**Mô hình Long Short-Term Memory**

**Khái quát**

1. Dữ liệu Time Series

* Chuỗi thời gian (Time Series): là tập hợp dữ liệu được thu thập hoặc ghi nhận theo thứ tự thời gian, thường với các mốc thời gian đều đặn.

A screenshot of a graph

Description automatically generated

*Dữ liệu về nhiệt độ trung bình năm*

A screenshot of a graph

Description automatically generated

*Dữ liệu bán hàng theo tháng và phân khúc khách hàng*

* Đặc điểm của Time Series:
* Trend (xu hướng): Xu hướng trong dữ liệu chuỗi thời gian đề cập đến sự chuyển động lên hoặc xuống của dữ liệu trong một khoảng thời gian dài. Bạn có thể xác định xu hướng chung dựa vào độ dốc của dữ liệu trên biểu đồ.
* Seasonality (tính mùa vụ): Tính mùa vụ trong dữ liệu chuỗi thời gian đề cập đến những biến động tăng hoặc giảm lặp đi lặp lại một cách đều đặn của dữ liệu trong một khoảng thời gian.
* Cyclicity (tính chu kỳ): Tính chu kỳ trong dữ liệu chuỗi thời gian đề cập đến những biến động lên xuống lặp lại, hoặc những thay đổi định kỳ, có thể kéo dài trong nhiều năm và chuyển từ giai đoạn này qua giai đoạn khác.
* Irregularity (tính bất thường): Sự bất thường trong dữ liệu chuỗi thời gian đề cập đến những sự thay đổi bất thường của dữ liệu, xảy ra một cách ngẫu nhiên, có thể trái ngược hoàn toàn với các dữ liệu trong quá khứ, khó có thể giải thích và không dự đoán được trước.

1. Phương pháp xây dựng Dữ liệu Time Series

* Thu thập dữ liệu:
* Xác định nguồn dữ liệu: chọn nguồn dữ liệu phù hợp như thu thập từ các ứng dụng và trang web.
* Tần suất thu thập dữ liệu: quyết định khoảng thời gian giữa các lần thu thập (giờ, ngày, tuần, tháng …).
* Tiền xử lý dữ liệu:
* Làm sạch dữ liệu: loại bỏ giá trị thiếu và xử lý giá trị không phù hợp
* Đồ bộ hóa mốc thời gian: xử lý các mốc thời gian không liên tục để đảm bảo dữ liệu có cùng khoảng cách thời gian giữa các đặc điểm hơn.
* Chuẩn hóa và biến đổi: áp dụng các phép biến đổi cần thiết  để dữ liệu trở nên ổn định và dễ phân tích hơn.
* Xây dựng các đặc trưng (Feature Engineering):
* Đặc trưng thời gian: các thuộc tính được trích xuất từ dữ liệu thời gian gốc. Đặc trưng này có thể giúp mô hình hiểu rõ hơn về các xu hướng và mẫu hình trong dữ liệu thời gian. Thêm các cột như ngày trong tuần, tháng …
* Độ trễ (Lag Features): tạo các cột thể hiện giá trị của dữ liệu tại các mốc thời gian trước.
* Trung bình động (Moving Average): tính giá trị trung bình trên một khoảng thời gian nhất định.
* Xu hướng và mùa vụ: xác định xu hướng và các mẫu hình định kỳ
* Áp dụng kỹ thuật mô hình hóa:
* Dự báo: sử dụng ARIMA, SARIMA, hoặc các mạng deep learning như GRUs, LSTMs.

1. Định nghĩa LSTM

Long short-term memory (LSTM) là một mạng thần kinh hồi quy (RNN) nhân tạo được sử dụng trong lĩnh vực học sâu. Không giống như các mạng thần kinh truyền thẳng (FNN) tiêu chuẩn, LSTM có chứa các kết nối phản hồi. Mạng không chỉ xử lý các điểm dữ liệu đơn lẻ (như các hình ảnh), mà còn xử lý toàn bộ chuỗi dữ liệu (chẳng hạn như lời nói hoặc video).

A diagram of a flowchart

Description automatically generated

*Hình minh họa cơ chế hoạt động của mô hình LSTM*

1. Phương pháp triển khai model LSTM

* Đánh giá và chọn thư viện phù hợp: chọn Tensorflow, PyTorch, MXNet hoặc là FastAI để phù hợp với dữ liệu Time Series và mô hình LSTM.

→ Chọn đúng thư viện không chỉ giúp tăng hiệu suất mà còn làm cho quá trình phát triển và triển khai mô hình trở nên dễ dàng hơn. Việc xem xét các yếu tố như hiệu suất, dễ sử dụng, hỗ trợ cộng đồng, khả năng mở rộng, tính linh hoạt, yêu cầu cụ thể của dự án, và khả năng bảo trì của mô hình.

* Chuẩn bị dữ liệu:
* Dữ liệu đầu vào: dữ liệu input phải ở dạng mà mô hình LSTM có thể hiểu được. Điều này bao gồm việc chuẩn hóa dữ liệu và chia nhỏ dữ liệu thành các sequences.
* Chia dữ liệu: chia dữ liệu thành các chuỗi thời gian nhỏ hơn và chia thành tập huấn luyện và tập kiểm tra nhằm giúp mô hình học từ các mẫu trước đó và dự đoán tốt hơn.
* Xây dựng mô hình:
* Xác định đặc trưng và nhãn**:** tìm ra các đặc điểm chung của dữ liệu - đặc trưng (features) và nhãn (labels) rất quan trọng để mô hình học và đánh giá kết quả.

→ Mục đích mô hình sẽ lấy những đặc trưng để học và dựa trên nhãn để đánh giá kết quả.

* Huấn luyện và kiểm thử mô hình:
* Thiết lập các thành phần:cần xác định và phân tích các thành phần của mô hình như số lượng đơn vị LSTM, số lượng lớp, hàm mất mát và bộ tối ưu.
* Chia tỉ lệ dữ liệu: cần chia tỉ lệ dữ liệu để tránh việc mô hình học thuộc lòng dữ liệu huấn luyện mà không thể tổng quát hóa cho dữ liệu mới.
* Dự đoán kết quả của mô hình:
* Nhập dữ liệu cũ: nhập dữ liệu mới và sử dụng mô hình để dự đoán kết quả.
* Đánh giá kết quả: so sánh kết quả dự đoán của mô hình với dữ liệu thực tế để đánh giá hiệu quả của mô hình.

**3.3.2 Triển khai mô hình**

1. Chọn thư viện:

Sử dụng các thư viện để xây dựng mô hình như sau:

* Thư viện Tensorflow:
* Hỗ trợ phần cứng như GPU và TPU để tăng tốc độ tính toán.
* Xử lý dữ liệu lớn và phức tạp, đặc biệt là trong deep learning.
* Hỗ trợ nhiều loại mô hình khác nhau, từ các mạng nơ-ron đơn giản đến các mô hình phức tạp như CNN, RNN, LSTM và GANs
* Thư viện numpy: là thư viện cơ bản cho việc xử lý mảng và tính toán số học trong Python. Nó cung cấp một đối tượng mảng n-dimensional (ndarray) rất mạnh mẽ và hiệu quả, cùng với các hàm toán học và tuyến tính (linear algebra) để xử lý dữ liệu.
* Thư viện pandas: là một thư viện mạnh mẽ trong Python cho việc thao tác và phân tích dữ liệu, đặc biệt là dữ liệu có cấu trúc như bảng (dataframe). Nó cung cấp các công cụ để xử lý, làm sạch và phân tích dữ liệu.
* Thư viện matplotlib: là thư viện vẽ đồ thị mạnh mẽ trong Python, thường được sử dụng để tạo các biểu đồ, đồ thị và hình ảnh từ dữ liệu. Nó hỗ trợ nhiều loại biểu đồ khác nhau như đường, cột, phân tán, và hình ảnh.
* Thư viện requests: là thư viện Python dễ sử dụng giúp gửi HTTP requests (như GET, POST, PUT, DELETE) để tương tác với các API hoặc dịch vụ web. Dùng để tải dữ liệu từ URL vào model để sử dụng cho việc xây dựng, train, test và dự đoán của model.
* Thư viện BytesIO: là một lớp trong thư viện **io** của Python, cho phép xử lý dữ liệu nhị phân (binary data) như thể nó là một tệp (file).
* Kết quả trên Google Collab:

A screenshot of a computer

Description automatically generated

1. Đánh giá dữ liệu Input

* Ta sử dụng dữ liệu đã qua đánh giá mức độ tương quan, cụ thể ở link sau:<https://drive.google.com/open?id=1_FrEnOMGx93QsDUojgPXUlC_4qNyh6fe&usp=chrome_ntp>  .
* Kiểm tra dữ liệu:

Tên tập dữ liệu: time\_series\_data\_input

Số dòng: 13608

Số cột: 26

Kiểu dữ liệu của mỗi cột:

mssv             object

dtbhk\_1         float64

dtbhk\_2         float64

dtbhk\_3         float64

dtbhk\_4         float64

dtbhk\_5         float64

dtbhk\_6         float64

dtbhk\_7         float64

dtbhk\_8         float64

dtbhk\_9         float64

dtbhk\_10        float64

dtbhk\_11        float64

dtbhk\_12        float64

dtbhk\_13        float64

dtbhk\_14        float64

dtbhk\_15        float64

dtbhk\_16        float64

dtbhk\_17        float64

dtbhk\_18        float64

dtbhk\_19        float64

dtbhk\_20        float64

dtbhk\_21        float64

dtbhk\_22        float64

dtbhk\_23          int64

dtbhk\_24          int64

dtb\_toankhoa    float64

dtype: object

* Phân tích dữ liệu, dự kiến kết quả đầu ra:
* Input: Mã số sinh viên, điểm trung bình qua từng học kỳ [Features]
* Dạng: Tensor
* Mô tả: dạng float64
* Chiều: (13607,25)
* Output: Điểm trung bình toàn khóa của mỗi sinh viên [Label]
* Dạng: Tensor
* Mô tả: dạng float64
* Chiều: (13607,1)
* Kết quả thực hiện trên Google Collab:

A screenshot of a computer

Description automatically generated

* Giải thích code:

B1 - Tách dữ liệu thành Features và Label, dữ liệu đầu vào của ta là một file .xlsx, gồm 26 cột nội dung.

* **df.iloc[:, 1:25].values**: trích xuất các cột từ cột thứ 2 đến cột thứ 25 (chỉ số từ 1 đến 24) từ DataFrame df và chuyển đổi chúng thành một mảng NumPy (.values). Đây sẽ là dữ liệu đầu vào x.
* **df.iloc[:, -1].values**: trích xuất cột cuối cùng của DataFrame df và chuyển đổi nó thành một mảng NumPy. Đây sẽ là biến mục tiêu y.

B2 - Chuẩn hóa dữ liệu:

* **StandardScaler()**: tạo một đối tượng chuẩn hóa dữ liệu. StandardScaler chuẩn hóa các đặc trưng sao cho chúng có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.
* **scaler.fit\_transform(x)**: tính toán các thống kê cần thiết (giá trị trung bình và độ lệch chuẩn) từ dữ liệu x và chuẩn hóa dữ liệu này.

B3 - Thay đổi hình dạng của dữ liệu:

* **x\_scaled.reshape((x\_scaled.shape[0], x\_scaled.shape[1], 1))**: Thay đổi hình dạng của dữ liệu x\_scaled từ dạng (số mẫu, số đặc trưng) thành (số mẫu, số đặc trưng, 1).

B4 - Chia dữ liệu thành tập huấn luyện và tập kiểm tra, sau đó in ra hình dạng của các tập dữ liệu:

* **train\_test\_split(x\_scaled, y, test\_size=0.2, random\_state=42)**: chia dữ liệu thành hai tập, một tập huấn luyện và một tập kiểm tra. 80% dữ liệu sẽ được sử dụng để huấn luyện (x\_train và y\_train), và 20% dữ liệu sẽ được sử dụng để kiểm tra (x\_test và y\_test). random\_state=42 đảm bảo rằng việc chia dữ liệu là tái lập được (nghĩa là mỗi lần chạy lại, kết quả chia sẽ giống nhau).
* Các hàm printf(): đưa ra hình dạng của các tập lệnh.
* Đánh giá kết quả:



* Input: đầu vào sẽ gồm 10886 mẫu sẽ được sử dụng cho việc huấn luyện, tức là 80% của tổng số mẫu 13607. Với 24 đặc trưng của dữ liệu, và 1 kênh - tức là các dữ liệu đặc trưng này đều chung 1 nhóm.
* Output: là một mảng một chiều với 10886 giá trị. Điều này phù hợp với việc mỗi mẫu trong tập huấn luyện có một giá trị mục tiêu duy nhất.

1. Build model

* Kết quả thực hiện trên Google Collab:

A screenshot of a computer program

Description automatically generated

* Giải thích code:
* Khởi tạo mô hình:  sử dụng **Sequential** từ tensorflow

A close-up of words

Description automatically generated

* Xây dựng mô hình:

A white background with black text

Description automatically generated

* Lớp đầu vào **Input(shape=(24,1))**: mỗi mẫu dữ liệu đầu vào là một chuỗi thời gian dài 24 bước, với 1 đặc trưng (feature) tại mỗi bước
* Thêm lớp LSTM đầu tiên **LSTM(128, return\_sequences=True)** : lớp LSTM với 128 đơn vị ẩn giúp mô hình học các thông tin phức tạp từ chuỗi thời gian.
* **return\_sequences=True**: khi được đặt là True, lớp LSTM trả về toàn bộ chuỗi đầu ra (output sequence) thay vì chỉ trạng thái cuối cùng.
* **Dropout(0.2)**: ngẫu nhiên bỏ đi 20% (tỷ lệ 0.2) số nơ-ron trong lớp hiện tại trong quá trình huấn luyện, giúp mô hình không phụ thuộc quá mức vào một số nơ-ron cụ thể.

A group of text with black text

Description automatically generated with medium confidence

* Thêm lớp LSTM thứ 2 **LSTM(64)** với 64 đơn vị ẩn: đây là lớp cuối cùng trong các lớp LSTM và ta chỉ cần đầu ra cuối cùng để chuyển tiếp đến các lớp Dense sau.
* **Dropout(0.2)**: được áp dụng để giảm overfitting, loại bỏ ngẫu nhiên 20% nơ-ron trong lớp hiện tại.

A blue and black text

Description automatically generated

* Lớp **Dense(1)** là lớp kết nối đầy đủ (fully connected layer) với  Với 1 đơn vị đầu ra, lớp này trả về một giá trị duy nhất, phù hợp với bài toán dự đoán chuỗi thời gian.
* Biên dịch mô hình với **compile(optimizer='adam', loss='mse')**
* **optimizer='adam'**: Sử dụng thuật toán tối ưu Adam, phổ biến trong việc huấn luyện mô hình học sâu vì khả năng hội tụ nhanh và hiệu quả.
* **loss=’mse’**: Hàm mất mát Mean Squared Error (MSE) được sử dụng, phù hợp cho các bài toán dự báo giá trị liên tục.
* Đánh giá kết quả:

A screenshot of a computer

Description automatically generated

* Tổng số tham số của mô hình là 116,033.
* Tất cả các tham số đều có thể huấn luyện (trainable), không có tham số không huấn luyện (non-trainable).
* Sử dụng hai lớp LSTM giúp mô hình học được cả các mối quan hệ ngắn hạn và dài hạn trong chuỗi thời gian.
* Có Dropout để giảm thiểu overfitting, đảm bảo mô hình hoạt động ổn định trên dữ liệu mới.

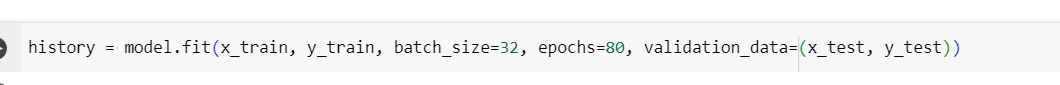
1. Train model

* Kết quả thực hiện trên Google Collab:

A screenshot of a computer

Description automatically generated

* Giải thích code:



* Hàm **fit()**: phương thức được sử dụng để huấn luyện mô hình với dữ liệu đầu vào và nhãn tương ứng.
* **x\_train**: Dữ liệu đầu vào dùng để huấn luyện mô hình.
* **y\_train**: Nhãn tương ứng với dữ liệu đầu vào dùng để huấn luyện mô hình.
* **batch\_size=32**: Số lượng mẫu dữ liệu được sử dụng trong mỗi lần cập nhật trọng số. Trong trường hợp này, mỗi batch sẽ chứa 32 mẫu dữ liệu.
* **epochs=80**: Số lần toàn bộ dữ liệu huấn luyện được đưa qua mô hình. Ở đây, mô hình sẽ được huấn luyện trong 80 epochs.
* **validation\_data:**  Bộ dữ liệu kiểm tra được sử dụng để đánh giá hiệu suất của mô hình sau mỗi epoch.
* x\_test:
* Đánh giá kết quả:

A white rectangular object with a black stripe

Description automatically generated

* Giá trị 0.0864 cho thấy mức độ sai số bình phương trung bình là rất nhỏ, tức là sự khác biệt giữa các giá trị dự đoán và thực tế không quá lớn.

1. Test model

* So sánh giữa giá trị thực tế và giá trị huấn luyện:

A screenshot of a computer

Description automatically generated

* Giải thích code:

A screenshot of a computer code

Description automatically generated

* Khởi tạo biểu đồ **figure(figsize=(10, 6))**: Tạo một hình mới với kích thước 10x6 inch. Điều này giúp xác định kích thước của biểu đồ.
* **plot(y\_test[:13607], label="True Values")**: Vẽ đường biểu diễn cho giá trị thực tế (y\_test) từ mẫu đầu tiên đến mẫu thứ 13607. label="True Values" đặt nhãn cho đường này là "True Values".
* **plot(y\_pred[:13607], label="Predicted Values")**: Vẽ đường biểu diễn cho giá trị dự đoán (y\_pred) từ mẫu đầu tiên đến mẫu thứ 13607. label="Predicted Values" đặt nhãn cho đường này là "Predicted Values".
* **xlabel('Sample index')**: Đặt nhãn cho trục x là "Sample index", biểu thị chỉ số của các mẫu.
* **ylabel('GPA')**: Đặt nhãn cho trục y là "GPA", biểu thị giá trị cần dự đoán
* **title('Comparison of True and Predicted Values')**: Đặt tiêu đề cho biểu đồ.
* **legend()**: Hiển thị chú giải (legend) trên biểu đồ, giúp phân biệt giữa giá trị thực tế và giá trị dự đoán.
* **show()**: Hiển thị biểu đồ.
* Đánh giá biểu đồ:

A graph showing a number of values

Description automatically generated with medium confidence

* Biểu đồ cho thấy sự trùng khớp tương đối giữa giá trị thực tế và giá trị dự đoán.
* Tuy nhiên, đường màu cam (Predicted Values) có xu hướng dao động mạnh hơn và bao phủ hầu hết các giá trị thực tế, điều này cho thấy mô hình có khả năng dự đoán gần đúng, nhưng không ổn định hoàn toàn.
* Trong một số vùng nhất định, như GPA ở mức thấp (dưới 2) hoặc rất cao (trên 8), dự đoán của mô hình không hoàn toàn khớp với giá trị thực tế.
* Đường Predicted Values (cam) che phủ hầu hết đường True Values (xanh), điều này cho thấy mô hình hoạt động khá tốt ở mức tổng thể. Tuy nhiên, trong một số vùng, các giá trị True Values bị lệch hẳn khỏi dự đoán, đặc biệt ở các mẫu có sự biến đổi đột ngột.
* So sánh giữa mất mát khi huấn luyện và mất mát khi kiểm tra:

A graph of a training and validation loss

Description automatically generated

* Giải thích code:

A computer screen shot of text

Description automatically generated

* Lấy dữ liệu loss từ đối tượng history:

A close up of words

Description automatically generated

* **epochs = range(1, len(train\_loss) + 1):** Tạo danh sách các epoch.
* **figure(figsize=(10, 6)):** Khởi tạo biểu đồ
* **plot(epochs, train\_loss, '--', label='Training Loss'):** Vẽ đồ thị mất mát khi huấn luyện
* **plot(epochs, val\_loss, '--', label='Validation Loss'):** Vẽ đồ thị mất mát khi kiểm tra
* **title('Training and Validation Loss')**: Đặt tiêu đề cho biểu đồ
* **xlabel('Epochs')**: Đặt nhãn cho trục x là "Epochs", biểu thị số lượng epoch.
* **ylabel('MSE Loss')**: Đặt nhãn cho trục y là "MSE Loss", biểu thị giá trị mất mát (Mean Squared Error).
* **ylim(0, 2)**: Đặt giới hạn cho trục y từ 0 đến 2 để dễ dàng quan sát sự thay đổi của giá trị mất mát.
* **legend()**: Hiển thị chú giải (legend) trên biểu đồ, giúp phân biệt giữa mất mát khi huấn luyện và mất mát khi kiểm tra.
* **show()**: Hiển thị biểu đồ.
* Đánh giá biểu đồ:

A graph of a training and validation loss

Description automatically generated

* Mô hình có xu hướng giảm cả Training Loss và Validation Loss, chứng tỏ khả năng học và khái quát hóa dữ liệu khá tốt.
* Đường Validation Loss nằm gần Training Loss ở giai đoạn sau (khoảng epoch 40 trở đi), điều này cho thấy mô hình không bị overfitting nghiêm trọng.
* Cả Training Loss và Validation Loss đều giảm xuống dưới 0.5, cho thấy khả năng dự đoán chính xác của mô hình với dữ liệu.

* Đánh giá kết quả kiểm tra model:

A screen shot of a computer code

Description automatically generated

* Input: người dùng đã nhập một chuỗi điểm tương ứng với 24 kỳ học, trong đó có một số giá trị bất thường 0 và -1 ở một số kì học vì người dùng không có điểm ở các kì học đó
* Output: mô hình trả về Predicted GPA: 5.8426, đại diện cho điểm trung bình dự đoán của học sinh. Kết quả này là hợp lý khi so với các giá trị đầu vào (đa số nằm trong khoảng 5 - 7)