Báo cáo Hệ Hỗ Trợ Quyết Định: Dự đoán giá cổ phiếu Axis Bank

Sử dụng Hồi quy Tuyến tính

Nguyễn Văn Dương

GVHD: TS. Trần Ngọc Thăng
Khoa Toán Tin

Ngày 21 tháng 4 năm 2025

Mục lục

1. Phát biểu bài toán
2. Tiền xử lý dữ liệu
3. Tạo, luyện, đánh giá mô hình
4. Ứng dụng mô hình
5. Kết luận

Phát biểu bài toán

Mô tả bài toán

Sử dụng mô hình hồi quy để dự đoán giá đóng cửa (Close) của cổ phiếu Axis Bank dựa trên dữ liệu lịch sử.

- **Đầu vào**: Dữ liệu lịch sử giá cổ phiếu từ bộ dữ liệu AXISBANK.csv.
- **Đầu ra**: Giá đóng cửa (Close) của cổ phiếu Axis Bank.
- **Yêu cầu xử lý**: Sử dụng mô hình hồi quy tuyến tính (Linear Regression) để dự đoán giá đóng cửa dựa trên các đặc trưng đầu vào.

Thu thập dữ liệu và Thông tin cơ bản

Thu thập dữ liệu

- **Nguồn**: Kaggle - https://www.kaggle.com/datasets/pritsheta/axis-bank/data
- **Tệp**: AXISBANK.csv
- **Cách thu thập**: Tải về và upload lên Google Colab (‘files.upload()‘).

Code: Tải và đọc dữ liệu

```
#Tải file dữ liệu (chạy trong Colab)
from google.colab import files
# uploaded = files.upload() # Commented out for LaTeX compilation

#Import các thư viện
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

#Đọc dữ liệu từ file AXISBANK.csv
df = pd.read_csv('AXISBANK.csv')
```

Thông tin cơ bản (Output)

Output (Tóm tắt)

- Số dòng và cột: (5306, 15)
- Tên các cột: 'Date', 'Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last', 'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume', '%Deliverble'
- Ngày đầu tiên: 2000-01-03
- Ngày cuối cùng: 2021-04-30

(Output chi tiết của df.head() quá dài cho slide, xem file gốc nếu cần)

Đánh nhãn và Thống kê dữ liệu

Xác định nhãn và đặc trưng

```
#Xác định nhãn (label) và đặc trưng (features)
X = df[['Open', 'High', 'Low', 'Volume']]  # Đặc trưng
y = df['Close']  # Nhãn (biến mục tiêu)
print("\nĐặc trưng (X):", X.columns.tolist())
print("Nhãn (y): 'Close'")
```

Output:

- Đặc trưng (X): ['Open', 'High', 'Low', 'Volume']
- Nhãn (y): 'Close'

Thống kê dữ liệu (Tóm tắt)

(Output chi tiết của df.describe() quá dài cho slide)

```
#Thống kê dữ liệu
print("\nThống kê dữ liệu:")
# print(df.describe()) # Output is very large
```

Ví dụ một vài thống kê:

- Close (mean): 585.89

Tiền xử lý: Giá trị thiếu và Chuẩn hóa

Kiểm tra và xử lý giá trị thiếu

```
#Kiểm tra và xử lý giá trị thiếu
print("\nGiá trị thiếu (trước):\n", df.isnull().sum())
df = df.interpolate()  # Điền giá trị thiếu bằng nội suy
print("\nSau khi điền giá trị thiếu:\n", df.isnull().sum())
```

Tóm tắt Output:

- Trước: Thiếu giá trị ở ‘Trades‘, ‘Deliverable Volume‘, ‘
- Sau: ‘interpolate()‘ xử lý được các cột số (‘Deliverable Volume‘, ‘

Chuẩn hóa dữ liệu (MinMaxScaler)

```
#Chuẩn hóa dữ liệu
scaler = MinMaxScaler()
df[['Open', 'High', 'Low', 'Volume']] = scaler.fit_transform(
    df[['Open', 'High', 'Low', 'Volume']])
print("\nDữ liệu sau khi chuẩn hóa (5 dòng đầu tiên):\n",
    df[['Open', 'High', 'Low', 'Volume']].head())
```

(Output của head() được hiển thị)

Tiền xử lý: Feature Engineering và Chia dữ liệu

Tạo đặc trưng mới: Moving Average 5 ngày (MA5)

```
#Tạo đặc trưng mới (ví dụ: Moving Average 5 ngày)
#Tính Moving Average 5 ngày
df['MA5'] = df['Close'].rolling(window=5).mean()
#Điền giá trị thiếu của MA5 bằng trung bình (do rolling tạo NaN ở đầu)
df['MA5'] = df['MA5'].fillna(df['MA5'].mean())
#Thêm đặc trưng MA5 vào X
X = df[['Open', 'High', 'Low', 'Volume', 'MA5']] # Update X
print("\nĐặc trưng sau khi thêm MA5:\n", X.head())
```

```
Đặc trưng sau khi thêm MA5:
       Open      High       Low    Volume         MA5
0  0.002831  0.001486  0.002876  0.000906  585.893931 # Filled with mean
1  0.002980  0.002476  0.002776  0.001922  585.893931 # Filled with mean
2  0.002483  0.002006  0.002271  0.001388  585.893931 # Filled with mean
3  0.002384  0.001634  0.002422  0.000823  585.893931 # Filled with mean
4  0.001987  0.001139  0.001640  0.000496   26.120000 # First actual MA5
```

Chia dữ liệu (Train/Test Split - Không xáo trộn)

```
#Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False) # Important for time series!
print("\nSố dòng tập huấn luyện:", len(X_train))
print("Số dòng tập kiểm tra:", len(X_test))
```

Chia dữ liệu (Output)

Output

```
Số dòng tập huấn luyện: 4244
Số dòng tập kiểm tra: 1062
Ngày bắt đầu tập huấn luyện: 2000-01-03
Ngày kết thúc tập huấn luyện: 2017-01-12
Ngày bắt đầu tập kiểm tra: 2017-01-13
Ngày kết thúc tập kiểm tra: 2021-04-30
```

Huấn luyện và Đánh giá mô hình Hồi quy Tuyến tính

Huấn luyện mô hình

```
#Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)
```

Đánh giá mô hình trên tập Test

```
#Đánh giá mô hình
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f"\nMSE: {mse:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.2f}")
```

Kết quả đánh giá

```
MSE: 24.40
RMSE: 4.94
R²: 1.00
```

Huấn luyện và Đánh giá mô hình Hồi quy Tuyến tính

Huấn luyện mô hình

```
#Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)
```

Đánh giá mô hình trên tập Test

```
#Đánh giá mô hình
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f"\nMSE: {mse:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.2f}")
```

Kết quả đánh giá

```
MSE: 24.40
RMSE: 4.94
R²: 1.00
```

Ứng dụng: Dự đoán 10 ngày cuối và So sánh

Dự đoán và Tạo bảng kết quả

```
#Ứng dụng mô hình (dự đoán 10 ngày cuối)
last_10_days_X = X_test.tail(10)
last_10_actual_y = y_test.tail(10)
last_10_pred_y = model.predict(last_10_days_X)
results = pd.DataFrame({
'Ngày': df['Date'].tail(10).values, # Get values to avoid index issues
'Giá thực tế': last_10_actual_y.values,
'Giá dự đoán': last_10_pred_y,
'Sai số': np.abs(last_10_actual_y.values - last_10_pred_y)})
# Ensure 'Ngày' column is used for plotting later if needed outside this frame
results_dates = results['Ngày']
print("\nKết quả dự đoán 10 ngày cuối:\n", results)
```

Kết quả dự đoán (Bảng)

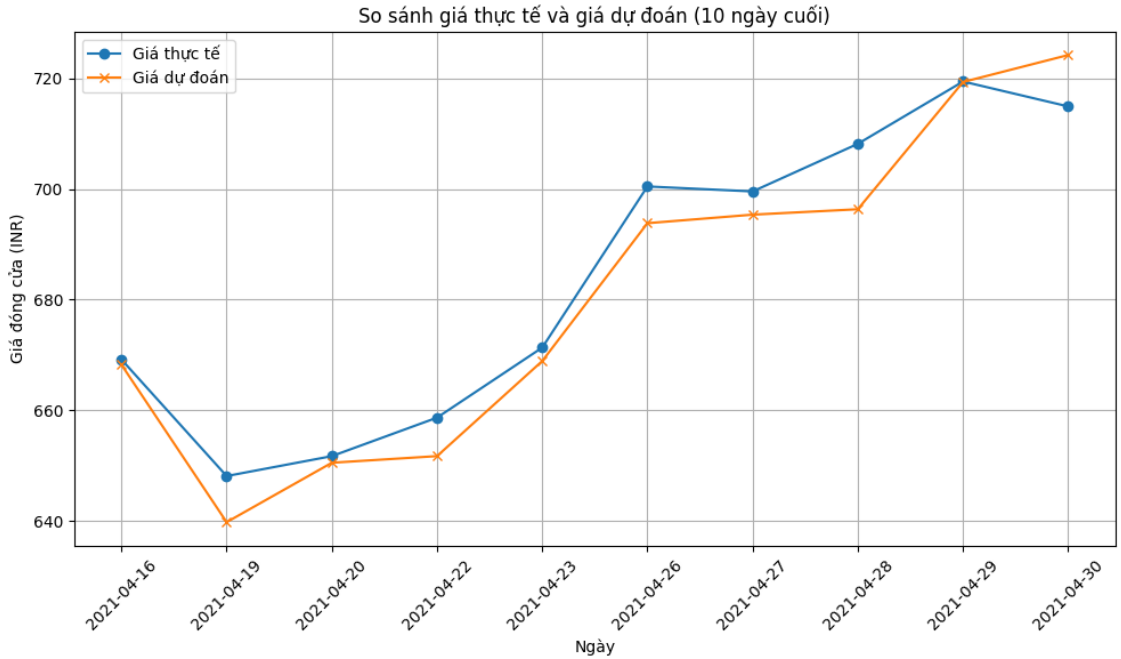
Kết quả dự đoán 10 ngày cuối:

	Ngày	Giá thực tế	Giá dự đoán	Sai số
0	2021-04-16	669.20	668.264058	0.935942
1	2021-04-19	648.15	639.823440	8.326560
2	2021-04-20	651.75	650.554805	1.195195
3	2021-04-22	658.70	651.751802	6.948198
4	2021-04-23	671.35	668.922369	2.427631
5	2021-04-26	700.45	693.817265	6.632735
6	2021-04-27	699.55	695.356392	4.193608
7	2021-04-28	708.15	696.333940	11.816060
8	2021-04-29	719.40	719.313362	0.086638
9	2021-04-30	714.90	724.188224	9.288224

Ứng dụng: Biểu đồ so sánh

Code vẽ biểu đồ

```
# Code to generate the plot (assuming 'results' DataFrame exists)
plt.figure(figsize=(10, 5)) # Adjusted size for slide
plt.plot(results['Ngày'], results['Giá thực tế'], label='Giá thực tế', marker='o')
plt.plot(results['Ngày'], results['Giá dự đoán'], label='Giá dự đoán', marker='x')
plt.xlabel('Ngày')
plt.ylabel('Giá đóng cửa (INR)')
plt.title('So sánh giá thực tế và giá dự đoán (10 ngày cuối)')
plt.legend()
plt.xticks(rotation=45, ha='right') # Adjusted rotation and alignment
plt.grid(True) # Changed from plt.grid()
plt.tight_layout()
plt.savefig('price_comparison.png')  # Lưu biểu đồ
# plt.show() # Not needed if just saving
```



Kết luận

Ưu điểm

- Sử dụng hồi quy tuyến tính đơn giản, dễ triển khai.
- Thêm đặc trưng MA5 có thể giúp nắm bắt xu hướng ngắn hạn.
- Cho kết quả R^2 rất cao trên tập test (cần xem xét kỹ lưỡng).

Nhược điểm

- Mô hình tuyến tính có thể không nắm bắt được các mối quan hệ phi tuyến phức tạp.
- $R^2 = 1.00$ là dấu hiệu của việc dự đoán giá Close dựa trên Open, High, Low cùng ngày (quá dễ dàng), không phải là dự đoán giá tương lai thực sự hiệu quả.
- Chỉ sử dụng dữ liệu giá cơ bản, bỏ qua các yếu tố bên ngoài (tin tức, chỉ số thị trường).
- Sai số có thể lớn vào những ngày biến động mạnh.