Mô tả dự án:
Dữ liệu đầu vào: start time, end time, number of ingredients, venue được lưu trữ trong file ./Data/Data_AIL.xlsx
Dữ liệu đầu ra: viewer feeling of youtuber's style
Link file project: Github

I. Xử lí dữ liệu

Sau khi quan sát mức độ ảnh hưởng của các dữ liệu đầu vào liên quan đến mức độ tốt của model. Em quan sát thấy được khi bỏ đi dữ liệu ở cột container thì không ảnh hưởng đến performance của model nhiều. Nên 4 dữ liệu đã được em xử lí đưa vào train model là

- start time
- end time
- number of ingredients
- venue

II. Model

Sau khi quan sát thấy được model Linear regression chỉ có thể đưa ra các giá tục thì việc giúp model của ta có thể phân loại được các dữ liệu đầu vào vào các class 0, 1, 2, 3, 4, 5
Do đó hướng tiếp cận tiếp theo của em là sử dụng Logistic Regression để có thể classify dữ liệu đầu vào của ta vào 6 class 0, 1, 2, 3, 4, 5
Ưu điểm:

- Có thể phân loại các dữ liệu đầu vào của ta vào các class 0, 1, 2, 3, 4, 5 thay vì chỉ đưa ra các giá trị liên tục như Linear Regression
- Là thuật toán đơn giản và dễ thực thi

Nhược điểm:

- Quá trình training cũng như prediction của model không được nhanh. Vì cần phải sử dụng 6 model Logistic Regression để có thể phân loại dữ liệu của ta vào 6 class
- Không thể đưa ra được prediction tốt về các class 0, 1, 2 vì dữ liệu thuộc các class này ít hơn rất nhiều so với các class 3, 4, 5

III. Train

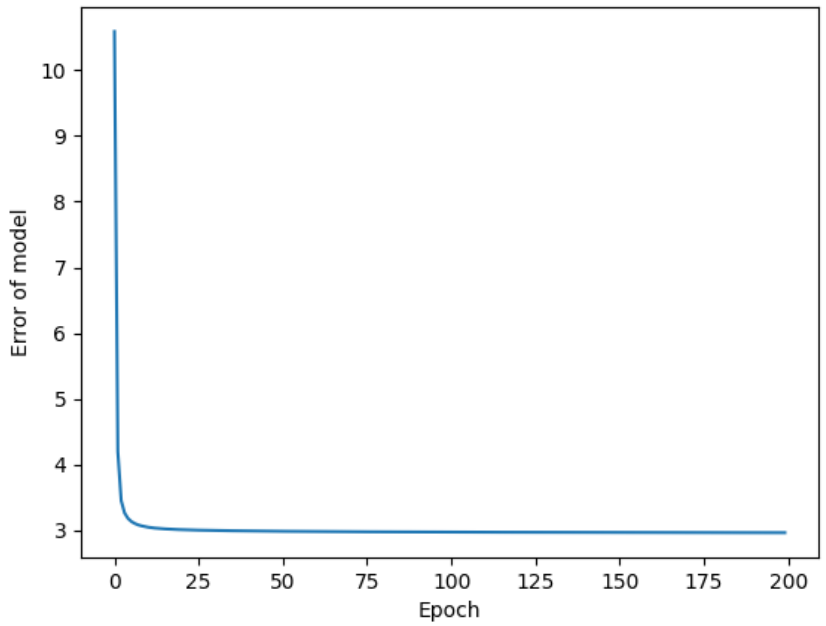
Để giúp model có thể cải thiện giá trị các parameter của model nhanh hơn thì em có những thay thế sau

- Bỏ đi cột *container* nhằm mục đích tăng tốc cho quá trình training là đưa ra dự đoán nhanh hơn
- Sử dụng SGD with momentum thay vì sử dụng SGD như thông thường để tối ưu các parameter, giúp tăng tốc quá trình training
- Thêm regularization vào trong hàm loss function để giúp model không bị overfitting
- Loss function được em sử dụng là Cross Entropy Loss
- Dữ liệu được dùng để training là 70% bộ dữ liệu và 30% được dùng để training

Ở lần đầu tiên khi chỉ thực hiện training với bộ dữ liệu chỉ gồm 3 feature: start time, end time, number of ingredients nhưng không thêm bias vào thì độ chính xác của thuật toán chỉ đạt được là 36%
Sau đó, bias cũng như venue được được thêm vào trong bộ dữ liệu giúp cho thuật toán được được hiệu quả tốt hơn 40.89% (ở test data) và các thông số của quá trình training này như sau

- Số lần training: 200
- Learning rate: 0.001

- Weight decay: 0.001 -> chỉ số được dùng trong phần regularization



Có thể thấy được quá trình tối ưu hóa các giá trị của model được diễn ra rất tốt. Nhưng càng về sau tốc độ đó càng giảm dần

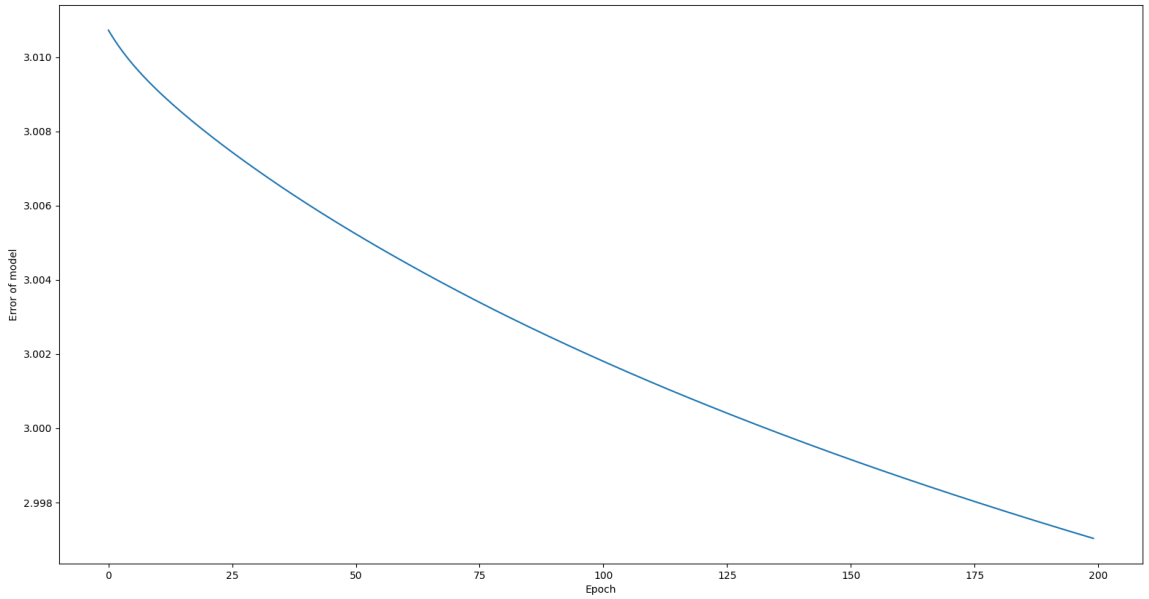
```
Mean Loss For testing data: 0.4089184060721063
test with precision tensor([[0.0000, 0.0000, 0.0000, 0.4092, 0.3891, 0.4645]])
test with recall tensor([[0.0000, 0.0000, 0.0000, 0.5650, 0.4388, 0.2802]])
test with f1 score tensor([[0.0000, 0.0000, 0.0000, 0.4746, 0.4125, 0.3495]])
test with macro f1 score 0.20610065758228302
```

Mặc dù đạt được hiệu quả cao hơn nhưng về các số liệu như precision, recall của model vẫn không đạt được hiệu quả tốt nhất. Chỉ có thể đưa ra dự đoán về các class 3, 4, 5. Ở đây có thể giải thích như sau

```
0.0       2
1.0      61
2.0     203
3.0    1190
4.0    1405
5.0     844
```

Vì đa phần dữ liệu của ta tập trung rất nhiều vào các class 3, 4, 5, nên việc model đưa ra dự đoán với các class 0, 1, 2 sẽ rất tệ

Sau đó, em thực hiện fine tune lại model này bằng cách giảm learning rate của model cũ xuống còn 0.0001 thì thấy được model của em vẫn thực hiện quá trình tối ưu rất tốt



Và hiệu quả của việc giảm đi learning rate này giúp cho model đạt được accuracy cũng như macro f1 score (giá trị trung bình của f1 score ứng với mỗi class)

```
Mean Loss For testing data: 0.4155597722960152
test with precision tensor([[0.0000, 0.0000, 0.0000, 0.4085, 0.4119, 0.4425]])
test with recall tensor([[0.0000, 0.0000, 0.0000, 0.5305, 0.4452, 0.3348]])
test with f1 score tensor([[0.0000, 0.0000, 0.0000, 0.4615, 0.4279, 0.3812]])
test with macro f1 score 0.2117740362882614
```

Lúc này độ hiệu quả đạt được đã tăng lên 0.3% và giá trị của macro f1 score của ta cũng tăng thêm

Và phương pháp để chọn được model nào tốt nhất của em là nếu model nào có macro f1 score cao hơn thì em sẽ chọn model đó

IV. Accuracy

Độ chính xác được cải thiện từ 36% lên 41,55%

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5
Precision	0.0	0.0	0.0	0.4085	0.4119	0.4425
Recall	0.0	0.0	0.0	0.5305	0.4452	0.3348
F1 score	0.0	0.0	0.0	0.4615	0.4279	0.3812