CHƯƠNG 4. XỬ LÝ DỮ LIỆU CHUỖI THỜI GIAN VÀ DỮ LIỆU BẢNG

4.1. Giới thiệu (dữ liệu) chuỗi thời gian

4.2. Trực quan hoá chuỗi thời gian

4.3. Xử lí dữ liệu từ tập tin

4.4. Tiền xử lý chuỗi thời gian

4.5. Tạo đối tượng dữ liệu chuỗi thời gian với Python

4.6. Dự đoán dựa trên dự liệu tài chính dùng các mô hình học học máy

- 4.6.1. k-NN
- 4.6.2. ARIMA
- 4.6.3. Xgboost
- 4.6.4. Các mô hình học máy khác: RF,

4.7 Xử lý dữ liệu dạng bảng

4.1. Giới thiệu Dữ Liệu Chuỗi Thời Gian (Time Series Data)

Dữ liệu kinh tế - tài chính: dữ liệu chéo, dữ liệu chuỗi thời gian, và dữ liệu mảng (Panel data). Hầu hết là dữ liệu chuỗi thời gian

Dữ liệu chuỗi thời gian là dữ liệu được thu thập theo định kỳ thời gian., thứ tự lấy mẫu (hay quan sát) là rất quan trọng và không thể thay đổi hoặc loại bỏ bất kỳ một quan sát nào.

Ví dụ: Giá cổ phiếu, các chỉ số kinh tế-tài chính, kinh doanh,…

Đặc điểm : Tính xu hướng, Tính chu kỳ, Tính mùa vụ, Tính bất thường và biến đổi ngẫu nhiên.

Đặc biệt là dữ liệu chuỗi thời gian có tính hành vi.

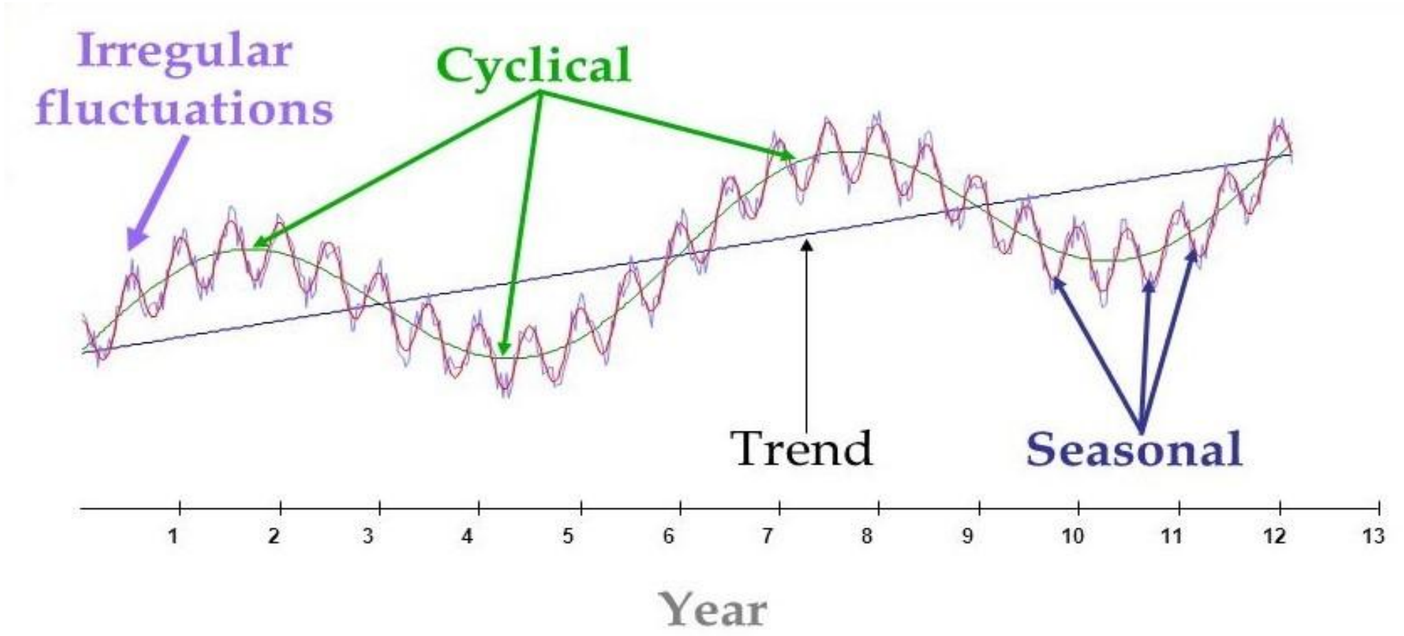
4.2. Trực quan hóa chuỗi thời gian

Trực quan hóa dữ liệu giúp hiểu rõ hơn về:

. Xu hướng (**Trend**): Dữ liệu tăng, giảm theo thời gian.

. Tính chất mùa vụ (**Seasonality**): Lặp đi lặp lại tại các khoảng thời gian cố định.

. Nhiễu (**Noise**): Biến động bất thường.



Cách thực hiện:

1. **Matplotlib** / **Seaborn**: Biểu đồ tuyến tính (line plot).
2. **Pandas**: Hiển thị chuỗi thời gian từ khung dữ liệu.

4.3. Xử lý dữ liệu từ tập tin

Trong công việc thực tế, dữ liệu thường được lưu trữ trong các tệp như **CSV, Excel** hoặc **Database**. Xử lý dữ liệu từ tệp (file) giúp chuyển dữ liệu thô thành dạng bảng hoặc tensor để có thể phân tích.

Các bước thực hiện

1. **Đọc dữ liệu** từ tệp tin bằng thư viện (**Pandas, Numpy, Matplotlib**).
2. **Xử lý lỗi định dạng**: Loại bỏ hàng lỗi, thiếu, hoặc định dạng thời gian.
3. **Xây dựng index thời gian** (datetime index): Biến cột ngày/thời gian thành mốc thời gian dùng để phân tích.

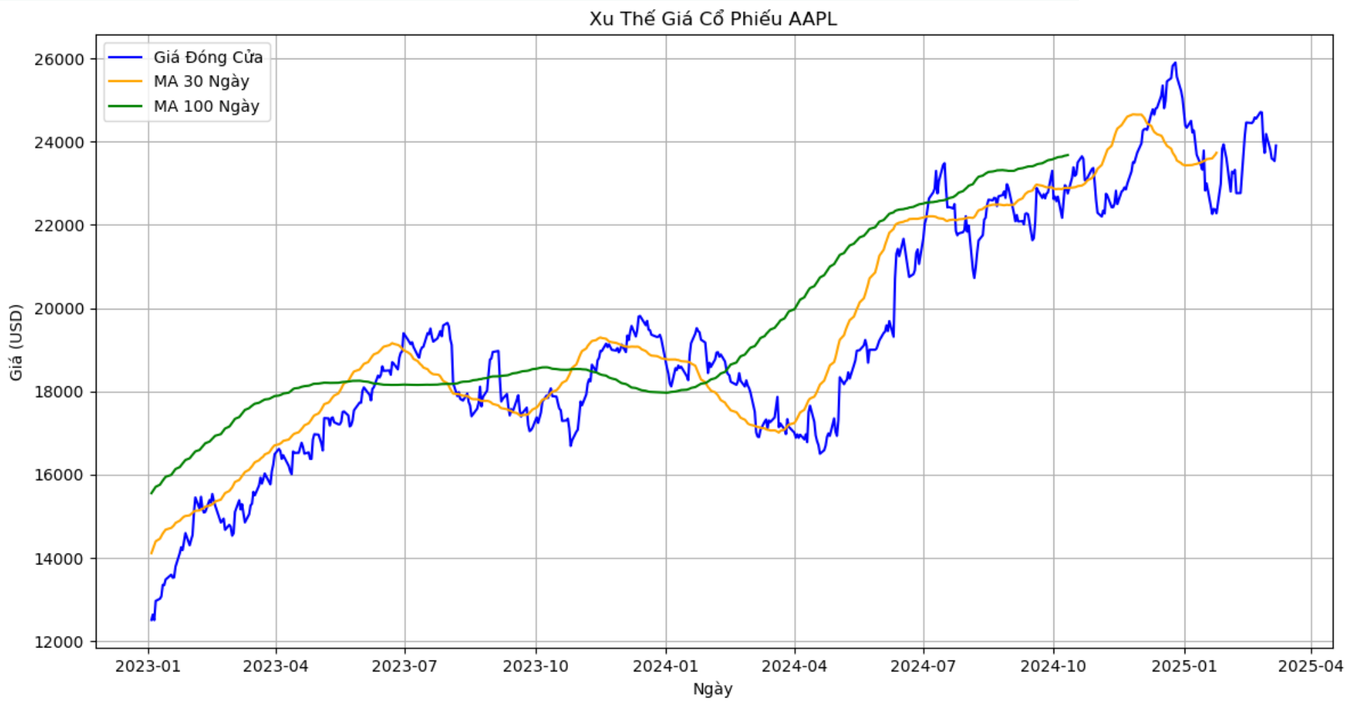
Ví dụ cụ thể:

```
import pandas as pd
# Bước 1: Đọc dữ liệu từ file CSV
file_path = '/Users/mrhai/Desktop/AAPL.xlsx'
df = pd.read_excel(file_path)
print(df.head())
# Chuyển đổi cột Date thành kiểu datetime
df['Date'] = pd.to_datetime(df['Date'])


# Đặt Date làm chỉ số cho DataFrame
df.set_index('Date', inplace=True)
```

```
# Tính toán đường trung bình động 30 ngày và 100 ngày
df['MA30'] = df['Close'].rolling(window=30).mean()
df['MA100'] = df['Close'].rolling(window=100).mean()

# Thêm cột tháng và năm để phân tích theo mùa vụ
df['Month'] = df.index.month
df['Year'] = df.index.year
```

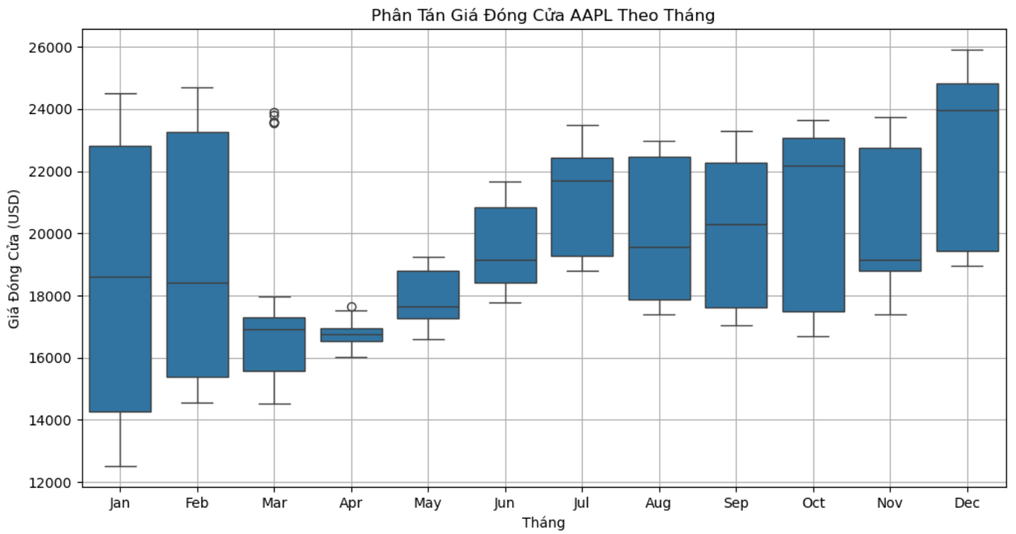
Biểu đồ này sẽ hiển thị giá đóng cửa và các đường trung bình động để hiểu rõ hơn về xu thế.

```
import matplotlib.pyplot as plt

# Vẽ biểu đồ giá đóng cửa và các đường trung bình động
plt.figure(figsize=(14, 7))
plt.plot(df['Close'], label='Giá Đóng Cửa', color='blue')
plt.plot(df['MA30'], label='MA 30 Ngày', color='orange')
plt.plot(df['MA100'], label='MA 100 Ngày', color='green')
plt.title('Xu Thế Giá Cổ Phiếu AAPL')
plt.xlabel('Ngày')
plt.ylabel('Giá (USD)')
plt.legend()
plt.grid()
plt.show()
```



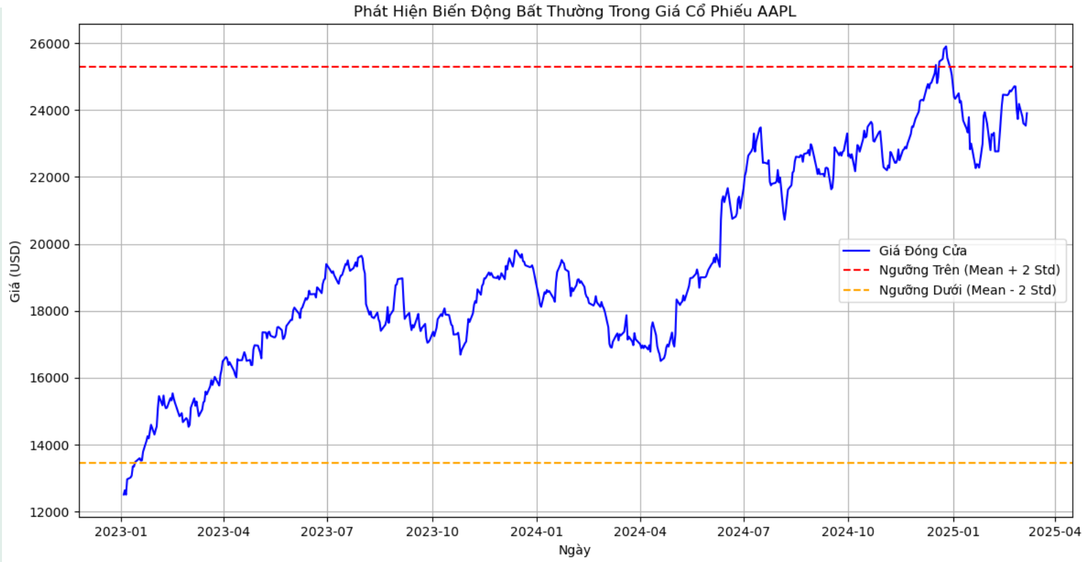
Chúng ta sẽ sử dụng biểu đồ box plot để trực quan hóa sự phân tán giá theo tháng, cho thấy tính mùa vụ của giá cổ phiếu.

```
import seaborn as sns

# Vẽ biểu đồ box plot để phân tích giá theo từng thán
plt.figure(figsize=(12, 6))
sns.boxplot(x='Month', y='Close', data=df)
plt.title('Phân Tán Giá Đóng Cửa AAPL Theo Tháng')
plt.xlabel('Tháng')
plt.ylabel('Giá Đóng Cửa (USD)')
plt.xticks(range(0, 12), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.savefig('/Users/mrhai/Desktop/biểu_do2.jpg',
bbox_inches='tight')
plt.grid()
plt.show()
```



 # Lưu biểu đồ thành file ảnh
plt.savefig('/Users/mrhai/Desktop/biểu_do2.jpg', bbox_inches='tight')
 # Thay đổi định dạng nếu cần

```
# Tính các ngưỡng cho biến động bất thường
mean_close = df['Close'].mean()
std_close = df['Close'].std()
threshold_upper = mean_close + 2 * std_close
threshold_lower = mean_close - 2 * std_close
# Vẽ biểu đồ giá cùng với ngưỡng bất thường
plt.figure(figsize=(14, 7))
plt.plot(df['Close'], label='Giá Đóng Cửa', color='blue')
plt.axhline(y=threshold_upper, color='red', linestyle='--',
label='Ngưỡng Trên (Mean + 2 Std)')
plt.axhline(y=threshold_lower, color='orange', linestyle='--',
label='Ngưỡng Dưới (Mean - 2 Std)')
plt.title('Phát Hiện Biến Động Bất Thường Trong Giá Cổ Phiếu AAPL')
plt.xlabel('Ngày')
plt.ylabel('Giá (USD)')
plt.legend()
plt.grid()
plt.show()
```



4.4. Tiền xử lý chuỗi thời gian

Chuỗi thời gian thường chứa nhiễu hoặc dữ liệu không đầy đủ. Tiền xử lý giúp chuẩn hóa, phát hiện xu hướng, và tối ưu hóa dữ liệu phục vụ mô hình.

Các bước tiền xử lý dữ liệu chuỗi thời gian

1. **Xử lý giá trị bị thiếu**: Dùng phương pháp **interpolation** hoặc **forward fill** để điền giá trị thiếu.
2. **Loại bỏ nhiễu**: Sử dụng Bộ lọc trung bình trượt (**moving average filter**) để làm mịn dữ liệu.
3. **Phân tích định kỳ (Resampling)**: Chuyển đổi chuỗi thời gian sang các khoảng thời gian ngắn hơn hoặc dài hơn (vd: ngày sang tháng).
4. **Chuẩn hóa tĩnh**: Biến đổi log hoặc z-score để loại bỏ giá trị cực đoan.

4.5. Tạo đối tượng dữ liệu chuỗi thời gian với Python

Python hỗ trợ xử lý dữ liệu chuỗi thời gian bằng **Pandas**, cung cấp các công cụ xử lý datetime mạnh mẽ.

Ví dụ cụ thể

```
import pandas as pd

# Tạo chuỗi thời gian
date_rng = pd.date_range(start='2022-01-01', end='2022-01-10', freq='D’)

time_series = pd.Series(range(len(date_rng)), index=date_rng)

print(time_series)
```

4.6 CÁC THUẬT TOÁN ỨNG DỤNG TRONG DỮ LIỆU CHUỖI THỜI GIAN

1. K-Nearest Neighbors (k-NN)

K-NN có thể được sử dụng để dự đoán giá trị của mục tiêu dựa trên các điểm dữ liệu gần kề trong chuỗi thời gian.

Ý tưởng chính của thuật toán

- **Dựa vào "k" láng giềng gần nhất (k điểm dữ liệu gần nhất), sử dụng khoảng cách để dự đoán giá trị mới.**

Các bước thực hiện

1. **Tiền xử lý chuỗi thời gian.**
2. **Tách dữ liệu lịch sử thành các cửa sổ (window size) để làm đặc trưng.**
3. **Tính khoảng cách (ví dụ: khoảng cách Euclidean) giữa các điểm để tìm "k" điểm gần nhất.**
4. **Dự đoán giá trị mới bằng cách trung bình các giá trị từ "k" điểm gần nhất.**

2. ARIMA (Auto-Regressive Integrated Moving Average)

- **Dự báo ngắn hạn** về xu hướng doanh số, lãi suất, giá cổ phiếu, hoặc chi phí hàng hóa.
- Áp dụng tốt cho chuỗi thời gian với **xu hướng (trend)** và **tính mùa vụ (seasonality)** rõ ràng.

Thuật toán

- **AR (Auto-regression - hồi quy tự hồi quy)**: Mô hình dựa trên giá trị của chính chuỗi thời gian trước đó (y_t được biểu diễn bởi

- **I (Integrated – sai phân)**: Xử lý chuỗi thời gian không ổn định (non-stationary) bằng cách tính chênh lệch (difference) giữa các giá trị.
- **MA (Moving Average - trung bình động)**: Mô hình dựa trên nhiễu của điểm dữ liệu trong quá khứ.

Các bước thực hiện

1. Chuẩn hóa và kiểm tra tính ổn định (stationarity) của chuỗi thời gian.
2. Tính các tham số ARIMA (p, d, q) bằng đồ thị ACF, PACF.
3. Xây dựng và huấn luyện mô hình, dự đoán giá trị tương lai.

Ví dụ (Dự báo lợi nhuận hàng tháng):

```
from pandas.plotting import autocorrelation_plot
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
import pandas as pd
#    Dữ liệu doanh thu hàng tháng (giả lập)
data = {'Month': pd.date_range(start-2023-01', periods=10, freq='M'), 'Profit': [200, 220, 215,
230, 240, 250, 260, 280, 300, 320]}
df = pd.DataFrame(data)
df.set_index('Month', inplace=True)

#    Autocorrelation plot để kiểm tra sự phụ thuộc giữa các giá trị
plt.figure()
autocorrelation_plot(df['Profit'])
plt.show()
#    Xây dựng mô hình ARIMA
model = ARIMA(df['Profit'], order=(1, 1, 1)) # (p=1, d=1, q=1)
model_fit = model.fit()
print(model_fit.summary())
# Dự đoán lợi nhuận trong 3 tháng tiếp theo
forecast = model_fit.forecast(steps=3)
print("Dự đoán lợi nhuận tiếp theo:", forecast) * •
```

3. XGBoost (Extreme Gradient Boosting)

Ý nghĩa

- **Dự báo phức tạp** với các chuỗi thời gian mà quan hệ của các đặc trưng là phi tuyến tính.
- Được áp dụng rộng rãi cho dữ liệu multi-dimensional, ví dụ:

Dự đoán tăng trưởng doanh nghiệp từ nhiều yếu tố đầu vào (doanh số, chi phí, sự kiện thị trường).

- Dự đoán giá trị cổ phiếu dựa trên các tín hiệu giao dịch.

Ý tưởng chính của thuật toán

- Kết hợp nhiều cây quyết định nhỏ (weak learners) để giảm lỗi của mô hình.
- Áp dụng trên chuỗi thời gian bằng cách tạo các **đặc trưng** từ chuỗi thời gian (windows).

Các bước thực hiện

1. Chuyển chuỗi thời gian thành bảng dữ liệu (bằng sliding window).
2. Phân chia dữ liệu thành train/test.
3. Áp dụng mô hình XGBoost để huấn luyện và dự đoán.

Ví dụ thực tế trong kinh doanh (Dự báo lượng khách đến cửa hàng):

```
import xgboost as xgb
import pandas as pd
import numpy as np
#    Dữ liệu giả lập: số lượng khách hàng hàng ngày
customers = {'Date': pd.date_range(start='2025-01-01', periods=15, freq='D'),
'Customers': [200, 210, 220, 215, 230, 240, 250, 260, 255, 265, 275, 280, 290, 300, 310]}
df = pd.DataFrame(customers)
#    Feature engineering: tạo sliding window (cửa sổ chuyển động)
X = []
y = []
window_size = 3
for i in range(len(df) - window_size): X.append(df['Customers'][i:i+window_size].values)
y.append(df['Customers'][i+window_size])
X = np.array(X)
y = np.array(y)
```

```
#    Chia train/test
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:] y_train, y_test = y[:train_size], y[train_size:]

#    Huấn luyện mô hình XGBoost
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100)
model.fit(X_train, y_train)

#    Dự đoán số khách hàng tiếp theo
preds = model.predict(X_test)
print(f"Thực tế: {y_test}")
print(f"Dự đoán: {preds}")
```

Dữ liệu dạng bảng thường đi đôi với bài toán phân tích dữ liệu và dự đoán phức tạp hơn:

Áp dụng các thuật toán

1. **Logistic Regression, Linear Regression**: Dự đoán doanh số hoặc phân loại sản phẩm.
2. **Random Forest**: Xử lý bài toán quyết định kinh doanh (chọn khuyến mãi đúng tệp khách hàng).
3. **Gradient Boosting (XGBoost, Catboost)**: Dự đoán doanh thu theo mùa, khách hàng tiềm năng.

PHẦN 2: XỬ LÝ DỮ LIỆU DẠNG BẢNG

1. Dữ liệu dạng bảng trong bài toán kinh doanh

Dữ liệu dạng bảng là gì?

Dữ liệu dạng bảng thường bao gồm các hàng và cột. Mỗi **hàng (row)** đại diện cho một đối tượng (sản phẩm hoặc giao dịch). Mỗi **cột (column)** là một đặc trưng (feature) liên quan như:

- **Hành vi khách hàng** (độ tuổi, số lần mua sắm, số tiền đã chi tiêu).
- **Hiệu suất doanh thu** (tháng, quý, lợi nhuận mỗi sản phẩm).
- **Sự tương tác** (số lần khách hàng bấm quảng cáo, tranh chấp thanh toán,...).

Bài toán dạng bảng phổ biến trong kinh doanh

Loại mô hình	Thuật toán	Mô tả	Ứng dụng
Dự báo	**Hồi quy tuyến tính**	Mô hình đơn giản để dự đoán giá trị liên tục dựa trên một hoặc nhiều biến độc lập.	Dự đoán doanh thu, chi phí, lợi nhuận cho một sản phẩm cụ thể.
	Hồi quy Logistic	Sử dụng cho các biến phụ thuộc nhị phân để dự đoán xác suất sự kiện xảy ra.	Dự đoán khả năng khách hàng sẽ thực hiện mua hàng hay không.
	ARIMA (AutoRegressive Integrated Moving Average)	Mô hình thống kê cho dữ liệu chuỗi thời gian, thường dùng cho dự đoán giá trị trong tương lai.	Dự đoán xu hướng giá chứng khoán, tỷ giá hối đoái theo thời gian.
	Học sâu (Deep Learning)	Sử dụng mạng nơ-ron để dự đoán giá trị liên tục, có khả năng xử lý dữ liệu phức tạp.	Dự đoán giá bất động sản dựa trên nhiều yếu tố như vị trí, diện tích, tiện ích.

Loại mô hình	Thuật toán	Mô tả	Ứng dụng
Phân loại	**Cây quyết định (Decision Tree)**	Cấu trúc cây để quyết định dựa trên các biến đầu vào, có thể tạo ra các quy tắc phân loại.	Phân loại khách hàng thành "tiềm năng", "không tiềm năng" dựa trên hành vi mua sắm.
	Random Forest	Kết hợp nhiều cây quyết định để cải thiện độ chính xác và giảm thiểu overfitting.	Phân loại khách hàng, nhận diện gian lận trong giao dịch tài chính.
	Support Vector Machine (SVM)	Phân loại dữ liệu bằng cách tìm kiếm hyperplane tối ưu trong không gian đa chiều.	Phân loại các giao dịch thành "hợp lệ" và "nghi ngờ".
	Học sâu (Deep Learning)	Sử dụng mạng nơ-ron tích chập (CNN) hoặc mạng nơ-ron hồi tiếp (RNN) cho phân loại.	Phân loại văn bản, chẳng hạn như phân tích cảm xúc từ phản hồi của khách hàng.

Loại mô hình	Thuật toán	Mô tả	Ứng dụng
Phân cụm	**K-Means**	Phân cụm dữ liệu thành K nhóm dựa trên khoảng cách giữa các điểm.	Phân khúc thị trường, xác định các nhóm khách hàng tương đồng.
	Phân cụm không gian dựa trên mật độ của các ứng dụng có nhiễu (DBSCAN: Density-Based Spatial Clustering of Applications with Noise)	Nhóm các điểm có mật độ dày đặc với nhau và phát hiện các điểm nhiễu.	Phát hiện gian lận trong ngân hàng, phân tích địa lý của khách hàng.
	Phân cụm phân cấp (Hierarchical Clustering)	Xây dựng một cây phân cấp để tổ chức các cụm, từ các cụm nhỏ đến lớn.	Xây dựng phân khúc thị trường hoặc phân loại sản phẩm dựa trên tính năng tương đồng.
	Học sâu (Deep Learning)	Sử dụng mạng nơ-ron phi cấu trúc để nhóm dữ liệu phức tạp mà không cần gán nhãn trước.	Phân cụm các sản phẩm tương tự để đề xuất cho khách hàng mua thêm.

Ví dụ: Dự báo lợi nhuận

Ngày	Khách hàng mới	Doanh số ($)	Chi phí quảng cáo ($)	Lợi nhuận ($) (mục tiêu)
01/01/2023	50	5000	1000	4000
02/01/2023	45	4500	900	3600
03/01/2023	70	7000	1500	5500

Bài toán sử dụng mô hình học máy sẽ: **Dự đoán lợi nhuận $ dựa trên các đặc trưng.**

Ví dụ: Dự đoán lợi nhuận ($)

```
import pandas as pd

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

#    Tạo dữ liệu giả lập dạng bảng

data = {'Khách_hàng_mới': [50, 45, 70, 65, 60],

'Doanh_số': [5000, 4500, 7000, 6500, 6000],

'Chi_phí_quảng_cáo': [1000, 900, 1500, 1200, 1100],

'Lợi_nhuận': [4000, 3600, 5500, 5000, 4900]} # Đây chính là giá trị target (output)
```

```
df = pd.DataFrame(data)

#   Tách dữ liệu (biến độc lập và phụ thuộc)
X = df[['Khách_hàng_mới', 'Doanh_số', 'Chi_phí_quảng_cáo']] # Biến input (features)
y = df['Lợi_nhuận'] # Biến output

#   Chia dữ liệu train và test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#   Huấn luyện mô hình Linear Regression

model = LinearRegression()

model.fit(X_train, y_train)

#   Dự đoán và đánh giá y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("Hệ số hồi quy các đặc trưng:", model.coef_)
print("Intercept (Giao với trục):", model.intercept_)
```

Kết quả:

- **Mối quan hệ tuyến tính giữa các đặc trưng:**

Lợi nhuận = W1 * * Khách hàng mới + W2 * Doanh số + W3 * Chi phí quảng cáo + Bias.

2.2. Random Forest - Dự đoán chính xác hơn

Ý nghĩa trong kinh doanh

- **Mô hình Random Forest là một tập hợp các cây quyết định để cải thiện độ chính xác. Tính phi tuyến giúp dự đoán chính xác hơn trong kinh doanh thực tế.**

Ví dụ: Dự đoán lợi nhuận ($) sử dụng Random Forest.

```
from sklearn.ensemble import RandomForestRegressor

#    Huấn luyện mô hình Random Forest Regressor

model_rf    =    RandomForestRegressor(n_estimators=100,    random_state=42)

 model_rf.fit(X_train, y_train)

#    Dự đoán

y_pred_rf = model_rf.predict(X_test)

print("MSE (Random Forest):", mean_squared_error(y_test, y_pred_rf))
```

- **Lưu ý:**
- Random Forest cho kết quả tốt hơn trên những dữ liệu có độ phi tuyến phức tạp.
- Hữu dụng trong dự đoán **doanh số dài hạn**, vì doanh số thường phi tuyến.

2.3. Phân loại với Logistic Regression

- Logistic Regression dùng để giải quyết bài toán phân loại.

Ví dụ:
- **Khách hàng có rời bỏ dịch vụ không (Customer Chum)?**
- **Xác định giao dịch gian lận (Fraud Detection)?**

Ví dụ: Phân loại khách hàng sẽ rời bỏ dịch vụ (churn)

Ví dụ: Phân loại khách hàng sẽ rời bỏ dịch vụ (churn)

Tuổi	Tần suất Mua hàng	Thời gian sử dụng dịch vụ (năm)	Tiền đã chi tiêu ($)	Rời bỏ (Yes/No)
25	10	1.2	500	Yes
45	30	3.0	2500	No
32	15	2.2	1200	Yes

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score


#    Dữ liệu khách hàng
data = {'Tuổi': [25, 45, 32, 40, 29],
        'Tan_suất_mua': [10, 30, 15, 20, 12],
        'Thời_gian_sử_dỤng': [1.2, 3.0, 2.2, 2.8, 1.4],
        'Chi_tiêu': [500, 2500, 1200, 1600, 800],
        'Rời_bỏ': [1, 0, 1, 0, 1]} # 1 = Yes, 0 = No

df = pd.DataFrame(data)

#    Tách biến và dữ liệu
X = df[['Tuổi', 'Tần_suất_mua', 'Thời_gian_sử_dụng', 'Chi_tiêu']]

y = df['Rời_bỏ']
```

```
#    Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

#    Huấn luyện mô hình Logistic Regression
model = LogisticRegression() model.fit(X_train, y_train)

#    Dự đoán trên tập test
y_pred = model.predict(X_test)
print("Độ chính xác:", accuracy_score(y_test, y_pred))
```

3. Nâng cao: Sử dụng Gradient Boosting (XGBoost) để dự đoán và phân loại

- **Gradient Boosting** sử dụng hiệu quả trong dự đoán dạng bảng phức tạp (multi-feature).
- Ứng dụng trong các bài toán kinh doanh lớn: **Hiệu suất khách hàng, phân loại giao dịch, phân tích tài chính.**

Ví dụ: Dự đoán doanh thu với XGBoost:

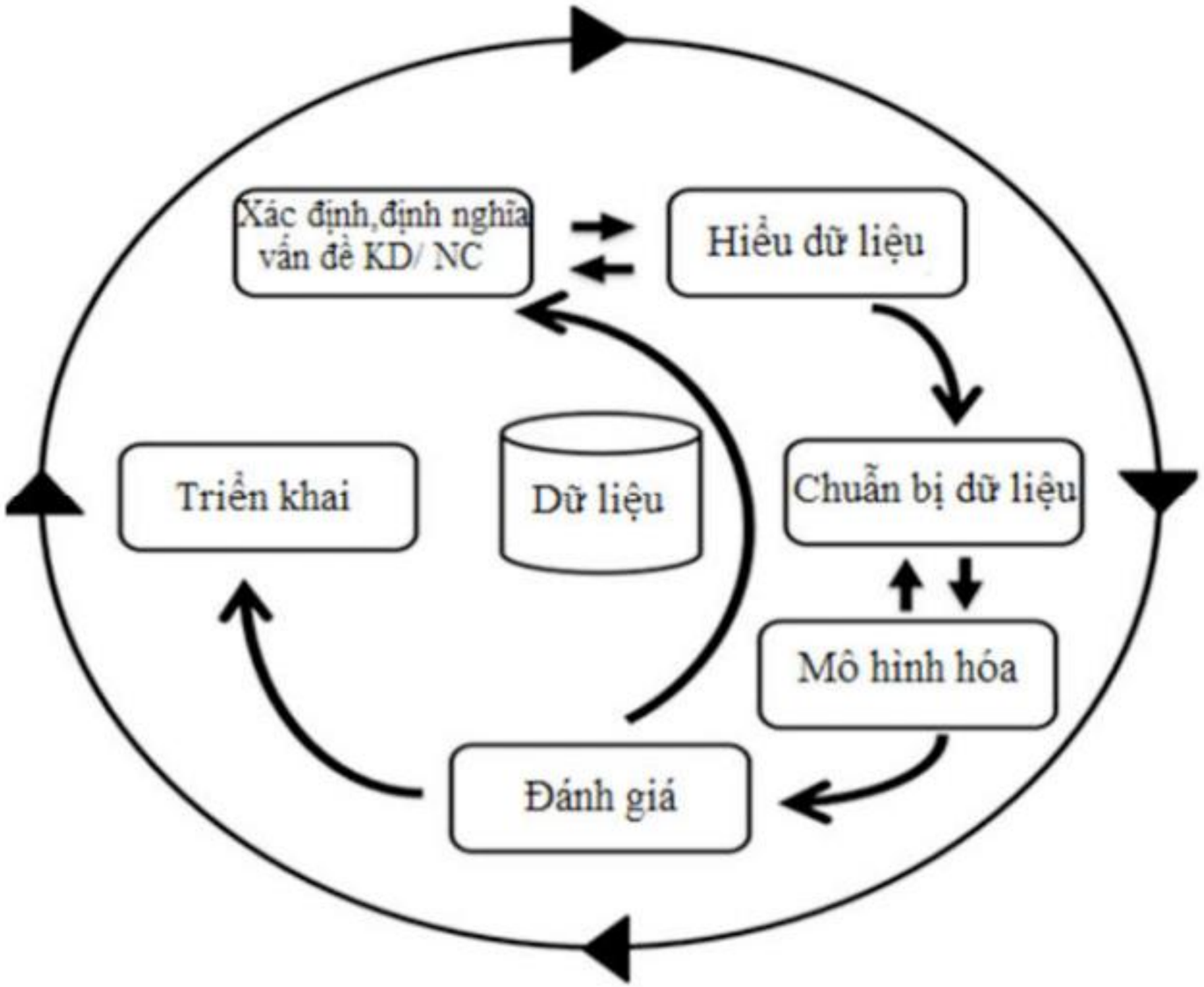
```
from xgboost import XGBRegressor

#    XGBRegressor cho bài toán hồi quy

xgb_model = XGBRegressor(n_estimators=100) xgb_model.fit(X_train, y_train)

#    Dự đoán

y_pred_xgb = xgb_model.predict(X_test)

print("Dự đoán bằng XGBoost:", y_pred_xgb)
```

Sự khác biệt khi sử dụng các thuật toán

Thuật toán	Mức độ phức tạp dữ liệu	Độ chính xác	Tính phi tuyến
Linear Regression	Dữ liệu tuyến tính	Trung bình	Không hỗ trợ
Random Forest	Dữ liệu phi tuyến	Cao	Có hỗ trợ
Logistic Regression	Bài toán phân loại	Trung bình	Không hỗ trợ
Gradient Boosting (XGB)	Bài toán phức tạp	Rất cao	Hỗ trợ tốt

The end.

Thank You!



Hình 1.4: Quá trình mô hình hóa dự báo kinh tế - tài chính [96]