Báo cáo Bài tập lớn Các vấn đề hiện trong Khoa học máy tính

1. Bài toán

Thiết kế model dự đoán tình hình tài chính của doanh nghiệp dựa vào báo cáo tài chính của các năm trước đó. Đầu vào là lịch sử tài chính của 5 công ty Việt Nam: FPT, Hòa Phát, MB, VNM, và HAGL, đưa ra biểu đồ tài chính dự đoán tương lai.

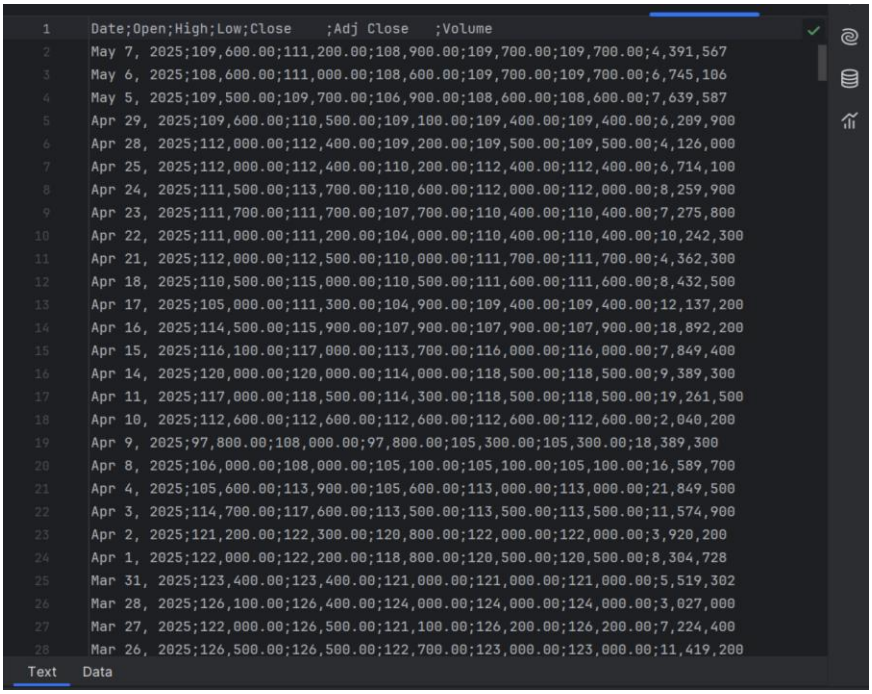
Từ bài toán này nhóm đưa ra ý tưởng áp dụng Transformer Encoder cho bài toán dự báo chuỗi thời gian tài chính, với quy trình đầy đủ từ tiền xử lý dữ liệu, huấn luyện mô hình, đến dự báo và đánh giá.

Link github: https://github.com/TrungHieu-alt/stockForecast.git

2. Xử lí dữ liệu

Tổng hợp dữ liệu về các bảng. Ở đây nhóm em đã thu thập dữ liệu từ sàn chứng khoán, trong đó:

- Ngày (Date): Ngày giao dịch cổ phiếu.
- Mở cửa (Open): Giá cổ phiếu khi thị trường mở cửa trong ngày giao dịch đó.
- Cao nhất (High): Giá cổ phiếu cao nhất trong ngày giao dịch.
- Thấp nhất (Low): Giá cổ phiếu thấp nhất trong ngày giao dịch.
- Đóng cửa (Close): Giá cổ phiếu khi thị trường đóng cửa trong ngày giao dịch.
- Giá điều chỉnh (Adj Close): Giá đóng cửa đã được điều chỉnh tính đến các yếu tố như chia cổ tức hoặc tách cổ phiếu.
- Khối lượng (Volume): Số lượng cổ phiếu đã được giao dịch trong ngày đó.



```
Date;Open;High;Low;Close    ;Adj Close   ;Volume
May 7, 2025;109,600.00;111,200.00;108,900.00;109,700.00;109,700.00;4,391,567
May 6, 2025;108,600.00;111,000.00;108,600.00;109,700.00;109,700.00;6,745,106
May 5, 2025;109,500.00;109,700.00;106,900.00;108,600.00;108,600.00;7,639,587
Apr 29, 2025;109,600.00;110,500.00;109,100.00;109,400.00;109,400.00;6,209,900
Apr 28, 2025;112,000.00;112,400.00;109,200.00;109,500.00;109,500.00;4,126,000
Apr 25, 2025;112,000.00;112,400.00;110,200.00;112,400.00;112,400.00;6,714,100
Apr 24, 2025;111,500.00;113,700.00;110,600.00;112,000.00;112,000.00;8,259,900
Apr 23, 2025;111,700.00;111,700.00;107,700.00;110,400.00;110,400.00;7,275,800
Apr 22, 2025;111,000.00;111,200.00;104,000.00;110,400.00;110,400.00;10,242,300
Apr 21, 2025;112,000.00;112,500.00;110,000.00;111,700.00;111,700.00;4,362,300
Apr 18, 2025;110,500.00;115,000.00;110,500.00;111,600.00;111,600.00;8,432,500
Apr 17, 2025;105,000.00;111,300.00;104,900.00;109,400.00;109,400.00;12,137,200
Apr 16, 2025;114,500.00;115,900.00;107,900.00;107,900.00;107,900.00;18,892,200
Apr 15, 2025;116,100.00;117,000.00;113,700.00;116,000.00;116,000.00;7,849,400
Apr 14, 2025;120,000.00;120,000.00;114,000.00;118,500.00;118,500.00;9,389,300
Apr 11, 2025;117,000.00;118,500.00;114,300.00;118,500.00;118,500.00;19,261,500
Apr 10, 2025;112,600.00;112,600.00;112,600.00;112,600.00;112,600.00;2,040,200
Apr 9, 2025;97,800.00;108,000.00;97,800.00;105,300.00;105,300.00;18,389,300
Apr 8, 2025;106,000.00;108,000.00;105,100.00;105,100.00;105,100.00;16,589,700
Apr 4, 2025;105,600.00;113,900.00;105,600.00;113,000.00;113,000.00;21,849,500
Apr 3, 2025;114,700.00;117,600.00;113,500.00;113,500.00;113,500.00;11,574,900
Apr 2, 2025;121,200.00;122,300.00;120,800.00;122,000.00;122,000.00;3,920,200
Apr 1, 2025;122,000.00;122,200.00;118,800.00;120,500.00;120,500.00;8,304,728
Mar 31, 2025;123,400.00;123,400.00;121,000.00;121,000.00;121,000.00;5,519,302
Mar 28, 2025;126,100.00;126,400.00;124,000.00;124,000.00;124,000.00;3,027,000
Mar 27, 2025;122,000.00;126,500.00;121,100.00;126,200.00;126,200.00;7,224,400
Mar 26, 2025;126,500.00;126,500.00;122,700.00;123,000.00;123,000.00;11,419,200
Text  Data
```

Đây là dữ liệu thô, cần được xử lí như xóa dấu cách, format lại ngày để lại bỏ nhiễu.

```python
def robust_parse_date(date_str):  1 usage  TrungHieu-alt
    """Parse messy date strings by collapsing spaces and coercing to datetime."""
    if pd.isna(date_str):
        return pd.NaT
    # Collapse multiple spaces and strip
    s = date_space_pattern.sub( repl: " ", str(date_str)).strip()
    try:
        # Try default parsing
        return pd.to_datetime(s, errors="raise")
    except Exception:
        # Fallback to coercion
        return pd.to_datetime(s, dayfirst=False, yearfirst=False, errors="coerce")
```

Chuyển chuỗi ngày tháng sang kiểu datetime, tránh các định dạng sai, gây nhiễu.

```python
def clean_stock_csv(input_file, output_file):  1 usage  TrungHieu-alt
    if 'Date' not in df.columns:
        raise KeyError(f"Column 'Date' not found in {input_file}")
    df['Date'] = df['Date'].apply(robust_parse_date)

    # Check date parsing ratio
    na_dates = df['Date'].isna().mean()
    if na_dates > 0.1:
        print(f"⚠️ Warning: {os.path.basename(input_file)} has {na_dates*100:.1f}% unparsed dates.")

    # Clean numeric columns: remove commas and spaces
    num_cols = [c for c in ['Open','High','Low','Close','Adj Close','Volume'] if c in df.columns]
    for col in num_cols:
        # Remove thousand separators and stray spaces
        df[col] = df[col].astype(str).str.replace(',', '', regex=False).str.replace(' ', '', regex=False)
        # Convert to numeric, coerce errors
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Drop rows missing critical data
    df = df.dropna(subset=['Date', 'Close']).reset_index(drop=True)
    # Sort by date ascending
    df = df.sort_values('Date').reset_index(drop=True)

    # Save cleaned CSV
    os.makedirs(os.path.dirname(output_file), exist_ok=True)
    df.to_csv(output_file, index=False, encoding='utf-8')
    print(f"✅ Cleaned & saved: {output_file} (Rows: {len(df)})")
    return df
```

Loại bỏ các cột thiếu dữ liệu và sắp xếp lại thứ tự ngày tháng.

```python
def normalize_and_split(df, out_dir, col='Close', train_ratio=0.8, val_ratio=0.1):  1 usage  TrungHieu-alt
    # Extract series and scale
    values = df[col].values.reshape(-1, 1)
    scaler = MinMaxScaler()
    series = scaler.fit_transform(values).flatten()

    # Compute split indices
    n = len(series)
    train_end = int(n * train_ratio)
    val_end = int(n * (train_ratio + val_ratio))

    train, val, test = series[:train_end], series[train_end:val_end], series[val_end:]

    # Save splits
    os.makedirs(out_dir, exist_ok=True)
    np.save(os.path.join(out_dir, 'train.npy'), train)
    np.save(os.path.join(out_dir, 'val.npy'), val)
    np.save(os.path.join(out_dir, 'test.npy'), test)
    print(f"✅ Saved splits → {out_dir} (train:{len(train)}, val:{len(val)}, test:{len(test)})")
```

Sử dụng `MinMaxScaler` để chuẩn hóa giá đóng cửa về khoảng [0,1]. Phân chia dữ liệu đã xử lí thành 3 tập: tập huấn luyện 80%, tập validation 10%, tập test10%

3. Thiết kế model

Để giải quyết bài toán này, nhóm em đã phát triển mô hình transformer để đưa ra dự đoán tài chính doanh nghiệp từ chuỗi giá cổ phiếu. Mô hình này dự báo giá Close cho ngày kế tiếp từ một.

Kiến trúc mô hình:

- Embedding Layer: Chuyển đổi giá đóng cửa (1 chiều) thành không gian `d_model` (mặc định 32 chiều).
- Positional Encoding: Thêm thông tin vị trí vào chuỗi thời gian.
- Transformer Encoder: Gồm 1 tầng (`num_layers=1`) với 2 đầu chú ý (`nhead=2`).
- Linear Layer: Xuất dự đoán giá cho ngày tiếp theo từ đầu ra của bước cuối cùng trong chuỗi.

Quy trình train model:

```
def parse_args():  1 usage   TrungHieu-alt
    parser = argparse.ArgumentParser(description="Train Transformer model on preprocessed stock data")
    parser.add_argument( *name_or_flags: '--ticker', type=str, required=True, help='Ticker name matching split_<
    parser.add_argument( *name_or_flags: '--seq_len', type=int, default=60, help='Input sequence length')
    parser.add_argument( *name_or_flags: '--pred_len', type=int, default=1, help='Output prediction length')  #
    parser.add_argument( *name_or_flags: '--batch_size', type=int, default=16, help='Batch size')
    parser.add_argument( *name_or_flags: '--epochs', type=int, default=100, help='Number of epochs')  # Tăng epo
    parser.add_argument( *name_or_flags: '--lr', type=float, default=1e-4, help='Learning rate')
    parser.add_argument( *name_or_flags: '--weight_decay', type=float, default=1e-4, help='Weight decay for opti
    parser.add_argument( *name_or_flags: '--patience', type=int, default=7, help='Early stopping patience on val
    parser.add_argument( *name_or_flags: '--clean_dir', type=str, default='cleanDataset', help='Directory contai
    parser.add_argument( *name_or_flags: '--checkpoint_dir', type=str, default='checkpoints', help='Where to sav
    return parser.parse_args()
```

Các tham số để huấn luyện mô hình:

--ticker	Mã cổ phiếu cần huấn luyện
--seq_len	Số ngày trong chuỗi đầu vào
--pred_len	Số bước cần dự báo (ở đây là 1-step ahead)
--batch_size	Số mẫu mỗi batch
--epochs	Số epoch tối đa
--lr	Learning rate
--weight_decay	Weight decay cho Adam
--patience	Dừng sớm nếu không cải thiện
--clean_dir	Thư mục chứa train/val/test
--checkpoint_dir	Nơi lưu mô hình đã huấn luyện

```python
class TimeSeriesDataset(Dataset):  2 usages  TrungHieu-alt
    def __init__(self, series, seq_len, pred_len):  TrungHieu-alt
        self.series = series
        self.seq_len = seq_len
        self.pred_len = pred_len
        self.samples = []
        n = len(series)
        for i in range(n - seq_len - pred_len + 1):
            x = series[i:i+seq_len]
            y = series[i+seq_len:i+seq_len+pred_len]
            self.samples.append((x, y))

    def __len__(self):  TrungHieu-alt
        return len(self.samples)

    def __getitem__(self, idx):  TrungHieu-alt
        x, y = self.samples[idx]
        return torch.FloatTensor(x), torch.FloatTensor(y)
```

Biến chuỗi giá trị 1 chiều thành nhiều cặp (x, y) để học, giúp mô hình học cách dự đoán giá tương lai từ chuỗi quá khứ.

```python
class PositionalEncoding(nn.Module):  1 usage  TrungHieu-alt
    def __init__(self, d_model, max_len=5000):  TrungHieu-alt
        super().__init__()
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len, dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-np.log(10000.0) / d_model))
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        self.register_buffer( name: 'pe', pe.unsqueeze(0))

    def forward(self, x):  # x shape: (batch, seq_len, d_model)  TrungHieu-alt
        x = x + self.pe[:, :x.size(1)]
        return x
```

Do mô hình Transformer không thể phân biệt thời gian nên Positional Encoding layer biến thông tin về vị trí (tức là ngày thứ mấy trong chuỗi) thành vector đặc trưng mà mô hình có thể sử dụng.

```python
class TransformerEncoderModel(nn.Module):  1 usage  TrungHieu-alt
    def __init__(self, d_model=32, nhead=2, num_layers=1, seq_len=60, pred_len=1, dropout=0.1):  TrungHieu
        super().__init__()
        self.seq_len = seq_len
        self.pred_len = pred_len
        self.embed = nn.Linear( in_features: 1, d_model)
        self.pos_enc = PositionalEncoding(d_model)
        encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_model, nhead=nhead, dropout=dropout, batch_first=True
        )
        self.transformer = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)
        self.fc = nn.Linear(d_model,  out_features: 1)  # Output layer tối ưu, chỉ lấy 1 giá trị

    def forward(self, x):  TrungHieu-alt
        x = x.unsqueeze(-1)  # (batch, seq_len, 1)
        x = self.embed(x)    # (batch, seq_len, d_model)
        x = self.pos_enc(x)  # Add positional encoding
        x = self.transformer(x)  # (batch, seq_len, d_model)
        x = x[:, -1, :]  # Lấy bước cuối cùng
        out = self.fc(x)  # (batch, 1)
        return out
```

Lớp TransformerEncoder gồm các tầng self-attention và feed-forward nhằm ứng dụng Attention vào mô hình giúp mô hình học được mối quan hệ giữa các ngày trong chuỗi.

Sau khi chạy qua các tầng Transformer, lấy bước cuối của chuỗi đưa vào fully connected vả đưa ra giá dự đoán (Linear layer).

4. Sử dụng mô hình dự đoán bài toán

Áp dụng mô hình đã xây dựng ở trên để dự đoán kết quả cho giá cổ phiếu trong 1000 ngày tới của 5 doanh nghiệp. Dự báo được thực hiện theo phương pháp đệ quy, dùng đầu ra làm đầu vào cho bước tiếp theo.

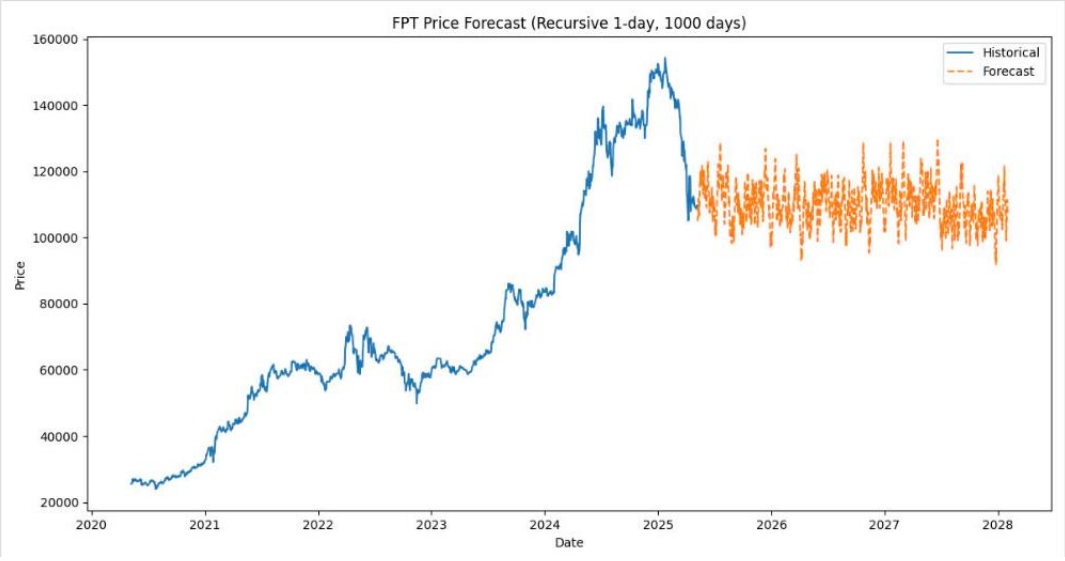
```
# Recursive daily forecast
window = scaled[-args.seq_len:]
preds = []
for i in range(args.forecast_days):
    x = torch.FloatTensor(window[-args.seq_len:]).unsqueeze(0).to(device)
    with torch.no_grad():
        y = model(x).cpu().numpy().flatten()
    next_val = y[0]
    preds.append(next_val)
    window.append(next_val)

forecast_prices = scaler.inverse_transform(np.array(preds).reshape(-1, 1)).flatten()

# Thêm nhiễu ngẫu nhiên để mô phỏng biến động (tăng độ nhiễu cho dự báo dài hạn)
forecast_prices += np.random.normal( loc: 0, 0.02 * forecast_prices.std(), forecast_prices.shape)

# Kiểm soát xu hướng: Giới hạn thay đổi hàng ngày dựa trên biến động lịch sử
for i in range(1, len(forecast_prices)):
    change = forecast_prices[i] - forecast_prices[i-1]
    if abs(change) > max_change:
```

Thêm nhiễu ngẫu nhiên (dựa trên độ lệch chuẩn của dữ liệu) để mô phỏng biến động thị trường.

```
    # Tính độ biến động lịch sử để kiểm soát xu hướng
    historical_changes = np.diff(prices.flatten())
    change_std = np.std(historical_changes)
    max_change = 3 * change_std  # Giới hạn thay đổi tối đa (3 sigma)
```

Dùng để giới hạn biến động tối đa của giá dự báo để phòng mô hình tạo ra giá nhảy quá mạnh, thiếu thực tế.

Lưu dự báo dưới dạng file CSV trong thư mục `forecastResults`, in ra biểu đồ so sánh giá lịch sử và giá dự báo.



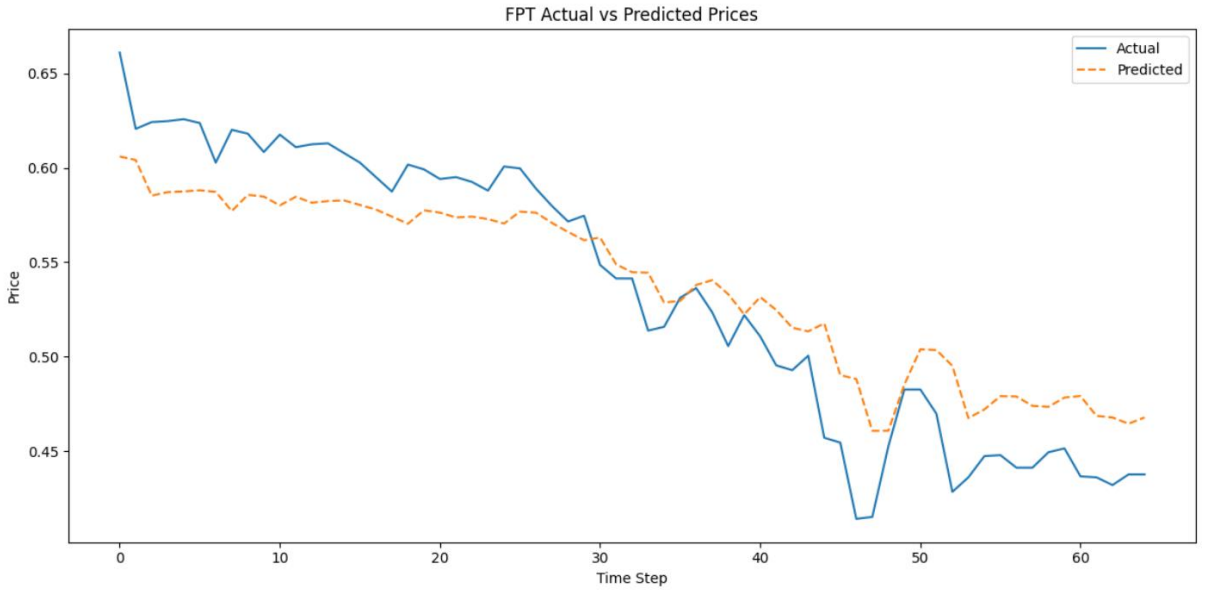
5. Đánh giá model

Đánh giá hiệu suất của mô hình theo phương pháp dự báo một bước trên tập test và so sánh với giá trị thực tế. Sử dụng cách chỉ số đánh giá Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE).

```python
# Plot Actual vs Predicted
plt.figure(figsize=(12, 6))
plt.plot( *args: targets_inv, label='Actual')
plt.plot( *args: predictions_inv, label='Predicted', linestyle='--')
plt.title(f"{args.ticker} Actual vs Predicted Prices")
plt.xlabel('Time Step')
plt.ylabel('Price')
plt.legend()
plt.tight_layout()
plot_path = os.path.join(args.output_dir, f"{args.ticker}_actual_vs_predicted.png")
plt.savefig(plot_path)
plt.close()

print(f"✅ Metrics and plot saved in {args.output_dir}")
```

161:59 CRLF UTF-8 4 spaces Python 3.12

Biểu diễn dưới dạng biểu đồ để so sánh kết quả thực tế và kết quả dự đoán.



Giá trị các chỉ số đánh giá model áp dụng trên tài chính của FPT.

```
Metric,Value
MSE,0.000891012707013897
RMSE,0.029849835962931136
MAE,0.025905503535817283
```

6. Tổng kết

Nhìn chung, mô hình trên hoạt động tốt trên tập dữ liệu đã chuẩn hóa, học được các xu hướng trung hạn đến dài hạn của thị trường chứng khoán, biểu đồ dự đoán đưa ra bám sát thực tế, chỉ số RMSE thấp. Tuy nhiên, giá đóng cửa được dự đoán thì chưa thực sự chính xác so với thực tế.

Bảng phân chia công việc:

Thành viên	Nguyễn Trung Hiếu	Lê Đình Nghĩa	Nguyễn Trần Phương Hà
Công việc	- Tổng hợp data - Thiết kế model - Train model	- Đánh giá model - Thiết kế model	- Xử lí dữ liệu - Làm báo cáo
Phần trăm(%)	40%	30%	30%