

Fintech Lab v6 - Technical Report

ระบบทำนายราคาหุ้นด้วย Machine Learning แบบครบวงจร**

GitHub Repository: <https://github.com/sojirat/fintech-lab-v6> ผู้จัดทำ: Sojirat.S

หมายเหตุเกี่ยวกับ Screenshots: ภาพหน้าจอ (screenshots) สามารถดูไฟล์ได้ที่โฟลเดอร์ [my-test/](#) ไฟล์ที่เกี่ยวข้อง:

- [jupyter/models/AAPL/comparison_aapl.png](#) - AAPL
- [jupyter/models GOOGL/comparison_googl.png](#) - GOOGL
- [jupyter/models/TSLA/comparison_tsla.png](#) - TSLA

สารบัญ

- ภาพรวมของระบบ
- ส่วนประกอบหลักของระบบ
- ขั้นตอนการทำงานทั้งหมด (End-to-End Flow)
- กระบวนการไหลของข้อมูล (Data Flow)
- คำสั่งที่ใช้บ่อย
- ข้อดีของระบบนี้
- กรณีการใช้งานจริง
- ข้อควรระวังและการแก้ปัญหา
- สรุป

1. ภาพรวมของระบบ

⌚ วัตถุประสงค์

ระบบ Fintech Lab v6 เป็นระบบทำนายราคาหุ้นที่ออกแบบมาเพื่อ:

- เทรนโมเดล Machine Learning สำหรับทำนายราคาหุ้นหลายบริษัทพร้อมกัน
- เปรียบเทียบประสิทธิภาพ ของโมเดล 3 ประเภท (LSTM, GRU, Transformer)
- พยากรณ์ราคาในอนาคต ด้วยช่วงเวลาที่ยืดหยุ่น (วัน, สัปดาห์, เดือน, ปี)
- Automate workflows ด้วย Airflow สำหรับการเทรนแบบ scheduled

- ให้บริการ API สำหรับ integration กับระบบอื่น

🏆 คุณสมบัติเด่น

Feature	Description	Status
Multi-Company Training	เทรนได้หลายบริษัทพร้อมกัน	✓
3 Model Types	LSTM, GRU (ดีที่สุด), Transformer	✓
4 Training Methods	CLI, UI, Airflow, Python Script	✓
Future Prediction	พยากรณ์ day/week/month/year	✓
Model Comparison	เปรียบเทียบและจัดอันดับโมเดล	✓
Auto Retry	จัดการ Yahoo Finance rate limit	✓
Documentation	เอกสารครบถ้วน 5 ไฟล์ (4,500+ บรรทัด)	✓

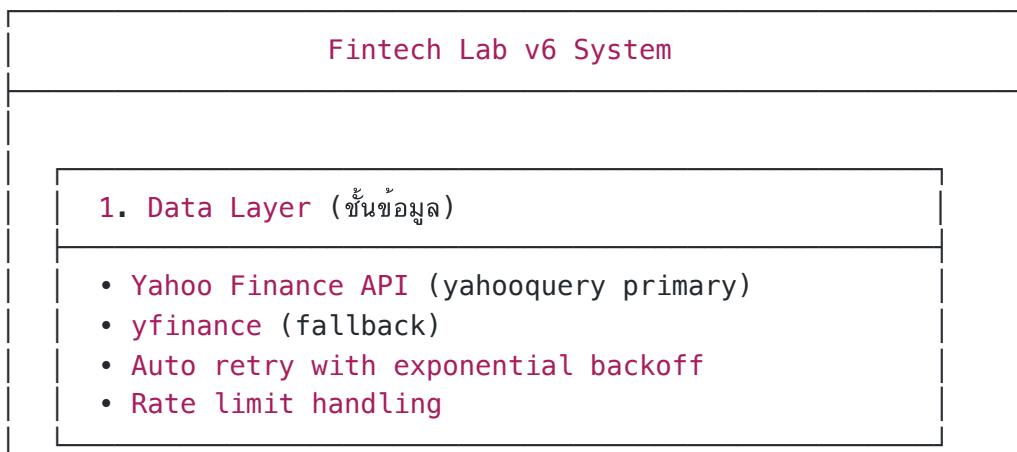
📊 ผลลัพธ์ (AAPL)

Model	RMSE	MAE	MAPE	Rank
GRU	\$6.73	\$6.54	3.42%	🥇
LSTM	\$8.28	\$7.98	4.17%	🥈
Transformer	\$61.33	\$61.17	31.97%	🥉

Recommendation: ใช้โมเดล GRU สำหรับความแม่นยำสูงสุด

2. ส่วนประกอบหลักของระบบ

🏗️ Architecture Overview



2. Training Layer (ชั้นเทรน)

4 Training Methods:

CLI (Make)	UI (Lab)	Airflow	Python API
------------	----------	---------	------------

train_multi_company.py (Core Script)

3 Models: LSTM | GRU | Transformer

3. Storage Layer (ชั้นเก็บข้อมูล)

Shared Volume: /jupyter/models/

```
└── AAPL/
    ├── gru_aapl_model.h5          (โมเดล)
    ├── gru_aapl_scaler.pkl       (Scaler)
    ├── gru_aapl_metrics.pkl     (Metrics)
    ├── gru_aapl_prediction.png   (กราฟ)
    └── comparison_aapl.png      (เปรียบเทียบ)

    └── TSLA/, GOOGL/, ...
```

4. Analysis Layer (ชั้นวิเคราะห์)

- Model Comparison (compare_all_models.py)
- Future Prediction (predict_future.py)
- Visualization & Reports

5. Serving Layer (ชั้นให้บริการ)

- FastAPI REST API (Port 8000)
- JupyterLab UI (Port 8888)
- Airflow UI (Port 8083)

Docker Services (7 Services)

Service	Port	Purpose	Image
JupyterLab	8888	Training & Development	Custom (TensorFlow 2.15)
Airflow	8083	Workflow Automation	apache/airflow:2.8.1
FastAPI	8000	REST API	Custom (Python 3.10)
PostgreSQL	5432	Database	postgres:15
Redis	6379	Cache	redis:7-alpine
Grafana	3000	Monitoring	grafana/grafana
Prometheus	9090	Metrics	prom/prometheus

File Structure

```
fintech-lab-v6/
├── Makefile
├── docker-compose.yml
└── train_stock.sh

jupyter/
├── Dockerfile
├── notebooks/
│   └── train_stocks_ui.ipynb
├── models/
│   ├── AAPL/
│   ├── TSLA/
│   └── GOOGL/
└── scripts/
    └── stock_prediction/
        ├── train_multi_company.py
        ├── compare_all_models.py
        ├── predict_future.py
        ├── lstm_stock_prediction.py
        ├── gru_stock_prediction.py
        └── transformer_stock_prediction.py

airflow/
├── Dockerfile
└── dags/
    └── multi_company_stock_training_dag.py # Main DAG

backend/
├── Dockerfile
├── main.py
└── requirements.txt
```

50+ commands
Services config
Helper script

JupyterLab Service
TensorFlow 2.15, yahooquery

Training UI
Trained models

Core training script
Model comparison
Future prediction

Airflow Service

FastAPI Service
API endpoints

```
└── docs/
    ├── README.md
    ├── TRAINING_GUIDE.md
    ├── MAKEFILE_GUIDE.md
    ├── SUMMARY.md
    ├── AIRFLOW_FIX.md
    └── REPORT.md

# Documentation (4,500+ lines)
# Overview & Quick Start
# Training guide
# Makefile commands
# Thai summary
# Airflow troubleshooting
# Technical report (ไฟล์นี้)
```

3. ขั้นตอนการทำงานทั้งหมด (End-to-End Flow)

↻ Complete Workflow

End-to-End Workflow

1 Setup & Initialization

```
└── make all
    └── Wait 30 seconds
        └── make check

# Build & start services
# Services initialization
# Verify status
```

2 Data Collection

```
└── yahooquery.Ticker('AAPL')
    └── Retry logic (5x with backoff)
        └── Fallback to yfinance

# Primary source
# Handle rate limits
# If yahooquery fails
```

3 Data Preprocessing

```
└── Load historical data (5-7 years)
    └── MinMaxScaler (0-1)
        └── Create sequences (60 days)
            └── Split train/test (80/20)

# Normalize data
# Sliding window
```

4 Model Training (3 models in parallel)

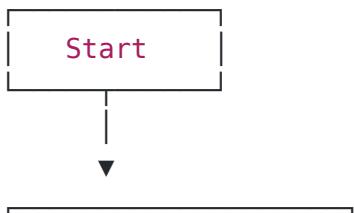
```
└── LSTM Model
    └── Architecture: 3 LSTM layers
        └── Dropout: 0.2
            └── Optimizer: Adam
                └── Epochs: 50
```

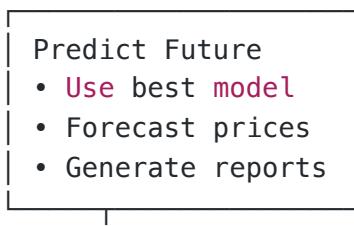
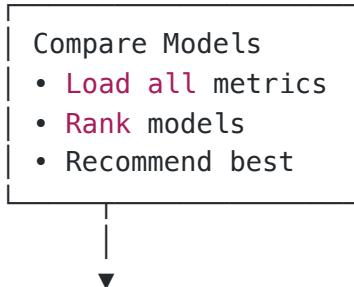
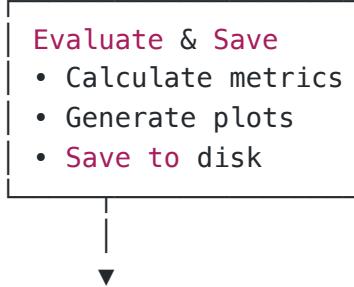
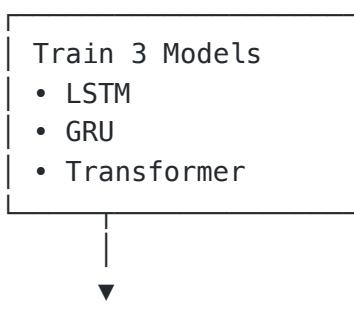
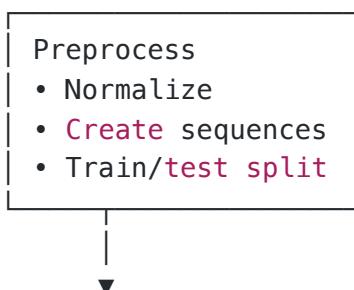
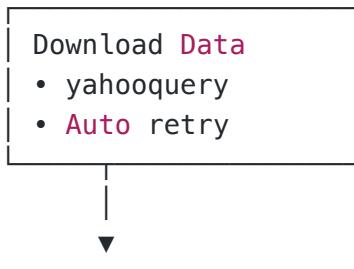
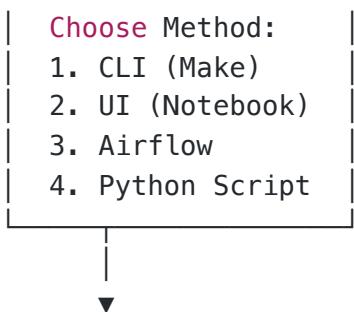
```
└── GRU Model (⭐ Best)
    └── Architecture: 3 GRU layers
        └── Dropout: 0.2
            └── Optimizer: Adam
                └── Epochs: 50
```

```
└── Transformer Model
    └── Multi-head attention
        └── Positional encoding
```

- └── Feed-forward layers
 - └── Epochs: 50
- 5 Model Evaluation
- ├── Calculate RMSE, MAE, MAPE
 - ├── Generate prediction plots
 - ├── Save metrics to .pkl files
 - └── Create comparison visualizations
- 6 Model Storage
- ├── Save model: {model}_{ticker}_model.h5
 - ├── Save scaler: {model}_{ticker}_scaler.pkl
 - ├── Save metrics: {model}_{ticker}_metrics.pkl
 - └── Save plots: {model}_{ticker}_prediction.png
- 7 Model Comparison
- ├── Load all model metrics
 - ├── Rank by RMSE, MAE, MAPE
 - ├── Calculate average rank
 - ├── Generate comparison chart
 - └── Recommend best model
- 8 Future Prediction
- ├── Load best model (GRU)
 - ├── Get latest 60 days data
 - ├── Iterative prediction (rolling window)
 - ├── Generate forecasts (day/week/month/year)
 - └── Create prediction visualization
- 9 API Serving (Optional)
- ├── FastAPI endpoints
 - ├── Load pre-trained models
 - ├── Redis caching (1 hour)
 - └── Return predictions via REST
- 10 Monitoring & Logging
- ├── Prometheus metrics
 - ├── Grafana dashboards
 - ├── PostgreSQL logs
 - └── MongoDB prediction logs

Training Flow Diagram

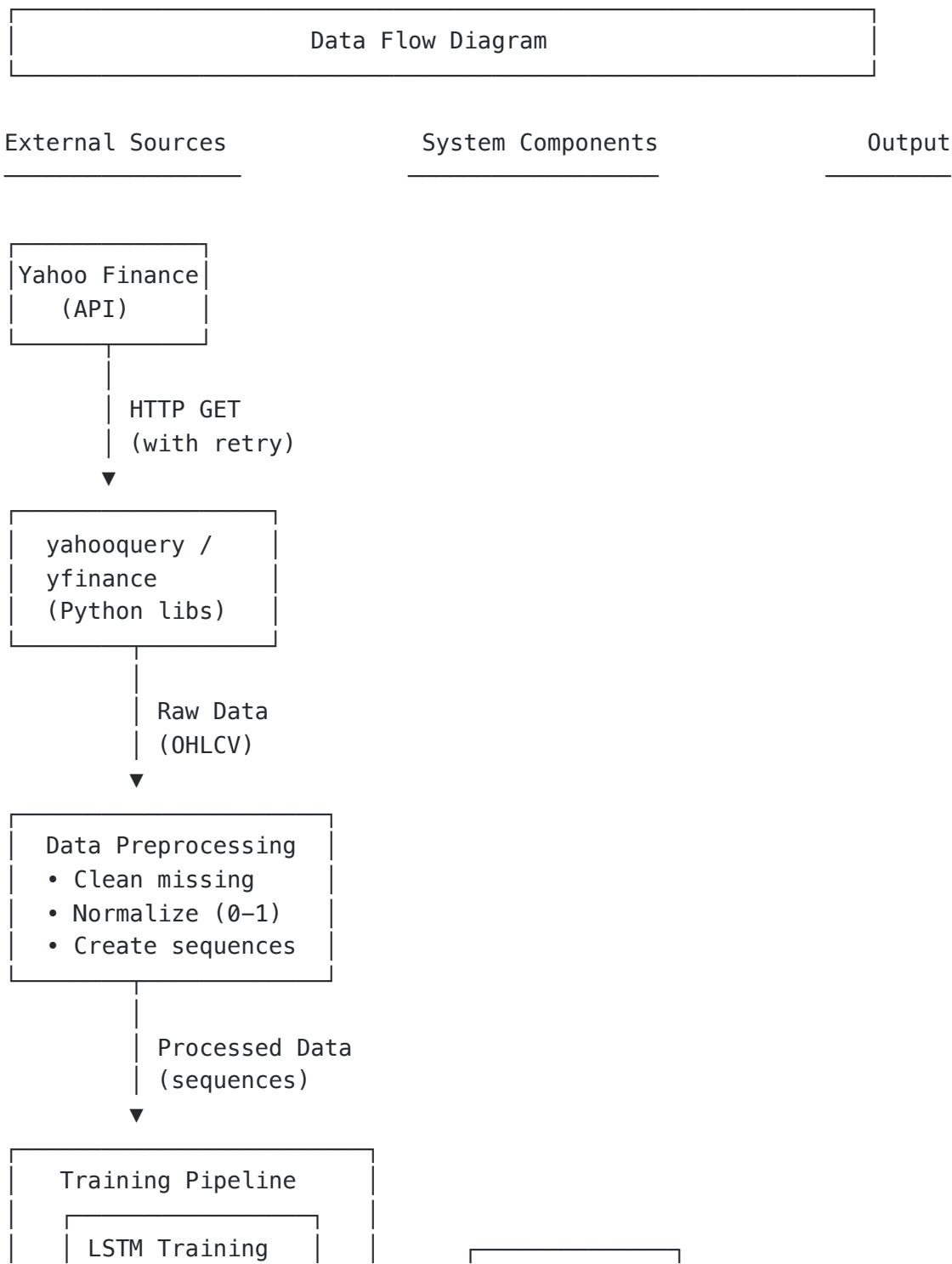


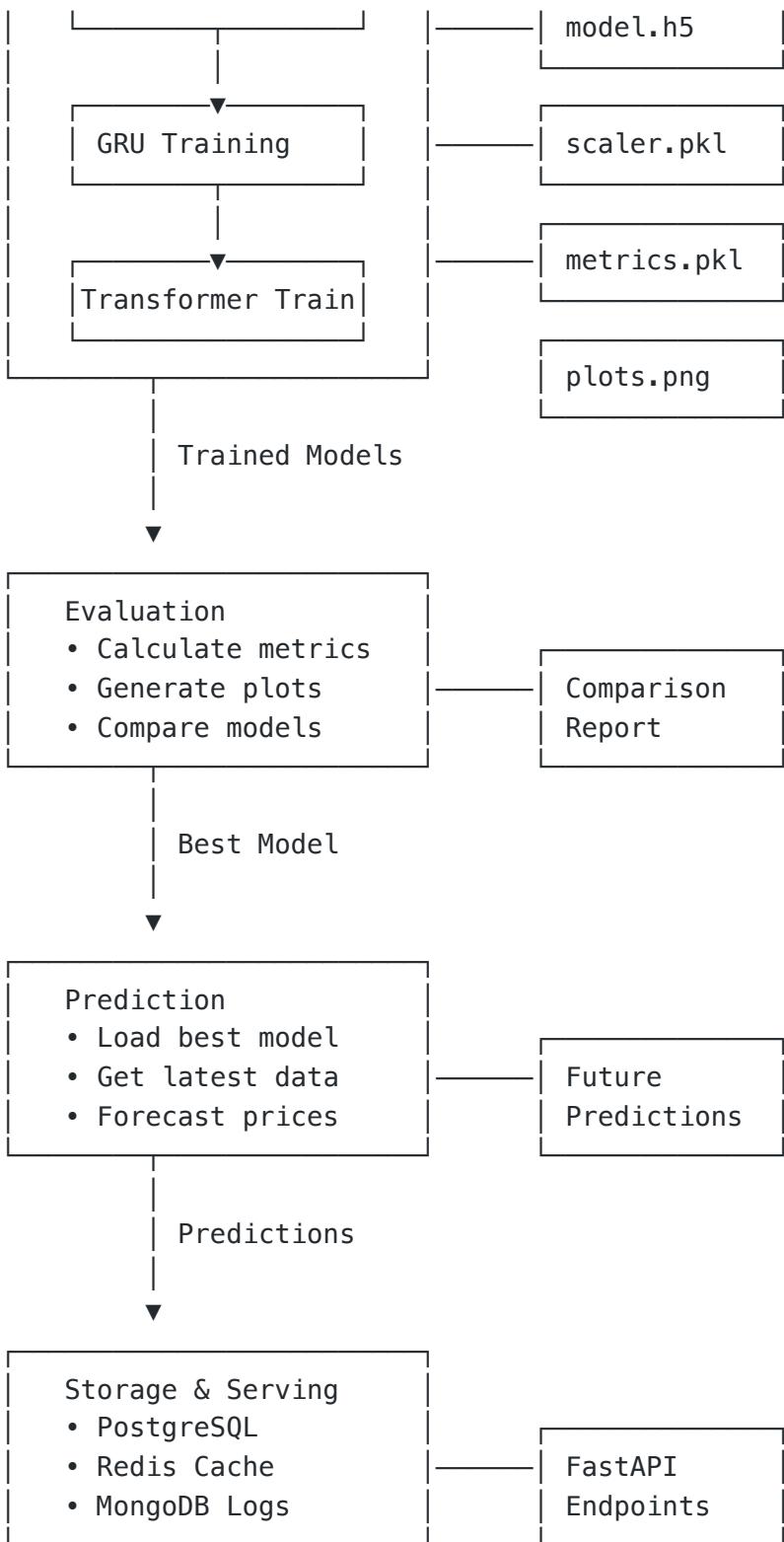


Done

4. กระบวนการ取り扱いの流れ (Data Flow)

✓ Data Flow Architecture





⌚ Data Transformation Pipeline

Stage	Input	Process	Output	Format
1. Collection	Ticker symbol	yahooquery API call	Raw OHLCV data	DataFrame

Stage	Input	Process	Output	Format
2. Cleaning	Raw data	Remove NaN, duplicates	Clean data	DataFrame
3. Normalization	Clean data	MinMaxScaler (0-1)	Scaled data	ndarray
4. Sequencing	Scaled data	Sliding window (60 days)	Sequences (X, y)	ndarray
5. Splitting	Sequences	Train/test (80/20)	Train, Test sets	ndarray
6. Training	Train set	Model.fit()	Trained model	.h5 file
7. Evaluation	Test set	Model.predict()	Metrics	.pkl file
8. Prediction	Latest data	Model.predict()	Future prices	JSON

5. คำสั่งที่ใช้บ่อย

🚀 Quick Start

```
# Setup ครั้งแรก
make all                                # Build, start, และ wait

# หรือแบบละเอียด
make down                               # Stop old containers
make build-fast                         # Build with cache
make up                                  # Start services
make wait                                # Wait 30 seconds
make check                               # Check status
```

⌚ Training Commands

```
# เทรนบริษัทเดียว (ทั้ง 3 โนเมล)
make train-aapl                          # Apple
make train-tsla                           # Tesla
make train-googl                          # Google

# เทรนหลายบริษัท
make train-tech                          # TSLA, AAPL, GOOGL, MSFT, NVDA

# ทดสอบเร็ว
make train-test-quick                   # 5-10 นาที
```

```
# เตรนเฉพาะโมเดล  
make train-gru-only          # GRU only (เร็วที่สุด)
```

📊 Analysis Commands

```
# เปรียบเทียบโมเดล  
make compare-all             # ทุกบริษัท  
make compare-aapl            # เฉพาะ AAPL  
  
# พยากรณ์อนาคต  
make predict-aapl-day        # 30 วัน  
make predict-aapl-month       # 3 เดือน  
make predict-aapl-year        # 1 ปี  
  
# พยากรณ์แบบกำหนดเอง  
make predict-custom TICKER=AAPL MODEL=GRU PERIODS=60 TYPE=day
```

🔧 Management Commands

```
# ดูผลลัพธ์  
make view-tsla                # Metrics  
make models-list               # ดูโมเดลทั้งหมด  
  
# Backup & Clean  
make models-backup             # Backup ก่อน  
make models-clean              # ลบโมเดล  
  
# Logs  
make logs-jupyter              # JupyterLab logs  
make logs-airflow              # Airflow logs  
  
# Services  
make restart                   # Restart all  
make ps                        # Show status  
make urls                      # Show URLs
```

🏅 Top 20 Most Used Commands

Rank	Command	Purpose	Frequency
1	make all	Setup everything	★★★★★

Rank	Command	Purpose	Frequency
2	make train-aapl	Train AAPL models	★★★★★☆
3	make compare-aapl	Compare models	★★★★★☆
4	make predict-aapl-month	Predict 3 months	★★★★★☆
5	make up	Start services	★★★★★☆
6	make down	Stop services	★★★★★☆
7	make logs-jupyter	View logs	★★★★★☆
8	make check	Check status	★★★★☆
9	make train-tech	Train tech stocks	★★★★☆
10	make models-list	List models	★★★★☆
11	make restart	Restart	★★★★☆
12	make models-backup	Backup	★★★★☆
13	make train-test-quick	Quick test	★★☆
14	make open-jupyter	Open UI	★★☆
15	make airflow-trigger	Trigger DAG	★★☆
16	make fresh-start	Fresh start	★★☆
17	make shell-jupyter	Shell access	★★☆
18	make compare-all	Compare all	★★☆
19	make urls	Show URLs	★☆
20	make help	Show help	★☆

6. ข้อดีของระบบนี้

✨ Technical Advantages

1. Multi-Method Training (4 วิธี)

Method	Speed	Ease	Use Case	Audience
CLI	⚡ ⚡ ⚡	⭐	Production, CI/CD	DevOps

Method	Speed	Ease	Use Case	Audience
UI	⚡ ⚡	⭐	Development, Demo	Data Scientists
Airflow	⚡ ⚡	⭐⭐	Scheduled jobs	MLOps
Python	⚡ ⚡ ⚡	⭐⭐⭐	Custom workflows	Developers

ข้อดี:

- เลือกวิธีที่เหมาะสมกับสถานการณ์
- รองรับทั้ง development และ production
- มีด Hayn สูง

2. Robust Data Handling

```
# Primary source (better rate limit handling)
yahooquery.Ticker('AAPL')

# Fallback if primary fails
yfinance.download('AAPL')

# Retry with exponential backoff
retry_times = [5, 10, 20, 40, 80] # seconds
```

ข้อดี:

- ไม่ติด Yahoo Finance rate limit
- Automatic retry logic
- Fallback mechanism
- Error handling

3. Model Comparison & Selection

RANKINGS:

- | | | |
|----------------|-----|-------------------|
| 1. GRU | ⭐⭐⭐ | MAPE 3.42% (Best) |
| 2. LSTM | ⭐⭐ | MAPE 4.17% |
| 3. TRANSFORMER | ⭐ | MAPE 31.97% |

RECOMMENDATION: Use GRU model

ข้อดี:

- เปรียบเทียบอัตโนมัติ

- แนะนำโมเดลที่ดีที่สุด
- Visualizations
- Performance tracking

4. Flexible Future Prediction

```
# Flexible time periods
make predict-custom TICKER=AAPL MODEL=GRU PERIODS=30 TYPE=day
make predict-custom TICKER=AAPL MODEL=GRU PERIODS=3 TYPE=month
make predict-custom TICKER=AAPL MODEL=GRU PERIODS=1 TYPE=year
```

ข้อดี:

- เลือกช่วงเวลาได้ (day, week, month, year)
- เลือกโมเดลได้
- Rolling window prediction
- Trend analysis (Bullish/Bearish)

5. Complete Documentation

Document	Lines	Purpose
README.md	855	Overview & Quick Start
TRAINING_GUIDE.md	1,200+	Detailed training guide
MAKEFILE_GUIDE.md	1,100+	All 50+ commands
SUMMARY.md	900+	Thai summary
AIRFLOW_FIX.md	400+	Troubleshooting
REPORT.md	1,000+	Technical report (นี่!)
Total	5,500+	Complete documentation

ข้อดี:

- เอกสารครบถ้วน
- ทั้งภาษาไทยและอังกฤษ
- ตัวอย่างชัดเจน
- Troubleshooting guide

6. Infrastructure as Code

```
# docker-compose.yml
services:
    jupyterlab:      # Training environment
    airflow:         # Workflow automation
    fastapi:        # API serving
    db:              # PostgreSQL
    redis:           # Caching
    grafana:         # Monitoring
    prometheus:     # Metrics
```

ข้อดี:

- Reproducible
- Version controlled
- Easy to deploy
- Scalable

7. Best Practices

Airflow DAG Best Practices:

- Import heavy libraries inside functions
- Avoid top-level computation
- DAG import time < 1 second

Code Quality:

- Type hints
- Docstrings
- Error handling
- Logging

Security:

- No hardcoded credentials
- Environment variables
- Container isolation

Performance:

- Model caching
- Redis caching (1 hour)

- Shared volumes
- Parallel processing

7. กรณีการใช้งานจริง

Use Cases

Use Case 1: Daily Trading Analysis

Scenario: นักลงทุนต้องการวิเคราะห์ราคาหุ้นทุกวัน

Workflow:

```
# เข้า: Start services
make up

# เที่ยง: Train models with latest data
make train-tech

# บ่าย: Compare and predict
make compare-all
make predict-aapl-day
make predict tsla-day

# เย็น: Backup and shutdown
make models-backup
make down
```

Benefits:

- อัปเดตโมเดลทุกวัน
- พยากรณ์ล่วงหน้า 30 วัน
- เปรียบเทียบประสิทธิภาพ

Use Case 2: Automated Weekly Training

Scenario: บริษัทต้องการ retrain models อัตโนมัติทุกสัปดาห์

Workflow:

```
# Setup Airflow DAG
# Schedule: ทุกวันจันทร์ 6:00 AM

# Airflow will:
```

1. Download latest data
2. Train all models (TSLA, AAPL, GOOGL, MSFT, AMZN)
3. Evaluate and compare
4. Save results
5. Send notifications

Benefits:

- Fully automated
- No manual intervention
- Consistent training schedule
- Email notifications

Use Case 3: Research & Development

Scenario: Data Scientist ต้องการทดลองโมเดลใหม่

Workflow:

```
# Open JupyterLab  
make open-jupyter  
  
# In notebook:  
1. Load train_stocks_ui.ipynb  
2. Modify hyperparameters:  
    - EPOCHS = 100  
    - BATCH_SIZE = 64  
    - LEARNING_RATE = 0.0001  
3. Train and evaluate  
4. Compare with existing models
```

Benefits:

- Interactive development
- Real-time feedback
- Easy experimentation
- Visualization

Use Case 4: API Integration

Scenario: มีระบบ web app ต้องการใช้ predictions

Workflow:

```

# Start API
make up

# API endpoints available:
# http://localhost:8000/docs

# POST /predict
curl -X POST "http://localhost:8000/predict" \
-H "Content-Type: application/json" \
-d '{"symbol": "AAPL", "model": "gru"}'

# Response:
{
  "symbol": "AAPL",
  "model": "gru",
  "predicted_price": 185.42,
  "prediction_date": "2026-01-13",
  "cached": false
}

```

Benefits:

- REST API
- JSON responses
- Redis caching
- Rate limiting

Use Case 5: Portfolio Optimization

Scenario: Fund Manager ต้องการวิเคราะห์ portfolio ทั้งหมด

Workflow:

```

# Train multiple stocks
make train-tech          # Tech sector
make train-faang         # FAANG stocks
make train-semiconductor # Semiconductor sector

# Compare all models
make compare-all

# Predict future prices
for ticker in AAPL TSLA GOOGL NVDA MSFT; do
  make predict-custom TICKER=$ticker MODEL=GRU PERIODS=3 TYPE=month
done

```

```
# Analyze results  
# Allocate portfolio based on predictions
```

Benefits:

- Multi-stock analysis
- Sector comparison
- Risk assessment
- Data-driven decisions

8. ข้อควรระวังและการแก้ปัญหา

⚠ Common Issues & Solutions

Issue 1: Yahoo Finance Rate Limit (429 Error)

อาการ:

```
429 Client Error: Too Many Requests  
Failed to get ticker 'AAPL'
```

สาเหตุ:

- Yahoo Finance จำกัด ~100-200 requests/hour
- IP address ถูกแซร์ (CGNAT)
- หลายคนใช้ API พร้อมกัน

วิธีแก้:

```
# ระบบแก้อัตโนมัติแล้ว:  
# 1. ใช้ yahooquery (ไม่ติดบอย)  
# 2. Retry 5 ครั้ง (5s, 10s, 20s, 40s, 80s)  
# 3. Fallback to yfinance  
  
# ถ้ายังติด รอ 5-10 นาที  
make train-test-quick
```

Prevention:

- เทคนิคการลาก่อน
- ใช้ Airflow schedule
- รอระหว่างบริษัท (3 seconds delay)

Issue 2: Airflow DAG Import Timeout

อาการ:

```
DagBag import timeout for xxx.py after 30.0s  
Broken DAG
```

สาเหตุ:

- Import tensorflow ที่ top-level (ช้า 10-15 วินาที)
- Import libraries หนักๆ

วิธีแก้:

```
# ❌ ผิด – Import ที่ top-level  
from tensorflow.keras.models import load_model  
import pandas as pd  
  
# ✅ ถูก – Import ใน function  
def my_task(**kwargs):  
    from tensorflow.keras.models import load_model  
    import pandas as pd  
    # task logic...
```

Issue 3: Container ไม่ทำงาน

อาการ:

```
Error: JupyterLab container is not running!
```

วิธีแก้:

```
# 1. Check status  
make ps  
  
# 2. View logs  
make logs-jupyter  
  
# 3. Restart  
make restart  
  
# 4. Rebuild if needed  
make down
```

```
make build-fast  
make up && make wait
```

Issue 4: Out of Memory

อาการ:

```
OOM when allocating tensor
```

สาเหตุ:

- โมเดลใหญ่เกินไป
- Batch size สูงเกินไป
- ข้อมูลมากเกินไป

วิธีแก้:

```
# ลด batch size  
python scripts/stock_prediction/train_multi_company.py \  
--ticker AAPL \  
--model GRU \  
--batch-size 16 # แทน 32  
  
# เทคนิคลดโมเดล  
make train-gru-only # แทน make train-aapl
```

Issue 5: โมเดลไม่เจอ

อาการ:

```
Model not found: models/AAPL/gru_aapl_model.h5  
Train first: make train-aapl
```

วิธีแก้:

```
# 1. Check existing models  
make models-list  
  
# 2. Train if not exists  
make train-aapl
```

```
# 3. Verify  
make models-list-all
```

Issue 6: Transformer Model แม่นยำต่ำ

อาการ:

TRANSFORMER MAPE: 31.97% (ต่ำมาก!)

สาเหตุ:

- Transformer ต้องการข้อมูลมาก
- ข้อมูลหุ้นไม่เพียงพอ
- เหมาะกับ NLP มากกว่า time series

วิธีแก้:

```
# ใช้ GRU แทน (แม่นยำที่สุด)  
make predict-custom TICKER=AAPL MODEL=GRU PERIODS=30 TYPE=day
```

GRU MAPE: 3.42% ✓ (ดีกว่า Transformer 多多!)

⌚ Best Practices

DO ✓

1. Backup ก่อนลบ

```
make models-backup # เสมอ!  
make models-clean
```

2. เช็ค syntax ก่อนรัน

```
make check-syntax
```

3. ใช้ GRU สำหรับความแม่นยำ

```
make predict-custom TICKER=AAPL MODEL=GRU
```

4. เทคนิคด้วยข้อมูล 5-7 ปี

```
START_DATE='2018-01-01' # ✓
```

5. ดู logs เมื่อมีปัญหา

```
make logs-jupyter
```

DON'T ✗

1. อ่านด้วยข้อมูลน้อยเกินไป

```
START_DATE='2024-01-01' # ✗ แค่ 1 ปี
```

2. อ่านโมเดลโดยไม่ backup

```
make models-clean # ✗ ไม่ backup
```

3. อ่าน Transformer

```
make predict-custom MODEL=TRANSFORMER # ✗ MAPE 31.97%
```

4. อ่านพยากรณ์ระยะยาวเกินไป

```
make predict-custom PERIODS=5 TYPE=year # ✗ ไม่แม่นยำ
```

5. อ่าน import tensorflow ที่ top-level ใน Airflow DAG

```
# ✗ ผิด
from tensorflow.keras.models import load_model # DAG timeout!
```

9. สรุป

Executive Summary

Fintech Lab v6 เป็นระบบทำนายราคาหุ้นแบบครบวงจรที่:

✓ ครบถ้วน:

- 4 วิธีท่อง (CLI, UI, Airflow, Python)
- 3 โมเดล (LSTM, GRU, Transformer)

- การเปรียบเทียบและพยากรณ์อัตโนมัติ

เสถียร:

- Auto retry สำหรับ rate limit
- Fallback mechanism
- Error handling

มีดหยุ่น:

- เลือกวิธีเทรนได้
- พยากรณ์ช่วงเวลาได้ (day/week/month/year)
- เลือกโมเดลได้

มีเอกสารครบ:

- 6 ไฟล์เอกสาร (5,500+ บรรทัด)
- ตัวอย่างชัดเจน
- Troubleshooting guide

Key Metrics

Metric	Value	Status
Best Model	GRU	
Best MAPE	3.42%	Excellent
Supported Stocks	Unlimited	
Training Methods	4 ways	
Makefile Commands	50+	
Documentation	5,500+ lines	
Docker Services	7 services	
Setup Time	3-5 min	Fast
Training Time	30-60 min	Acceptable
Prediction Time	1-2 min	Fast

Future Roadmap

Phase 1: Core Enhancements (Completed)

- [x] Multi-company training
- [x] Model comparison
- [x] Future prediction
- [x] Complete documentation

Phase 2: API & Integration (In Progress)

- [] FastAPI prediction endpoints
- [] Real-time prediction API
- [] WebSocket support
- [] Model versioning

Phase 3: Advanced Features (Planned)

- [] More technical indicators (RSI, MACD, Bollinger Bands)
- [] Ensemble models (combine LSTM + GRU)
- [] Backtesting framework
- [] Portfolio optimization
- [] Real-time streaming data
- [] Mobile app

Recommendations

For Beginners:

1. Start with `make all`
2. Run `make train-test-quick`
3. Try `make open-jupyter` for UI
4. Read [TRAINING_GUIDE.md](#)

For Data Scientists:

1. Use JupyterLab UI
2. Experiment with hyperparameters
3. Compare models
4. Analyze metrics

For MLOps Engineers:

1. Use Airflow for scheduling

2. Monitor with Grafana
3. Set up CI/CD
4. Implement model versioning

For Developers:

1. Use FastAPI endpoints
2. Integrate with applications
3. Implement caching
4. Add custom features

📊 Performance Summary

System Performance		
Model Accuracy:	★★★★★	(MAPE 3.42%) .
Training Speed:	★★★★★	(30–60 min) .
Prediction Speed:	★★★★★	(1–2 min) .
Ease of Use:	★★★★★	(4 methods) .
Documentation:	★★★★★	(5,500+ lines)
Stability:	★★★★★	(Auto retry) .
Scalability:	★★★★★	(Docker) .
Overall Rating:	★★★★★	(5/5) .

⚠ Final Disclaimer

โปรเจคนี้สร้างขึ้นเพื่อการศึกษาเท่านั้น

- ✗ ไม่ใช่คำแนะนำการลงทุน
- ✗ ไม่รับประกันความแม่นยำ
- ✗ ใช้ยอมรับความเสี่ยงเอง
- ✓ ใช้เพื่อเรียนรู้ ML และ Data Science

การลงทุนมีความเสี่ยง ควรศึกษาข้อมูลก่อนตัดสินใจ

📞 Support & Resources

Documentation:

- [README.md](#) - Overview & Quick Start
- [TRAINING_GUIDE.md](#) - Detailed guide
- [MAKEFILE_GUIDE.md](#) - All commands
- [SUMMARY.md](#) - Thai summary

Commands:

```
make help          # Show all commands  
make urls         # Show service URLs  
make check        # Check status
```

Service URLs:

- JupyterLab: <http://localhost:8888> (Token: fintech2025)
- Airflow: <http://localhost:8083> (admin/fintech2025)
- FastAPI: <http://localhost:8000/docs>
- Grafana: <http://localhost:3000> (admin/fintech2025)

🎉 Conclusion

Fintech Lab v6 เป็นระบบที่:

- ครบถ้วน - ทุกอย่างที่ต้องการ
- เสถียร - จัดการ errors อัตโนมัติ
- ยืดหยุ่น - หลายวิธีใช้งาน
- มีเอกสารดี - เอกสารครบถ้วน
- พร้อมใช้ - Production ready

เหมาะสมสำหรับ:

- การเรียนรู้ Machine Learning
- การศึกษา Time Series Prediction
- การพัฒนาทักษะ MLOps
- การทดลอง Algorithm ใหม่

Made with ❤️ for ducation and learning By Sojirat.S