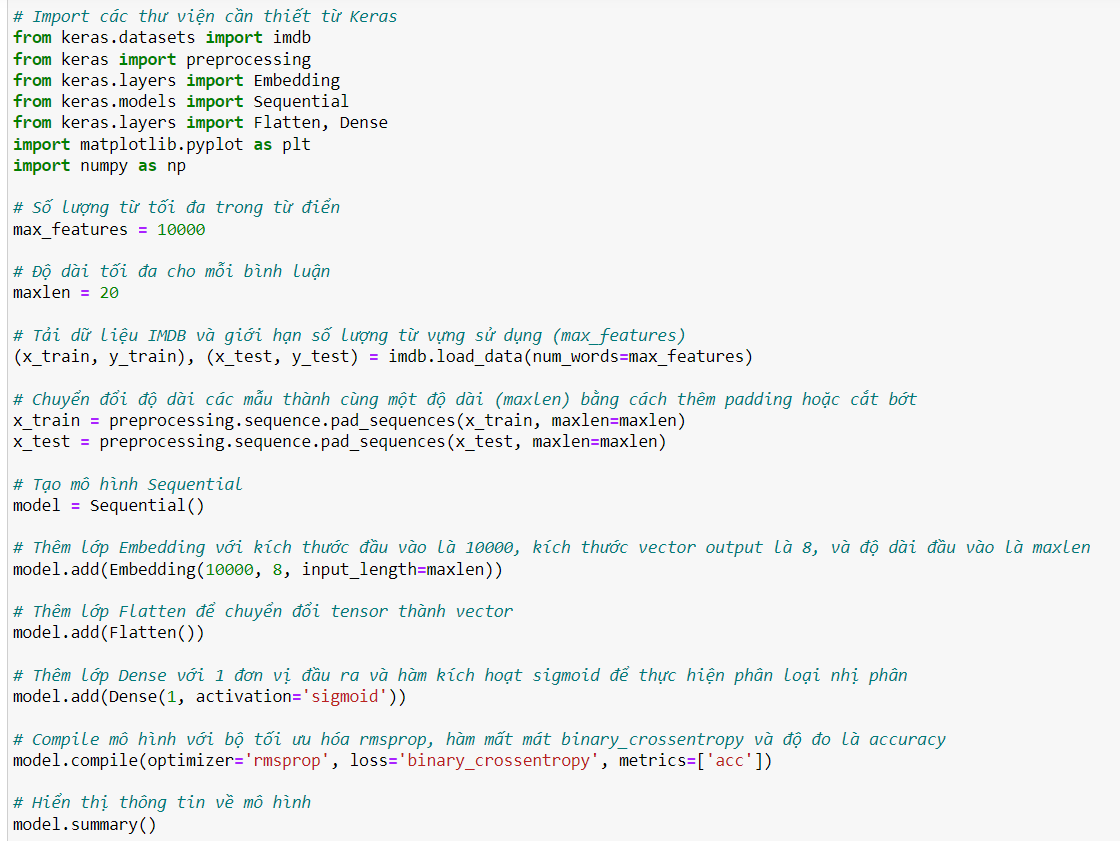
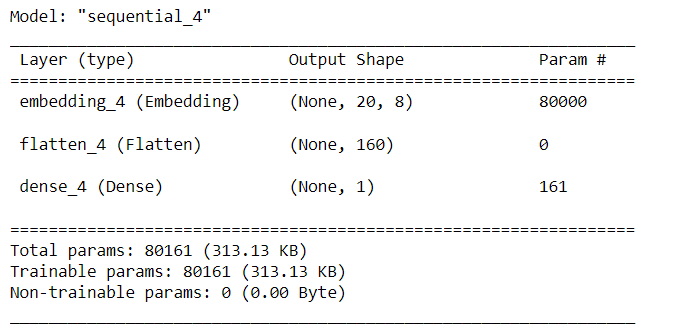
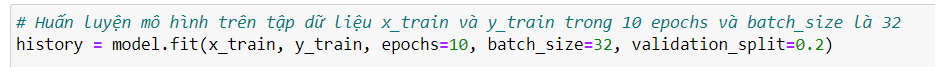
**5.2.b**

* **Chạy code**

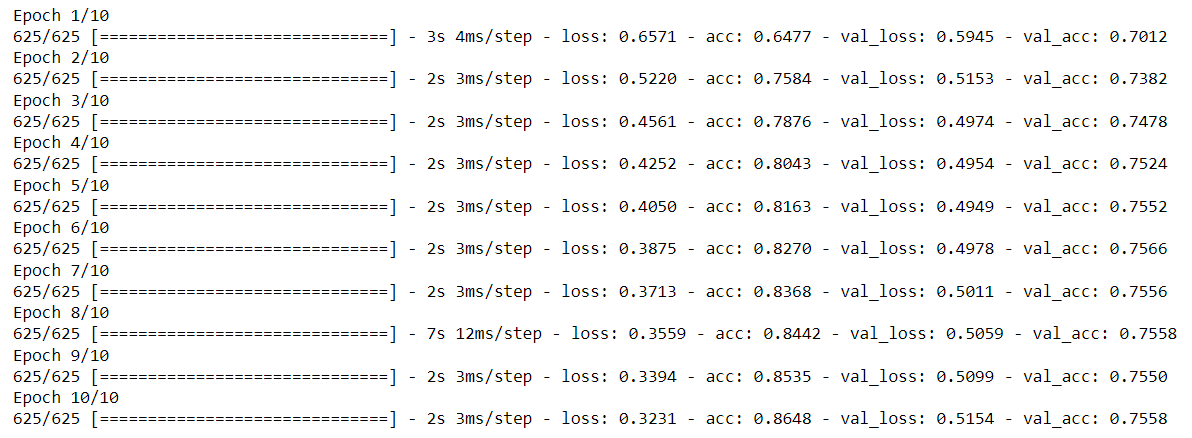
****

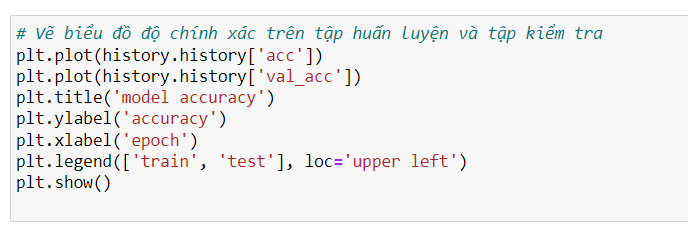
* ***Output***

******

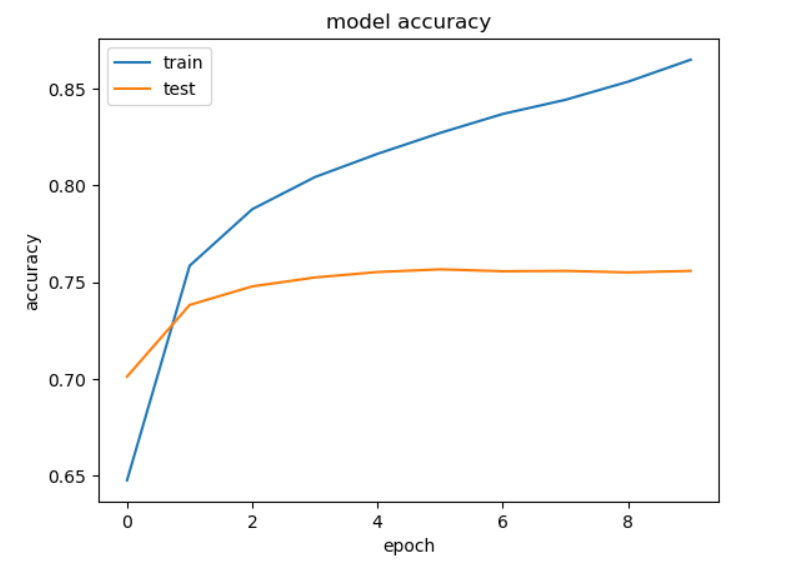
******

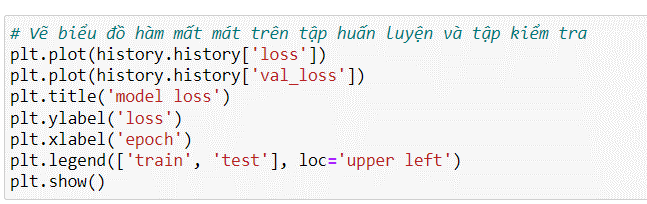
* ***Output***



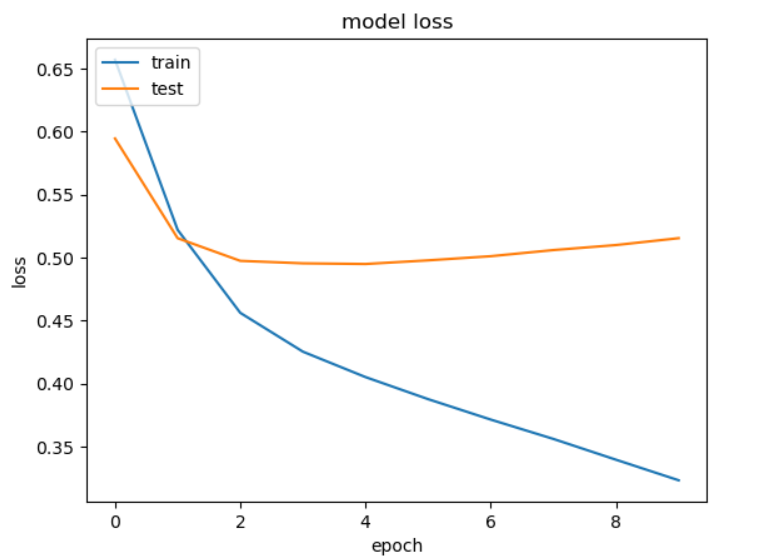


* ***Output***

******



* ***Output***



* **Giải thích code**

Code này sử dụng dữ liệu từ bộ dữ liệu IMDB để xây dựng mô hình phân loại văn bản dựa trên nội dung các bình luận phim. Code tiến hành tiền xử lý dữ liệu, xây dựng mô hình mạng nơ-ron, và sau đó huấn luyện mô hình và vẽ biểu đồ để theo dõi độ chính xác và hàm mất mát qua các epoch.

1. **keras.datasets**: Thư viện này cung cấp các tập dữ liệu mẫu để huấn luyện mô hình. Trong code này, nó được sử dụng để tải dữ liệu IMDB, một tập dữ liệu về các bình luận phim.
2. **keras.preprocessing**: Thư viện này cung cấp các công cụ cho việc tiền xử lý dữ liệu. Trong code này, nó được sử dụng để tiền xử lý văn bản, bao gồm cắt và thêm padding cho các bình luận.
3. **keras.layers**: Thư viện này chứa các lớp mạng nơ-ron chuẩn được sử dụng để xây dựng mô hình. Trong code này, nó được sử dụng để thêm các lớp như Embedding, Flatten và Dense vào mô hình.
4. **keras.models**: Thư viện này chứa Sequential, một kiểu mô hình mạng nơ-ron phổ biến trong Keras, cho phép bạn xây dựng mô hình mạng nơ-ron bằng cách thêm lần lượt các lớp vào mô hình.
5. **matplotlib.pyplot**: Thư viện này được sử dụng để vẽ biểu đồ và hiển thị đồ thị. Trong code này, nó được sử dụng để vẽ biểu đồ độ chính xác và hàm mất mát sau quá trình huấn luyện.
6. **numpy**: Thư viện cho tính toán số học. Trong code này, nó có thể được sử dụng để xử lý các phần của dữ liệu một cách thuận tiện.
7. **max\_features**: Biến này xác định số lượng từ vựng tối đa được sử dụng trong tập dữ liệu. Chỉ các từ xuất hiện thường xuyên nhất trong tập dữ liệu sẽ được sử dụng, và số lượng từ này không vượt quá giới hạn của max\_features.
8. **maxlen**: Độ dài tối đa cho mỗi bình luận. Các bình luận dài hơn sẽ bị cắt bớt, và các bình luận ngắn hơn sẽ được thêm padding để có độ dài maxlen.
9. **(x\_train, y\_train), (x\_test, y\_test):** Dữ liệu huấn luyện và kiểm tra được chia thành hai phần, mỗi phần bao gồm các mẫu và nhãn tương ứng. x\_train và x\_test chứa các mẫu dữ liệu, và y\_train và y\_test chứa nhãn tương ứng cho mỗi mẫu.
10. **Sequential**: Lớp này được sử dụng để xây dựng và định nghĩa mô hình mạng nơ-ron tuần tự. Mô hình mạng nơ-ron tuần tự là một kiểu mô hình mạng nơ-ron phổ biến trong deep learning, nơi các lớp mạng nơ-ron được xếp chồng lên nhau một cách tuần tự, từ lớp đầu tiên đến lớp cuối cùng.
11. **Embedding**: Lớp này thực hiện biểu diễn từ điển dưới dạng các vector số thực có kích thước 8 cho mỗi từ trong văn bản đầu vào. Lớp Embedding được sử dụng để biểu diễn các từ dưới dạng các vector có chiều thấp hơn để đưa vào mạng nơ-ron.
12. **Flatten**: Lớp này được sử dụng để chuyển đổi tensor thành vector. Trong trường hợp này, nó được sử dụng để làm phẳng dữ liệu sau khi áp dụng lớp Embedding.
13. **Dense**: Lớp này thêm một lớp fully connected layer với 1 đơn vị đầu ra và hàm kích hoạt sigmoid để thực hiện phân loại nhị phân. Trong trường hợp này, nó dự đoán xem một bình luận có tích cực (positive) hay tiêu cực (negative).
14. **compile():** Phương thức này cấu hình mô hình với các thông số tối ưu hóa (ở đây là 'rmsprop'), hàm mất mát (binary\_crossentropy) và các độ đo (accuracy) được sử dụng trong quá trình huấn luyện.
15. **fit():** Phương thức này huấn luyện mô hình trên tập dữ liệu huấn luyện (x\_train, y\_train) trong một số lượng epochs cụ thể (ở đây là 10) với kích thước batch cụ thể (ở đây là 32). Nó trả về lịch sử của quá trình huấn luyện, bao gồm các giá trị của độ chính xác và hàm mất mát trên tập huấn luyện và tập kiểm tra sau mỗi epoch.
16. **plt.plot():** Hàm này được sử dụng để vẽ biểu đồ độ chính xác và hàm mất mát trên tập huấn luyện và tập kiểm tra sau quá trình huấn luyện. Nó sử dụng dữ liệu lịch sử trả về từ fit() để vẽ biểu đồ.