문혜지
201804202

이혜규
201804236

함도윤
201804255

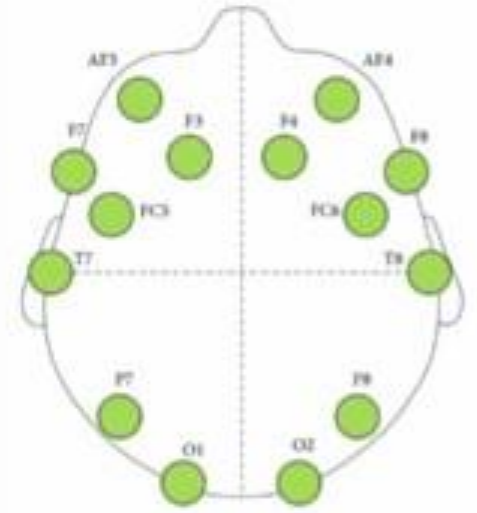
함양훈
202104397

Content

- Introduction
- Algorithm & Result
- Conclusion

Introduction

- 뇌파 분류 경진대회
- emotiv사의 EPOC 장비 이용하여 14개 채널로 각 뇌파를 받음
- 10초동안 '정지(0)', '왼쪽으로 이동(1)', '오른쪽으로 이동(2) ' 의 뇌파를 받음
- 뇌파를 받은 사람은 subjectA, subjectB, subjectC, subjectD, 총 4명



DATA

- 결측치 확인

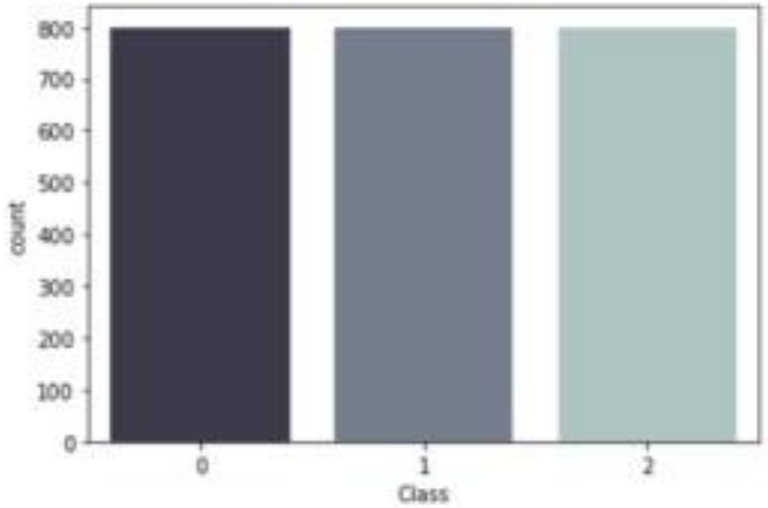
```
a = df_train.isnull().sum()
a[a > 0]
```

```
Series([], dtype: int64)
```

- 종속변수(Y) 확인

```
subject[target].value_counts()
```

```
0    800
2    800
1    800
Name: Class, dtype: int64
```



- 데이터 확인

	Class	AF3 delta std	AF3 delta m	AF3 theta std	AF3 theta m	AF3 alpha std	AF3 alpha m	AF3 beta std	AF3 beta m	F7 delta std
0	1	3574.0	2068.0	1.334961	2.294922	0.742188	2.052734	3.605469	4.855469	3576.0
1	1	3570.0	2064.0	1.639648	2.572266	1.096680	2.677734	2.314453	4.726562	3574.0
2	1	3570.0	2064.0	0.706543	2.613281	1.086914	2.222656	2.410156	4.937500	3574.0
3	1	3570.0	2066.0	0.939941	2.564453	0.836426	2.152344	2.6		
4	1	3568.0	2064.0	0.961914	2.373047	1.046875	1.783203	2.4		

F8 beta std	F8 beta m	AF4 delta std	AF4 delta m	AF4 theta std	AF4 theta m	AF4 alpha std	AF4 alpha m	AF4 beta std	AF4 beta m
3.625000	8.859375	3522.0	2045.0	0.750488	2.169922	1.372070	2.099609	2.417969	3.884766
3.917969	7.472656	3526.0	2048.0	0.799805	2.408203	1.110352	2.154297	2.048828	4.238281
4.355469	8.320312	3532.0	2050.0	1.411133	2.642578	1.518555	1.738281	1.457031	3.861328
3.683594	5.964844	3530.0	2050.0	1.221680	3.437500	1.272461	2.111328	1.679688	3.505859
3.498047	6.484375	3520.0	2044.0	1.796875	3.386719	0.770996	3.505859	1.477539	3.687500

DATA

- **정규화(normalization)**

$$x_i\_new = \frac{x_i - mean(x)}{std(x)}$$

각 데이터의 스케일을 맞추기 위해 진행

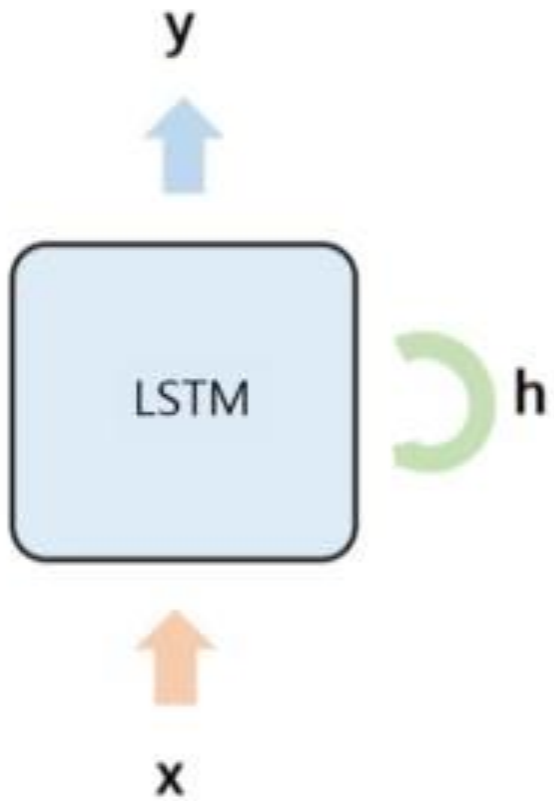
- **TRAIN TEST 분리**

TRAIN 7 : TEST 3로 지정

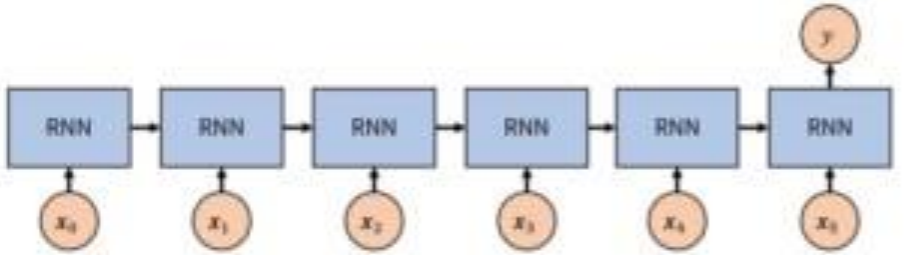
Algorithm & Result

LSTM

- LSTM 은 Long Short Term Memory의 약자
- 주로 시계열 처리나 자연어 처리를 사용하는 데 사용
- X가 Input으로 들어가 여러 번의 순환을 거쳐 output인 y가 나오는 구조. 이때 h(hidden state)는 그 중간다리 역할을 함.
- 최종적으로 y값이 나오는 데는 hidden state와 연속적인 x의 입력에 의해 결정



순환 신경망



Subject A의 과정

```
Epoch 1/100
53/53 [==============================] - 5s 17ms/step - loss: 0.5262 - accuracy: 0.4585
Epoch 2/100
53/53 [==============================] - 1s 17ms/step - loss: 0.1496 - accuracy: 0.6025
Epoch 3/100
53/53 [==============================] - 1s 17ms/step - loss: 0.0730 - accuracy: 0.6551
Epoch 4/100
53/53 [==============================] - 1s 17ms/step - loss: 0.0311 - accuracy: 0.6712 0s - loss: 0.0338
Epoch 5/100
53/53 [==============================] - 1s 17ms/step - loss: 0.0219 - accuracy: 0.6730
Epoch 6/100
53/53 [==============================] - 1s 17ms/step - loss: 0.0175 - accuracy: 0.6742
Epoch 7/100
53/53 [==============================] - 1s 17ms/step - loss: 0.0163 - accuracy: 0.6730
Epoch 00007: early stopping
```

LSTM

Subject_A Subject A 정확도 : 63.0%

```
LSTM train
 [[571   0   0]
 [223 337   0]
 [  1 353 188]]

LSTM test
 [[227   2   0]
 [ 88 142   0]
 [  2 175  81]]
```

Subject_B Subject B 정확도 : 67.0%

```
LSTM train
 [[555   0   0]
 [271 292   0]
 [  0 363 192]]

LSTM test
 [[245   0   0]
 [110 127   0]
 [  1 151  83]]
```

Subject_C Subject A 정확도 : 66.0%

```
LSTM train
 [[556   0   0]
 [205 348   0]
 [  0 200 364]]

LSTM test
 [[239   5   0]
 [ 90 147   0]
 [  0 133 103]]
```

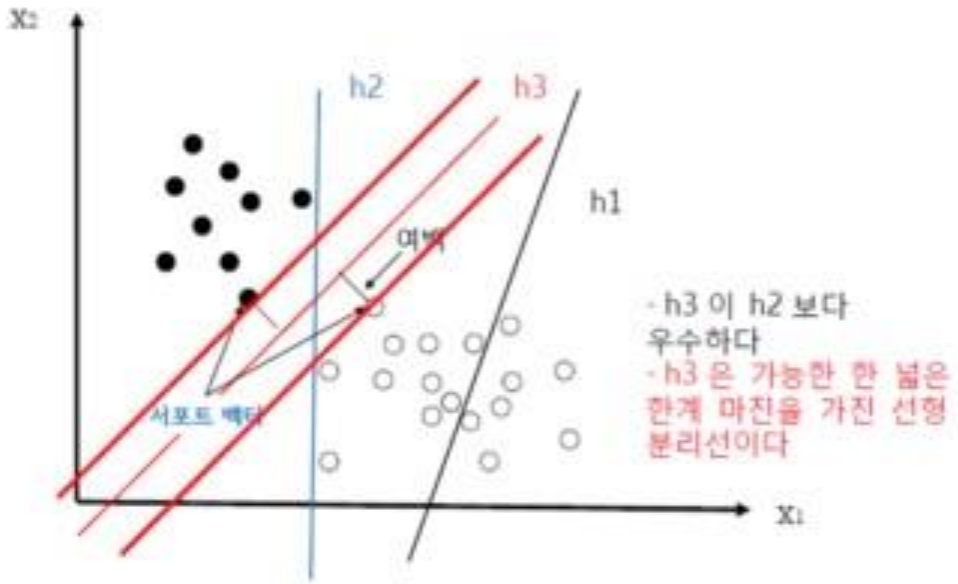
Subject_D Subject D 정확도 : 65.0%

```
LSTM train
 [[545   0   0]
 [412 164   0]
 [  0 389 163]]

LSTM test
 [[254   1   0]
 [152  62   0]
 [  0 186  62]]
```

SVM

- 서포트 벡터 머신(support vector machine)의 약자
- SVM은 기계 학습의 분야 중 하나로 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용됨.
- 분류된 두 범주 사이에서 해당 범주에 속하는 데이터의 분류 오차를 줄이면서 여백(margin)을 최대화하는 결정 경계(decision boundary)를 찾는 방법
- 두 범주 사이에 여러 경계가 존재할 수 있지만, 여백을 최대화하는 경계를 최대 마진 분류기라 함




```
> a_final_svm_model
══ Workflow [trained] ═══════════════════════════════════════════════════════
Preprocessor: Recipe
Model: svm_rbf()

── Preprocessor ────────────────────────────────────────────────────────────
1 Recipe Step

• step_normalize()

── Model ───────────────────────────────────────────────────────────────────
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 13.4494075098975

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.00767386792870953

Number of Support Vectors : 1674

Objective Function Value : -1580.792 -1699.441 -2087.19
Training error : 0.001667
Probability model included.
```

SVM

Subject_A

Subject A 정확도 : 87.6%

```
A SVM train
 [[496  37  23]
 [ 32 470  45]
 [ 23  48 506]]

A SVM test
 [[205  17  22]
 [ 20 191  42]
 [ 21  21 181]]
```

Subject_B

Subject B 정확도 : 94.9%

```
B SVM train
 [[534  14  17]
 [  6 531  12]
 [  7  29 530]]

B SVM test
 [[198  17  20]
 [ 11 230  10]
 [ 15  27 192]]
```

Subject_C

Subject C 정확도 : 82.1%

```
C SVM train
 [[469  38  51]
 [ 45 459  45]
 [ 44  78 451]]

C SVM test
 [[153  38  51]
 [ 40 168  43]
 [ 51  58 118]]
```

Subject_D

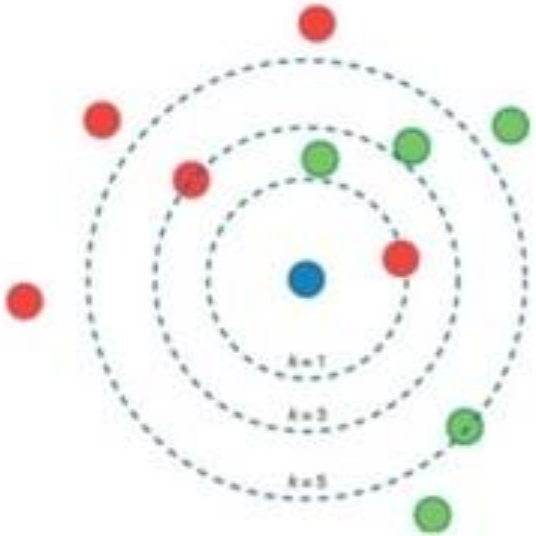
Subject D 정확도 : 90.7%

```
D SVM train
 [[515  22  33]
 [ 21 519  23]
 [ 28  30 489]]

D SVM test
 [[176  24  30]
 [ 21 192  24]
 [ 28  36 189]]
```

KNN

- k-Nearest Neighbor 모델의 약자
- 새로운 데이터가 어느 그룹에 속해있는지를 판단하기 위해서, 인접한 학습데이터의 개수를 기준으로 판단
- 이때 인접한 학습데이터의 개수가 바로 kNN의 'k ' 가 됨



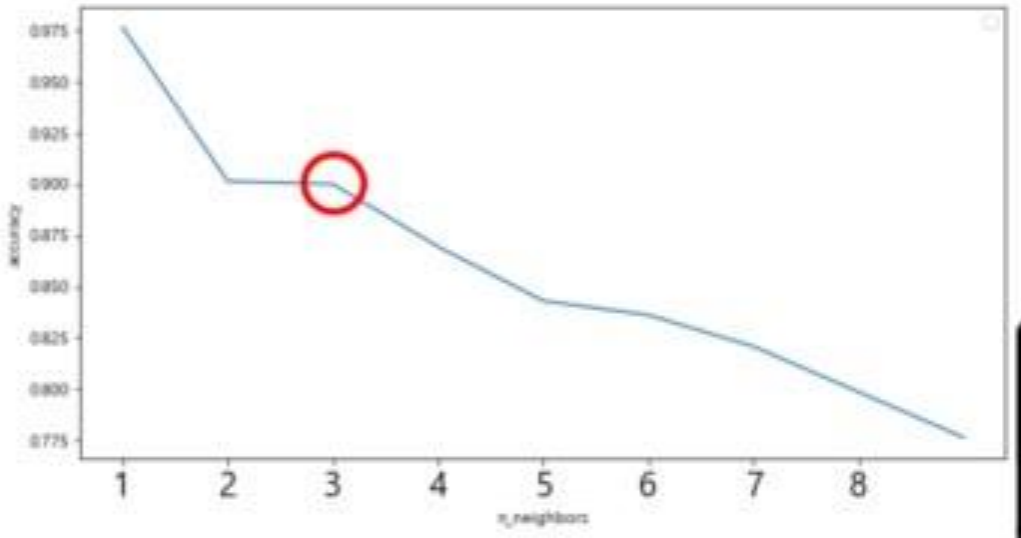
빨간색과 녹색은 학습된 데이터이고 파란색은 새로운 데이터

k=1이면 빨간색
k=3이면 빨간색
k=5이면 녹색

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)
knn_train_pred = knn.predict(x_train)
knn_test_pred = knn.predict(x_test)
```

k=3 으로 사용



KNN

Subject_A

Subject A 정확도 : 96.6%

```
KNN train
 [[535   7   9]
 [  8 550   8]
 [ 12  13 538]]

KNN test
 [[215  17  17]
 [ 15 208  11]
 [ 14  23 200]]
```

Subject_B

Subject B 정확도 : 98.6%

```
B KNN train
 [[571   1   3]
 [  4 533   6]
 [  5   5 552]]

B KNN test
 [[212   7   6]
 [ 19 220  18]
 [ 15   4 219]]
```

Subject_C

Subject A 정확도 : 89.7%

```
D KNN train
 [[539   3   1]
 [  2 560   5]
 [  6   2 562]]

D KNN test
 [[241   6  10]
 [ 11 214   8]
 [ 13  10 207]]
```

Subject_D

Subject D 정확도 : 98.9%

```
C KNN train
 [[509  19  22]
 [ 30 508  25]
 [ 49  28 490]]

C KNN test
 [[190  33  27]
 [ 41 172  24]
 [ 50  24 159]]
```

Conclusion

Conclusion

Algorithm	LSTM	SVM	KNN
Subject A의 정확도	63.0%	87.6%	96.0%
Subject B의 정확도	67.0%	94.9%	98.6%
Subject C의 정확도	66.0%	82.1%	89.7%
Subject D의 정확도	65.0%	90.7%	98.9%

LSTM

Subject_A, B, C, D 예측값 error

```
model.predict(x_test)
```

```
>>>     outputs = call_fn(inputs, *args, **kwargs)
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\utils\traceback_utils.py", line 92, in error_handler
>>>     return fn(*args, **kwargs)
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\layers\recurrent_v2.py", line 1254, in call
>>>     runtime) = lstm_with_backend_selection(**normal_lstm_kwargs)
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\layers\recurrent_v2.py", line 1649, in lstm_with_backend_selection
>>>     last_output, outputs, new_h, new_c, runtime = defun_standard_lstm(**params)
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\layers\recurrent_v2.py", line 1380, in standard_lstm
>>>     last_output, outputs, new_states = backend.rnn(
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\backend.py", line 4654, in rnn
>>>     final_outputs = tf.compat.v1.while_loop(
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\backend.py", line 4638, in _step
>>>     current_input = tuple(ta.read(time) for ta in input_ta)
>>>
>>>   File "C:\Users\yhg31\Anaconda3\lib\site-packages\keras\backend.py", line 4638, in <genexpr>
>>>     current_input = tuple(ta.read(time) for ta in input_ta)
```

SVM

Subject_A

Class	0	1	2
예측값	195	160	125

Subject_B

Class	0	1	2
예측값	228	168	84

Subject_C

Class	0	1	2
예측값	243	139	98

Subject_D

Class	0	1	2
예측값	223	151	106

KNN

Subject_A

Class	0	1	2
예측값	209	146	125

Subject_B

Class	0	1	2
예측값	192	172	116

Subject_C

Class	0	1	2
예측값	226	132	122

Subject_D

Class	0	1	2
예측값	195	149	136

Thanks!