靜宜大學資訊工程學系
Computer Science & Information Engineering

股票研究

報告日期:2025/11/13

專題學生:
資工四A 411147217 溫佑倫
資工四A 411147110 江奉奕
資工四A 411150812 張縉傑
資工四A 411147152 黃瀚杰
資工四A 411147398 張范裕

指導教授:羅峻旗
實驗室:主顧517

大綱

壹 簡介

1.專題動機

2.專題目的

3.主要內容

4.架構圖與工具

貳 系統說明

1.模型流程訓練

2.網頁

3.DEMO

參 結論

1.結語與未來展望

2.工作分配

3.參考資料

專題動機

1.股票市場資訊龐大，減少人為情緒與直覺判斷的誤差

2.利用歷史數據探索價格變化的潛在特徵

3.結合人工智慧技術提升預測準確率與穩定性

專題目的

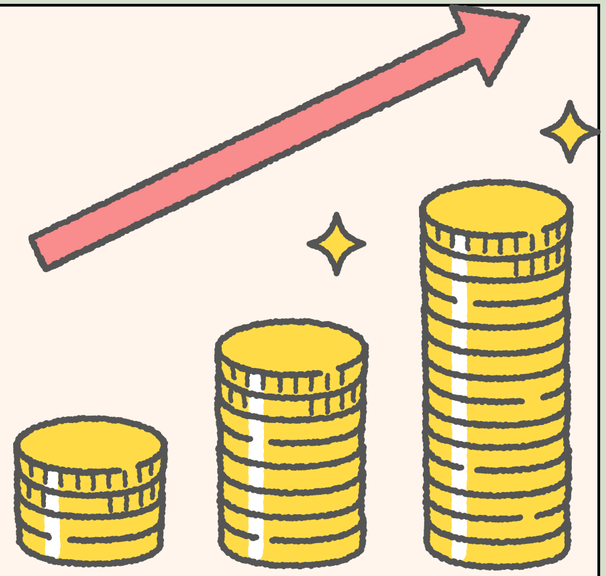
設計並實作出一個能夠即時查詢股票資訊，並利用歷史收盤價和成交量數據進行短期技術分析的工具

預測股價走勢

學習風險管理

獲利

主要內容



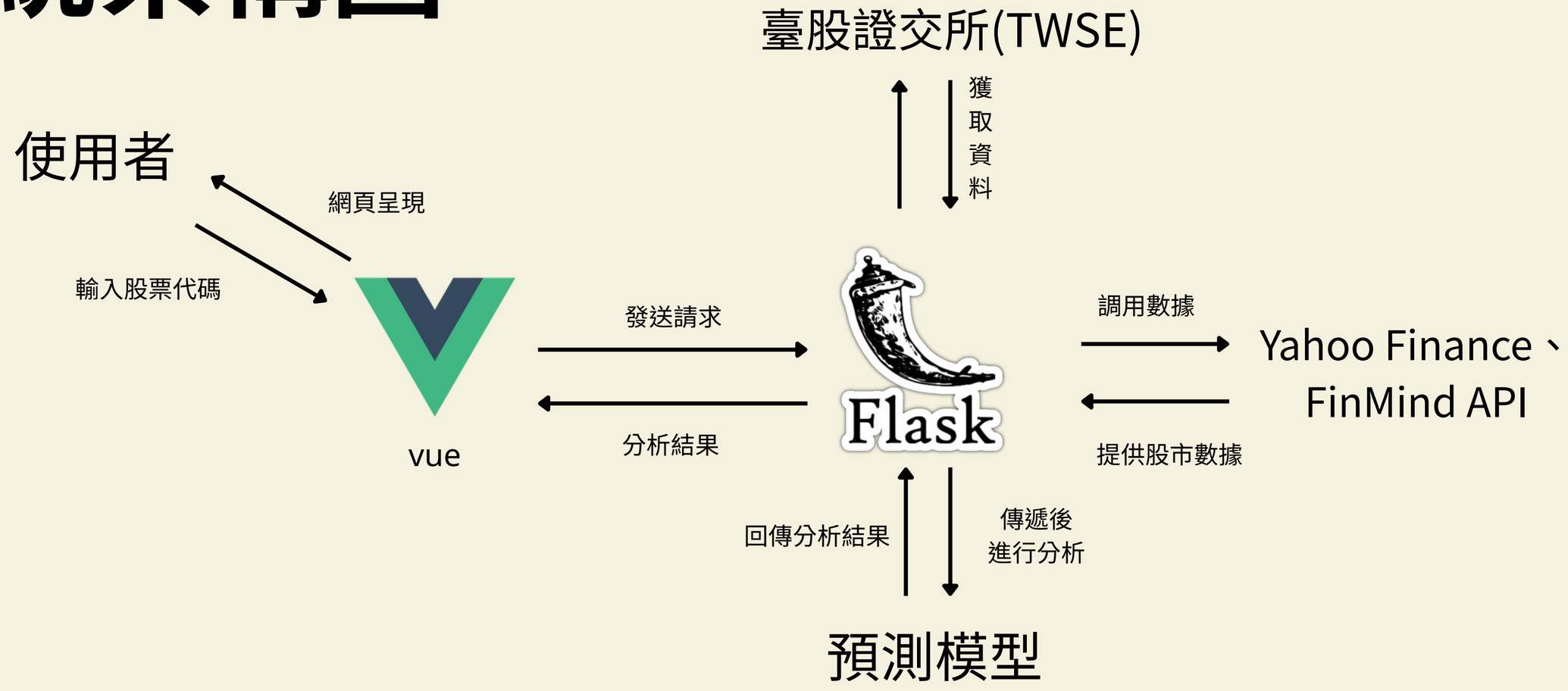
1.股票查詢系統(折線圖、k線圖)

2.三大法人(外資、投信、自營商)

3.各季EPS

4.模型預測股價趨勢

系統架構圖


臺股證交所(TWSE)
獲取資料
使用者
網頁呈現
輸入股票代碼
vue
發送請求
分析結果
Flask
調用數據
提供股市數據
Yahoo Finance、FinMind API
回傳分析結果
傳遞後進行分析
預測模型

開發工具

vscode

flask

vue

python

時間尺度	過去兩周至未來一周
預估所需資料	"收盤價", "最高價", "最低價", "成交股數"
模型與資料來源	CNN TWSE FinMind Yahoo Finance

實作股票挑選

水泥 1101 台泥
食品 1216 統一
化學 1722 台肥
汽車 2201 裕隆
半導 2303 聯電

電子 2317 鴻海
半導 2330 台積電
電腦 2331 精英
光電 2409 友達
通訊 2412 中華電

通訊 2498 宏達電
航空 2618 長榮航
觀光 2727 王品
金融 2882 國泰金
百貨 2903 遠百

模型訓練流程

資料來源與前處理

從臺灣證卷交易所抓取

2015年到2025年9月資料

```python
import requests
import pandas as pd
from datetime import datetime

def fetch_twse(stock_id, start_date, end_date):
    start = datetime.strptime(start_date, "%Y%m%d")
    end = datetime.strptime(end_date, "%Y%m%d")

    all_data = []
    current = start

    while current <= end:
        date_str = current.strftime("%Y%m%d")
        url = f"https://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date={date_str}&stockNo={stock_id}"
```


```
{
  "stat": "OK",
  "date": "20251001",
  "title": "114年10月 2330 台積電           各日成交資訊",
  "fields": [
    "日期",
    "成交股數",
    "成交金額",
    "開盤價",
    "最高價",
    "最低價",
    "收盤價",
    "漲跌價差",
    "成交筆數"
  ],
  "data": [
    [
      "114/10/01",
      "36,184,828",
      "48,288,398,204",
      "1,325.00",
      "1,350.00",
      "1,325.00",
      "1,325.00",
      "+20.00",
      "47,879"
    ],
    [
      "114/10/02",
      "32,693,509",
      "44,593,995,225",
      "1,360.00",
      "1,370.00",
      "1,355.00",
      "1,365.00",
      "+40.00",
      "61,282"
```


確認資料完整性

匯出所需時間尺度
所有交易日

→

與抓取資料對照檢查
是否有缺漏

```
PS C:\Users\User\Desktop\C\trainAI> python -u "c:\Users\User\Desktop\C\trainAI\find.py"
Total missing trading dates: 0
```

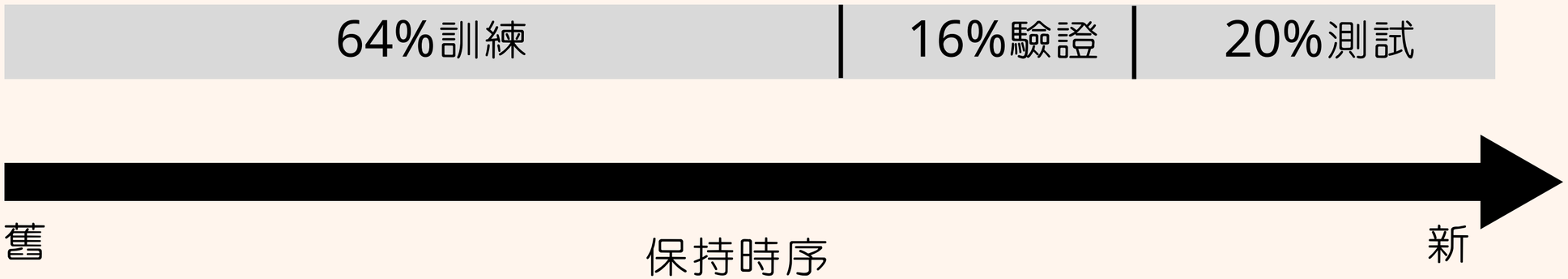
資料分類

訓練/測試資料分類

```
# train/test分類
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

驗證分類

```
# Train model
history = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_split=0.2,
```



技術指標與漲跌標準

採用特徵:

開盤價 最高價 最低價 成交量 移動平均(MA)

強弱指標(RSI) 平滑異同移動平均線(MACD)

```python
# 特徵
features = ['Close', 'High', 'Low', 'Volume', 'MA5', 'MA10', 'MA20', 'RSI', 'MACD','MACD_signal']
X = []
y = []

# 過去 14 天預測未來 7 天的平均值
for i in range(len(df) - 21):
    past_14 = df[features].iloc[i:i+14].values
    past_14_avg = df['Close'].iloc[i:i+14].mean()
    future_7_avg = df['Close'].iloc[i+14:i+21].mean()
    label = int(future_7_avg > past_14_avg)
    X.append(past_14)
    y.append(label)
X = np.array(X)
y = np.array(y)
```

漲跌標準:

過去14天平均股價 vs 未來7天平均股價

模型架構

cnn架構

```
# 模型
model = Sequential([
    Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(14, len(features))),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(32, activation='relu', kernel_regularizer=l1_l2(l1=0.001, l2=0.001)),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

資料平衡

```
weights = compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train)
class_weights = {int(k): round(float(v), 2) for k, v in zip(np.unique(y_train), weights)}
```

callback

```
# 回調
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True) #
reduce_lr = ReduceLROnPlateau(monitor='val_loss', patience=5, factor=0.5, min_lr=1e-5, verbose=1)
```

評估與繪圖

評估模型

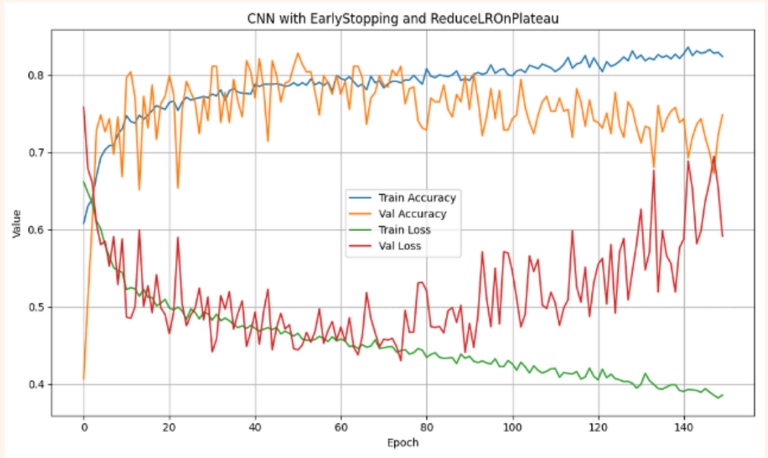
```
# 評估模型
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
y_pred = (model.predict(X_test) > 0.5).astype(int)

best_epoch_idx = np.argmin(history.history['val_loss'])
train_loss_at_best = history.history['loss'][best_epoch_idx]
val_loss_at_best = history.history['val_loss'][best_epoch_idx]
loss_diff_at_best = val_loss_at_best - train_loss_at_best
print(f"訓練損失: {train_loss_at_best:.6f}")
print(f"驗證損失: {val_loss_at_best:.6f}")
print(f"EarlyStopping 儲存模型權重時的損失差 (val - train): {loss_diff_at_best:.6f}")
```

繪圖

```
# 繪圖
plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
#red line
best_epoch = best_epoch_idx
plt.axvline(best_epoch, color='red', linestyle='--', label=f'EarlyStopping at Epoch {best_epoch+1}')
plt.text(best_epoch + 0.5, max(history.history['val_loss']), f'Epoch {best_epoch+1}', color='red', fontsize=10)
# blue line
stopped_epoch = early_stop.stopped_epoch
if stopped_epoch == 0:
    stopped_epoch = len(history.history['loss']) - 1
plt.axvline(stopped_epoch, color='blue', linestyle='--', label=f'Training Stopped at Epoch {stopped_epoch+1}')
plt.text(stopped_epoch + 0.5, max(history.history['val_loss']), f'Stop Epoch {stopped_epoch+1}', color='blue', fontsize=10)
```

訓練過程

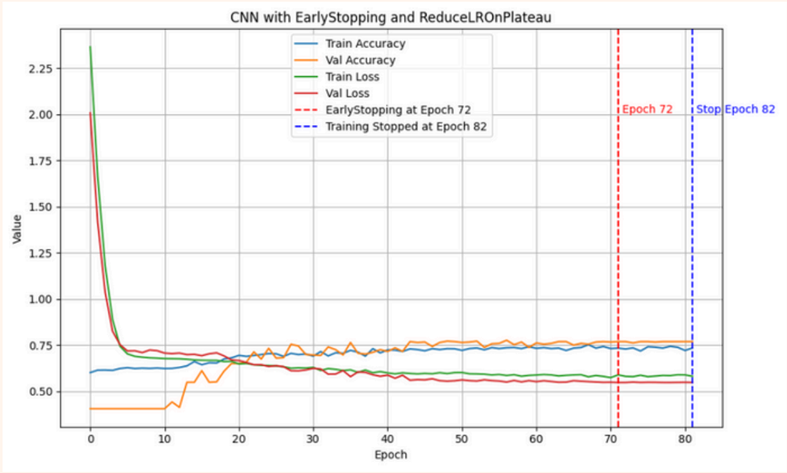
2330台積電



無正規化與訓練控制策略

150epoch

正規化與訓練控制策略

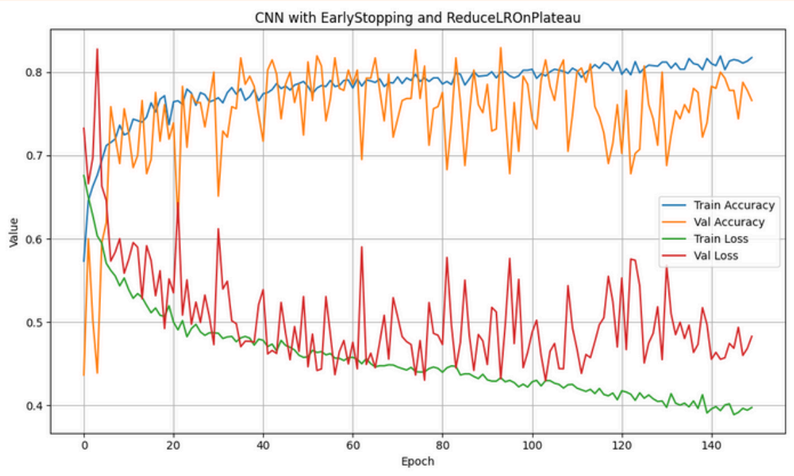


使用模型停止時的權重測試

```
Test Accuracy: 76.60%
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
訓練損失: 0.589824
驗證損失: 0.548445
EarlyStopping 儲存模型權重時的損失差 (val - train): -0.041379
Confusion Matrix:
 [[127  42]
 [ 79 269]]

Classification Report:
              precision    recall  f1-score   support

           0     0.6165    0.7515    0.6773       169
           1     0.8650    0.7730    0.8164       348

    accuracy                         0.7660       517
   macro avg     0.7407    0.7622    0.7469       517
weighted avg     0.7837    0.7660    0.7709       517
```

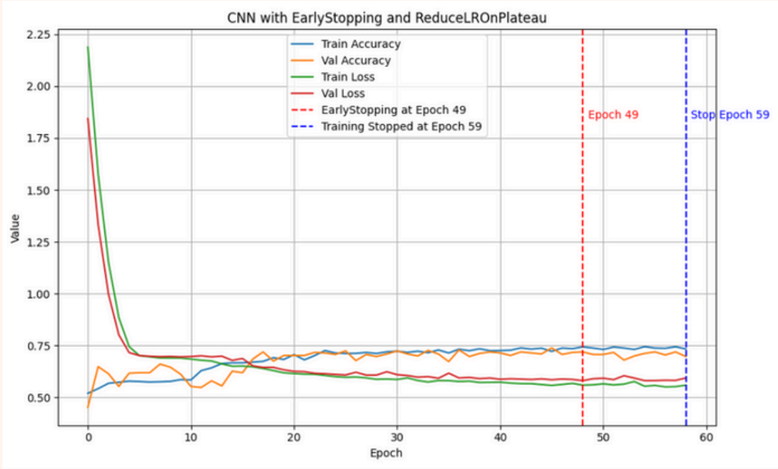
訓練過程

2303聯電



無正規化與訓練控制策略
150epoch

正規化與訓練控制策略



使用模型停止時的權重測試

```
Test Accuracy: 73.29%
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
訓練損失: 0.570468
驗證損失: 0.544742
EarlyStopping 儲存模型權重時的損失差 (val - train): -0.025726
Confusion Matrix:
 [[212  54]
 [ 83 164]]

Classification Report:
              precision    recall  f1-score   support

           0     0.7186    0.7970    0.7558       266
           1     0.7523    0.6640    0.7054       247

    accuracy                         0.7329       513
   macro avg     0.7355    0.7305    0.7306       513
weighted avg     0.7348    0.7329    0.7315       513
```

網頁

收盤價


全球股票查詢
股票代碼：2330
快速區間選擇：
1週 1個月 半年 1年
或手動輸入日期查詢：
起始日期：2025/05/04 結束日期：2025/11/04 查詢
圖表類型：
折線圖 K線圖
2330 收盤價
1,600
1,500
1,400
1,300
1,200
1,100
1,000
900

K線


股票查詢
股票代碼：2330
快速區間選擇：
1週 1個月 半年 1年
或手動輸入日期查詢：
起始日期：2025/05/07 結束日期：2025/11/07 查詢
圖表類型：
折線圖 K線圖
K線圖
1,600
1,500
1,400
1,300
1,200
1,100
1,000
900
2025-05-07
2025-05-27
2025-06-17
2025-07-07
2025-07-25
2025-08-15
2025-09-04
2025-09-24
2025-10-17
2025-11-07

AI 預測


已訓練台股
1101
1216
1722
2201
2303
2317
2330
2331
2409
2412
2498
2618
2727
2882
2903
預測結果
預測下跌
上漲機率：10.05%
下跌機率：89.95%
最新資料日：2025-11-07
最近 14 天收盤價
日期
收盤
2025-11-07 1,460
2025-11-06 1,465
2025-11-05 1,460
2025-11-04 1,505
2025-11-03 1,510
2025-10-31 1,500
2025-10-30 1,505
2025-10-29 1,505
2025-10-28 1,475
2025-10-27 1,480
2025-10-23 1,450
2025-10-22 1,460
2025-10-21 1,480
2025-10-20 1,480

三大法人

三大法人淨買賣

2330 | 近15日 | 張 | 包含今天 | 查詢

日期	外資（張）	投信（張）	自營商（張）	合計（張）
2025-11-06	-2,097	459	1,083	-554
2025-11-05	-30,801	538	2,031	-28,232
2025-11-04	2,907	939	-73	3,772
2025-11-03	3,528	497	783	4,808
2025-10-31	-7,726	292	3,058	-4,376
2025-10-30	-1,913	-171	-78	-2,162
2025-10-29	6,209	221	557	6,987
2025-10-28	-8,298	212	768	-7,318
2025-10-27	979	-301	2,346	3,023
2025-10-23	-8,491	99	546	-7,846
2025-10-22	-14,952	732	1,573	-12,647
2025-10-21	-300	1,583	236	1,519
2025-10-20	5,823	1,075	566	7,464
2025-10-17	-12,682	564	-21	-12,138
2025-10-16	-6,983	1,698	2,003	-3,282

EPS

EPS（近幾季）

股票代碼： 2330 季數： 8 查詢 EPS

年度	季別	EPS	TTM（近4季合計）	涵蓋期間
2025	Q2	15.36	56.31	2025-04-01~2025-06-30
2025	Q1	13.95	50.50	2025-01-01~2025-03-31
2024	Q4	14.45	45.25	2024-10-01~2024-12-31
2024	Q3	12.55	40.01	2024-07-01~2024-09-30
2024	Q2	9.55	35.60	2024-04-01~2024-06-30
2024	Q1	8.70	—	2024-01-01~2024-03-31
2023	Q4	9.21	—	2023-10-01~2023-12-31
2023	Q3	8.14	—	2023-07-01~2023-09-30

收盤價

```vue
<div style="max-width: 1000px; margin: auto;">
  <h2>📈 股票查詢</h2>

  <label>股票代碼：</label>
  <input v-model="symbol" placeholder="例:2330" required />

  <h4>快速區間選擇：</h4>
  <div>
    <button @click="setRange('1w')">1週</button>
    <button @click="setRange('1m')">1個月</button>
    <button @click="setRange('6m')">半年</button>
    <button @click="setRange('1y')">1年</button>
  </div>

  <h4>或手動輸入日期查詢：</h4>
  <form @submit.prevent="fetchStockData">
    <label>起始日期：</label>
    <input type="date" v-model="start" required />
    <label>結束日期：</label>
    <input type="date" v-model="end" required />
    <button type="submit">查詢</button>
  </form>

  <h4>圖表類型：</h4>
  <div>
    <button @click="chartType = 'line'">折線圖</button>
    <button @click="chartType = 'kline'">K 線圖</button>
  </div>

  <div v-if="error" style="color:red;">{{ error }}</div>

  <canvas id="lineChart" width="1000" height="500" v-if="chartType === 'line' && chartData.length"></canvas>
  <KLineChart v-if="chartType === 'kline' && klineData && klineData.length" :klineData="klineData" />
```

```python
# 收盤價折線圖資料
@app.route("/api/stock", methods=["POST"])
def get_stock_data():
    data = request.json
    symbol = data.get("symbol")
    start = data.get("start")
    end = data.get("end")

    if not symbol or not start or not end:
        return jsonify({"error": "請提供 symbol, start, end"}), 400

    try:
        end_date = datetime.strptime(end, "%Y-%m-%d") + timedelta(days=1)

        chosen = None
        stock = pd.DataFrame()
        for s in _yf_candidates(symbol):
            tmp = yf.download(s, start=start, end=end_date.strftime("%Y-%m-%d"))
            tmp = tmp.loc[start:end]
            if not tmp.empty:
                chosen = s
                stock = tmp
                break

        if stock.empty:
            return jsonify({"error": "查無資料"}), 404

        close_series = stock["Close"]
        if isinstance(close_series, pd.DataFrame):
            close_series = close_series.squeeze()

        response_data = {
            "symbol_used": chosen or symbol,
            "dates": stock.index.strftime("%Y-%m-%d").tolist(),
            "close_prices": close_series.round(2).tolist(),
        }
        return jsonify(response_data)

    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

K線

```vue
  <div ref="chartRef" class="kline-container"></div>
</template>

<script setup>
import { ref, watch, nextTick } from 'vue'
import * as echarts from 'echarts'

const props = defineProps({
  klineData: {
    type: Array,
    required: true
  }
})

const chartRef = ref(null)
let chartInstance = null

watch(() => props.klineData, async (newData) => {
  if (!newData.length) return
  await nextTick()
  renderChart()
}, { immediate: true })

function renderChart() {
  if (!chartRef.value) return
  if (!chartInstance) {
    chartInstance = echarts.init(chartRef.value)
  }

  const dates = props.klineData.map(item => item.date)
  const values = props.klineData.map(item => [
    item.open, item.close, item.low, item.high
  ])

  const option = {
    title: { text: 'K 線圖', left: 'center' },
    tooltip: { trigger: 'axis' },
    xAxis: {
      type: 'category',
      data: dates,
      boundaryGap: false,
      min: 'dataMin',
      max: 'dataMax' // ✅ 保證最後一筆資料在最右邊顯示
    },
```

```python
@app.route("/api/kline", methods=["POST"])
def get_kline_data():
    data = request.json
    symbol = data.get("symbol")
    start = data.get("start")
    end = data.get("end")

    if not symbol or not start or not end:
        return jsonify({"error": "請提供 symbol, start, end"}), 400

    try:
        end_dt = datetime.strptime(end, "%Y-%m-%d") + timedelta(days=1)

        chosen = None
        stock = pd.DataFrame()
        for s in _yf_candidates(symbol):
            tmp = yf.download(s, start=start, end=end_dt.strftime("%Y-%m-%d"))
            tmp = tmp.loc[start:end]
            if not tmp.empty:
                chosen = s
                stock = tmp
                break

        if stock.empty:
            return jsonify({"error": "查無資料"}), 404

        kline_data = []
        for dt_idx in stock.index:
            row = stock.loc[dt_idx]
            kline_data.append(
                {
                    "date": dt_idx.strftime("%Y-%m-%d"),
                    "open": float(row["Open"]),
                    "high": float(row["High"]),
                    "low": float(row["Low"]),
                    "close": float(row["Close"]),
                    "volume": int(row["Volume"]),
                }
            )

        return jsonify({"symbol_used": chosen or symbol, "data": kline_data})

    except Exception as e:
        print("❌ 錯誤：", e)
        return jsonify({"error": str(e)}), 500
```

AI預測

```vue
<div class="predict-wrap">
  <aside class="predict-side">
    <h3>🔮 已訓練台股</h3>
    <div v-if="modelsLoading">清單載入中…</div>
    <div v-else-if="modelsError" class="err">{{ modelsError }}</div>
    <ul v-else class="model-list">
      <li v-for="code in models" :key="code">
        <button
          class="model-btn"
          :class="{ active: selectedModel === code }"
          @click="predictClick(code)"
        >
          {{ code }}
        </button>
      </li>
    </ul>
  </aside>

  <section class="predict-panel">
    <h3>預測結果</h3>
    <div v-if="predLoading">預測中…</div>
  <div v-else-if="predError" class="err">{{ predError }}</div>
  <div v-else-if="pred">
      <div class="row">
        <span class="chip" :class="pred.label ? 'up' : 'down'">
          {{ pred.label ? '📈 預測上漲' : '📉 預測下跌' }}
        </span>
        <span class="chip">上漲機率：{{ (pred.up_prob*100).toFixed(2) }}%</span>
        <span class="chip">下跌機率：{{ (pred.down_prob*100).toFixed(2) }}%</span>
        <span class="chip">最新資料日：{{ pred.latest_date }}</span>
      </div>

      <h4>最近 14 天收盤價</h4>
      <table class="mini">
        <thead><tr><th>日期</th><th>收盤</th></tr></thead>
        <tbody>
          <tr v-for="r in [...pred.recent].reverse()" :key="r.Date">
            <td style="text-align:left">{{ r.Date }}</td>
            <td style="text-align:right">{{ Number(r.Close ?? 0).toLocaleString('zh-TW') }}</td>
          </tr>
        </tbody>
      </table>
    </div>
    <div v-else class="muted">從左側清單選一檔股票，就會顯示預測。</div>
  </section>
```

```python
#列出可用模型
from tensorflow.keras.models import load_model
from sklearn.preprocessing import MinMaxScaler

MODEL_DIR = os.path.dirname(os.path.abspath(__file__))
_model_cache = {}

@app.get("/api/predict/models")
def list_available_models():
    try:
        items = []
        for fn in os.listdir(MODEL_DIR):
            if fn.lower().endswith(".keras"):
                code = os.path.splitext(fn)[0]
                if code.isdigit():
                    items.append(code)
        items = sorted(set(items), key=lambda x: (len(x), x))
        return jsonify({"ok": True, "models": items})
    except Exception as e:
        return jsonify({"ok": False, "error": f"{type(e).__name__}: {e}"}), 500

# TWSE
def _twse_fetch(stock_id: str, start_date: str, end_date: str) -> pd.DataFrame:
    """
    從 TWSE 月報抓日資料，日期格式 yyyymmdd；回傳欄位：Date, Close, High, Low, Volume, Change
    """
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

    start = datetime.strptime(start_date, "%Y%m%d")
    end = datetime.strptime(end_date, "%Y%m%d")

    all_rows, fields = [], []
    cur = start.replace(day=1)
    while cur <= end:
        url = f"https://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date={cur.strftime('%Y%m%d')}&stockNo={stock_id}"
        try:
            res = requests.get(url, timeout=12, verify=False)
            data = res.json()
            if "data" in data:
                all_rows.extend(data["data"])
                fields = data["fields"]
        except Exception as e:
            print("TWSE error:", e)
```

三大法人

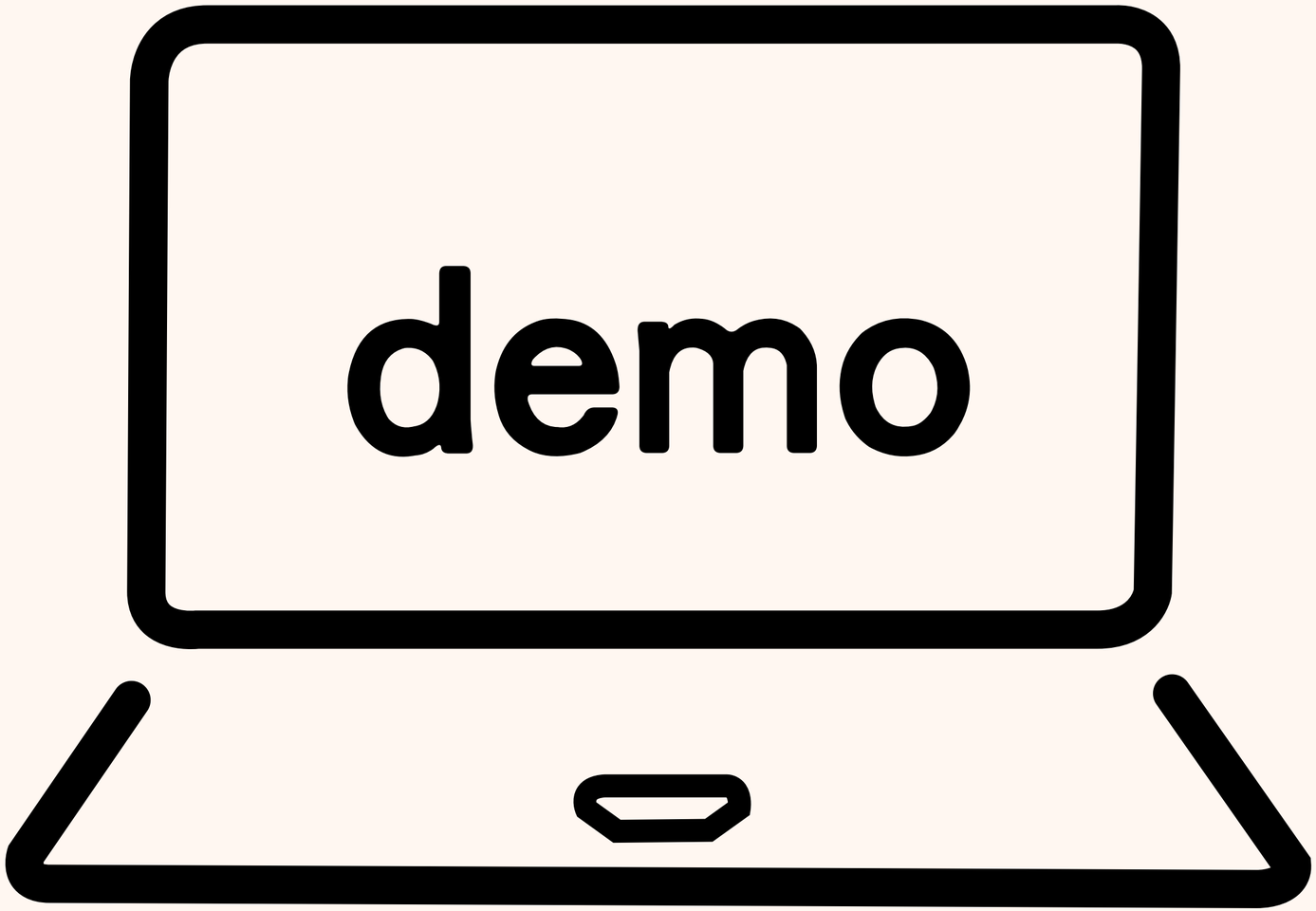
```html
<div class="card inst-card">
  <h3>三大法人淨買賣</h3>
  <div class="controls">
    <input v-model="instSymbol" placeholder="例:2330" @keyup.enter="fetchInst">


    <select v-model.number="instDays" title="區間" @change="fetchInst">
      <option :value="5">近5日</option>
      <option :value="10">近10日</option>
      <option :value="15">近15日</option>
      <option :value="20">近20日</option>
      <option :value="25">近25日</option>
      <option :value="30">近30日</option>

    </select>

    <select v-model="instUnit" title="單位" @change="fetchInst">
      <option value="lot">張</option>
      <option value="share">股</option>
    </select>

    <label style="display:flex;align-items:center;gap:6px;">
      <input type="checkbox" v-model="instIncludeToday" @change="fetchInst"> 包含今天
    </label>

    <button @click="fetchInst">查詢</button>
  </div>

  <div v-if="instError" style="color:#e11d48; margin-bottom:6px;">{{ instError }}</div>

  <table v-if="instRows.length">
    <thead>
      <tr>
        <th>日期</th>
        <th>外資 ({{ instUnit==='lot' ? '張' : '股' }})</th>
        <th>投信 ({{ instUnit==='lot' ? '張' : '股' }})</th>
        <th>自營商 ({{ instUnit==='lot' ? '張' : '股' }})</th>
        <th>合計 ({{ instUnit==='lot' ? '張' : '股' }})</th>
      </tr>
    </thead>
```

```python
@app.get("/api/inst")
def get_institutional_flows():
    from collections import defaultdict

    raw = request.args.get("stock", "")
    stock_id = normalize_tw_id(raw)
    if not stock_id or not stock_id.isdigit():
        return jsonify({"error": "請提供正確的台股代號，例如 ?stock=2330"}), 400

    # 參數
    try:
        days = int(request.args.get("days", 20))
    except Exception:
        days = 20
    days = max(1, min(days, 365))
    unit = (request.args.get("unit", "lot") or "lot").lower()  # lot=張, share=股
    include_today = request.args.get("include_today", "0").lower() in ("1", "true", "yes")

    FINMIND_API = "https://api.finmindtrade.com/api/v4/data"
    FINMIND_TOKEN = os.getenv("FINMIND_TOKEN", "")

    start_date = (date.today() - timedelta(days=days + 12)).isoformat()
    params = {
        "dataset": "TaiwanStockInstitutionalInvestorsBuySell",
        "data_id": stock_id,
        "start_date": start_date,
    }
    if FINMIND_TOKEN:
        params["token"] = FINMIND_TOKEN

    try:
        r = requests.get(FINMIND_API, params=params, timeout=20)
        r.raise_for_status()
        data = r.json().get("data", [])
    except Exception as e:
        return jsonify({"error": f"外部來源連線失敗：{e}"}), 502
```

EPS

```vue
<section class="eps-card">
  <h3>📈 EPS（近幾季）</h3>

  <div class="eps-row">
    <label>股票代碼：</label>
    <input v-model="symbol" placeholder="例:2330" style="max-width:200px;" />
    <label>季數：</label>
    <input type="number" v-model.number="epsQuarters" min="1" max="16" style="width:80px;" />
    <button @click="fetchEPS" :disabled="epsLoading">{{ epsLoading ? '查詢中…' : '查詢 EPS' }}</button>
    <span v-if="epsError" class="eps-error">{{ epsError }}</span>
  </div>

  <div v-if="epsLoading">載入中…</div>

  <table v-if="epsRows.length" class="eps-table">
    <thead>
      <tr>
        <th>年度</th>
        <th>季別</th>
        <th>EPS</th>
        <th>TTM（近4季合計）</th>
        <th>涵蓋期間</th>
      </tr>
    </thead>
    <tbody>
      <tr v-for="(r, idx) in epsRows" :key="idx">
        <td>{{ r.year }}</td>
        <td>Q{{ r.season }}</td>
        <td>{{ r.eps == null ? '–' : Number(r.eps).toFixed(2) }}</td>
        <td>{{ r.ttm == null ? '–' : Number(r.ttm).toFixed(2) }}</td>
        <td>{{ r.range }}</td>
      </tr>
    </tbody>
  </table>

  <div v-else-if="!epsLoading && !epsError">
    目前沒有資料，請換個代碼或增加季數
  </div>
```

```python
def fetch_eps_recent_quarters(stock_id: str, quarters: int = 8, token: str = ""):
    import pandas as pd
    from datetime import datetime

    url = "https://api.finmindtrade.com/api/v4/data"
    stock_id = normalize_tw_id(stock_id)
    finmind_token = os.getenv("FINMIND_TOKEN", token or "")

    params = {
        "dataset": "TaiwanStockFinancialStatements",
        "data_id": stock_id,
        "report_type": "季報",
        "start_date": "2010-01-01",
        "end_date": datetime.today().strftime("%Y-%m-%d"),
    }
    if finmind_token:
        params["token"] = finmind_token

    try:
        res = requests.get(url, params=params, timeout=30)
        res.raise_for_status()
        payload = res.json()
        data = payload.get("data", [])
        if not data:
            return pd.DataFrame()

        df = pd.DataFrame(data)

        def is_eps_type(t: str) -> bool:
            if not isinstance(t, str):
                return False
            t_strip = t.strip()
            t_upper = t_strip.upper()
            if "EPS" in t_upper or "EARNINGSPERSHARE" in t_upper:
                return True
            return ("每股" in t_strip)

        df = df[df["type"].apply(is_eps_type)].copy()
        if df.empty:
            return pd.DataFrame()
```


demo

結語

本研究以股票歷史交易資料結合多種技術指標，建立卷積神經網路模型以預測股票漲跌趨勢。結果顯示，適當的正規化與訓練策略能有效減少過擬合現象，提升模型在驗證資料上的準確率與穩定性。整體而言，深度學習方法確實具有應用於股價趨勢預測的潛力。

未來展望

未來可嘗試結合更多面向的資料，如新聞文本分析或總體經濟指標，並挑戰更長時間序列的建模，以同時捕捉長期趨勢與短期波動。此外，也可比較不同深度學習架構於不同時間尺度下的表現，進一步提升模型的預測準確率與泛化能力，發展更具實務價值的智慧投資輔助系統。

工作分配

溫佑倫： 訓練模型、前端開發
黃瀚杰： 訓練模型、前端開發
張范裕： 訓練模型、前端開發
江奉奕： 訓練模型、後端開發
張縉傑： 訓練模型、後端開發

參考資料

Vue

TWSE

FinMind

flask

Chatgpt

Yahoo股市

Copliot

THANK YOU!