

\includegraphics[width=0.48\textwidth]{figures/high\_level\_pipeline.png}

\caption{High-Level Pipeline: (1) Data Sources, (2) Preprocessing Pipeline, (3) Database, (4) Analysis \& Visualization, (5) Model Training \& Prediction, (6) Flask Dashboard \& Chatbots.}

\label{fig:high\_level\_pipeline}

\end{figure}

Quy trình được chia thành 7 bước liên tiếp (xem Hình \ref{fig:high\_level\_pipeline}):

\begin{enumerate}

\item \*\*Business Understanding \& Data Collection\*\*: Xác định mục tiêu, thu thập dữ liệu multi-modal.

\item \*\*Data Description \& Preprocessing\*\*: Khám phá dữ liệu sơ bộ, tiền xử lý (normalization, imputation, embedding).

\item \*\*Data Analysis\*\*: Phân tích correlation, decomposition, sentiment, network.

\item \*\*Data Visualization (Advanced + XAI)\*\*: Interactive line plots, histogram, box, word cloud, DAVOTS heatmap, ICFTS causal plot.

\item \*\*Chatbot Integration\*\*: Nhúng Gemini Chatbot và Chatbot thứ hai (Watson) để hỏi đáp.

\item \*\*Model Building \& Prediction\*\*: Xây T-GNN++ Transformer, baseline, evaluate.

\item \*\*Application Development\*\*: Flask Dashboard real-time + Chatbots + Prediction on-demand.

\end{enumerate}

\subsection{Step 1: Business Understanding \& Data Collection}

\label{subsec:step1}

\subsubsection{Business Goals \& Key Questions}

Lab 2 tập trung vào intraday forecasting (giá 5-phút) cho 10 cổ phiếu US (AAPL, MSFT, TSLA, AMZN, GOOGL, NVDA, JPM, UNH, V, WMT). Key questions:

\begin{itemize}

\item Yếu tố nào (Price, Volume, News Sentiment, Social Media Sentiment, Macro Indicators) ảnh hưởng nhiều nhất đến giá?

\item Causal relationship giữa News Sentiment và Price trong các sự kiện quan trọng (ví dụ 11/04/2025).

\item Hiệu quả so với Baseline (SARIMAX, Transformer, T-GNN thuần).

\item Hỗ trợ quyết định theo MiFID II (giải thích rõ ràng, minh bạch).

\end{itemize}

\subsubsection{Data Sources}

\begin{table}[!ht]

\caption{Các nguồn dữ liệu chính}

\label{tab:data\_sources}

\centering

\begin{tabular}{>{\raggedright}p{1.6cm} p{1.2cm} p{1.6cm} p{3.0cm}}

\toprule

\textbf{Nguồn} & \textbf{Kiểu} & \textbf{Volume} & \textbf{Thời gian} \\

\midrule

Yahoo Finance & Price \& Volume & 100,000 rows & 2018–2025 \\

News API & News Articles & 10,000 articles & 2018–2025 \\

X Platform (Twitter) & Social Media Posts & 5,000 tweets & 2018–2025 \\

FRED & Macro Indicators & Daily records & 2018–2025 \\

\bottomrule

\end{tabular}

\end{table}

- \*\*Yahoo Finance\*\*: Sử dụng `yfinance` để tải lịch sử giá (Open, High, Low, Close, Adjusted Close, Volume) với interval=1d hoặc 5min (nếu API hỗ trợ).

- \*\*News API\*\*: Tải JSON articles (title, description, content, date) về các công ty trong danh sách.

- \*\*X Platform (Twitter)\*\*: Dùng Twitter API (v2) để crawl tweets có chứa ticker (ví dụ “\$TSLA”). Phân loại sentiment (positive/neutral/negative).

- \*\*FRED\*\*: Dữ liệu macro (lãi suất, CPI, GDP) từ FRED API (JSON → CSV).

\subsubsection{Automated Data Collection Pipeline}

- Viết 4 script Python:

1. `download\_yahoo.py`:

```python

import yfinance as yf

import pandas as pd

from datetime import datetime, timedelta

def download\_yahoo(date):

tickers = ["AAPL","MSFT","TSLA","AMZN","GOOGL","NVDA","JPM","UNH","V","WMT"]

# Nếu interval=1d, set start=end=date; nếu 5min, set start=date 00:00, end=date 23:59

df = yf.download(tickers,

start=date, end=date, interval="1d", progress=False)

df = df["Adj Close"].reset\_index()

df.to\_csv(f"data/raw/yahoo\_finance/raw\_{date}.csv", index=False)

if \_\_name\_\_=="\_\_main\_\_":

from sys import argv

if len(argv)>1:

dt = argv[1]

else:

dt = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")

download\_yahoo(dt)

```

2. `download\_news.py`:

```python

import requests, json

from datetime import datetime, timedelta

API\_KEY = "YOUR\_NEWSAPI\_KEY"

def download\_news(date):

url = f"https://newsapi.org/v2/everything?q=(AAPL OR MSFT OR TSLA)…&from={date}&to={date}&pageSize=100&apiKey={API\_KEY}"

r = requests.get(url)

data = r.json()

with open(f"data/raw/news\_api/raw\_{date}.json","w") as f:

json.dump(data, f)

if \_\_name\_\_=="\_\_main\_\_":

from sys import argv

if len(argv)>1:

dt = argv[1]

else:

dt = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")

download\_news(dt)

```

3. `download\_twitter.py`:

```python

import tweepy

import json

from datetime import datetime, timedelta

BEARER\_TOKEN = "YOUR\_TWITTER\_BEARER\_TOKEN"

client = tweepy.Client(bearer\_token=BEARER\_TOKEN)

def download\_twitter(date):

query = "($AAPL OR $TSLA OR $AMZN) lang:en -is:retweet"

start = date + "T00:00:00Z"

end = date + "T23:59:59Z"

tweets = client.search\_all\_tweets(query=query, start\_time=start, end\_time=end, max\_results=500)

with open(f"data/raw/x\_platform/raw\_{date}.json","w") as f:

json.dump(tweets.data, f)

if \_\_name\_\_=="\_\_main\_\_":

from sys import argv

if len(argv)>1:

dt = argv[1]

else:

dt = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")

download\_twitter(dt)

```

4. `download\_fred.py`:

```python

import requests, pandas as pd

from datetime import datetime, timedelta

API\_KEY = "YOUR\_FRED\_API\_KEY"

def download\_fred(date):

series = ["FEDFUNDS","GDP"]

records = []

for s in series:

url = f"https://api.stlouisfed.org/fred/series/observations?series\_id={s}&observation\_start={date}&observation\_end={date}&api\_key={API\_KEY}&file\_type=json"

r = requests.get(url).json()

for obs in r["observations"]:

records.append({"Date":obs["date"], "Series":s, "Value":obs["value"]})

df = pd.DataFrame(records)

df.to\_csv(f"data/raw/fred/raw\_{date}.csv", index=False)

if \_\_name\_\_=="\_\_main\_\_":

from sys import argv

if len(argv)>1:

dt = argv[1]

else:

dt = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")

download\_fred(dt)

```