# 표정분석을 통한 노래 추천 AI

잘해보조 – 문성욱, 김경선, 서주완

# 목차

# 01

## 프로젝트 소개

프로젝트 방향 및 진행 전 준비사항 소개

# 회의록

## 회의록

| 회의일시 | 2022년 11월 4일 17:00~17:30 | 장소 | |
|---|---|---|---|
| 참석자 | 문성욱, 김경선, 서주완 | | |

| 회의안건 | 개발 |
|---|---|

| 회의내용 | 내용 |
|---|---|
| | 1. 데이터셋 사용<br>데이터셋은 얼굴인식과 표정분류로 구분된 두 단계를 거치는 얼굴표<br>OpenCV 라이브러리를 사용하여 cascade 기반으로 미리 학습된 정<br>얼굴 식별 후, 표정을 구분할 수 있도록 얼굴의 구성요소를 찾기위해<br>얼굴 식별 -> 얼굴의 구성요소 찾기 -> 표정 구분 순서<br><br>2. 파이썬 패키지 및 모듈<br>Python 언어, Dilb Python Wrapper, OpenCV, SciP, Matplotl<br>itertools, sys, os, argparse, pandas, warnings<br><br>3. 추천해줄 음악 선정<br>각 기분별 예시로 사용할 음악 하나씩 선정<br>ex) 신나는 음악, 차분한 음악, 웅장한 음악 등<br><br>4. 추후 계획<br>- 개발 중간 점검 : 2022-11-16 예정<br>- 최종 발표 자료 준비 : 2022-12-07 예정 |

## 회의록

| 회의일시 | 2022년 11월 16일 17:00~17:30 | 장소 | |
|---|---|---|---|
| 참석자 | 문성욱, 김경선, 서주완 | | |

| 회의안건 | 개발 중간 점검 |
|---|---|

| 회의내용 | 내용 |
|---|---|
| | 1. 진행하면서 문제점<br>정확도, gtts, OpenCV<br><br>2. 표정인식 정확도<br>표정인식시 정확도가 30% 이하로 나옴<br>높이기 위하여 데이터셋 추가 수집 및 ephocs 훈련진행<br>-> 50% 이상으로 올리려 했지만 40% 정도로 상승<br><br>3. gtts 문제점<br>관련해서 playsound가 제대로 작동하지 않음<br>다른 방법 모색결과 audio display로 대체가능한걸 알아냈고<br>audio display 사용하기로 함<br><br>4. OpenCV 문제점<br>OpenCV를 사용하려 했지만 사용이 미숙하여 추가적인 공부필요<br><br>5. 추후 계획<br>- 개발 중간 점검 : 2022-11-16 예정<br>- 최종 발표 자료 준비 : 2022-12-07 예정 |

## 회의록

| 결재 | 팀장 | 팀원 | 팀원 |
|---|---|---|---|
| | 문성욱 | 김경선 | 서주완 |

| 회의일시 | 2022년 12월 07일 17:00~17:30 | 장소 | 학교 카페 | 작성자 | 문성욱 |
|---|---|---|---|---|---|
| 참석자 | 문성욱, 김경선, 서주완 | | | | |

| 회의안건 | 최종 발표 자료 준비 |
|---|---|

| 회의내용 | 내용 |
|---|---|
| | 1. 콘텐츠 어떻게 보여줄 것 인가?<br>영상촬영 프로그램을 통해 과정을 영상으로 촬영 -> PPT에 기입<br><br>2. 발표 PPT 틀 구상<br>PPT에 어떤 내용을 넣을 것 인지에 대한 구상<br>1) 개발내용<br>2) 개발과정<br>3) 시연영상<br>4) 시행착오<br>5) 느낀 점<br><br>3. 발표자 대본 설정<br>최종 발표를 어떤식으로 진행하고, 어떤 말을 할 것 인지에 회의<br>발표시간은 10분정도, 개발과정에서 데이터 셋이 어떻게 사용되었고,<br>파이썬 패키지 및 모듈이 어떤식으로 사용 되었는지,<br>시행착오 (즉 막혔던 부분이나 쉽게 진행되었던 부분 등),<br>이번 프로젝트를 진행하면서 각 팀원이 느꼈던 점 |

# 프로젝트 방향

## 쉬운 코드

AI 모델을 처음 개발
→ 모델 직접 구현보단
기존의 모델 이해하는 방향

## 모델 활용

AI 모델 코드를
더 잘 이해할 수 있도록
모델 활용해보기

## 실용적

실제로 사용할 수 있는
방향으로 활용해보기

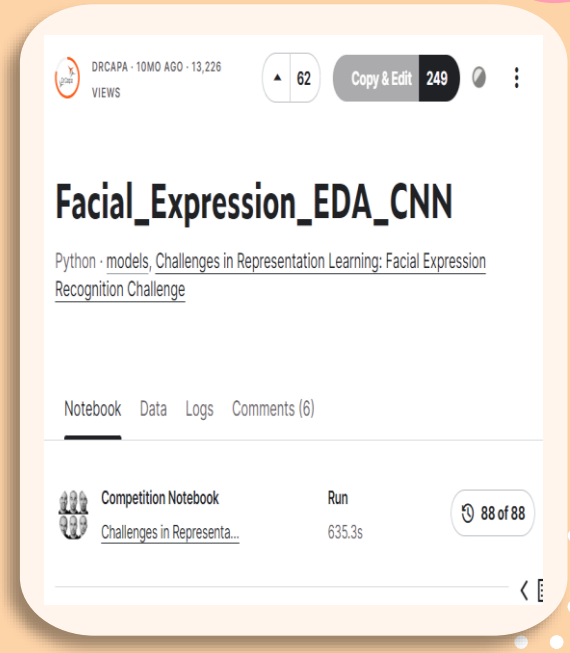
# 얼굴인식, 표정분석

## 모델 활용 노래 추천

# 표정분석 사용 데이터셋 및 참고한 코드

| | emotion | pixels |
|---|---|---|
| 1 | 0 | 70 80 82 72 58 58 60... |
| 2 | 0 | 151 150 147 155 148 ... |
| 3 | 2 | 231 212 156 164 174 ... |
| 4 | 4 | 24 32 36 30 32 23 19... |
| 5 | 6 | 4 0 0 0 0 0 0 0 0 0 0 ... |
| 6 | 2 | 55 55 55 55 55 54 60... |
| 7 | 4 | 20 17 19 21 25 38 42... |
| 8 | 3 | 77 78 79 79 78 75 60... |
| 9 | 3 | 85 84 90 121 101 102... |
| 10 | 2 | 255 254 255 254 254 ... |
| 11 | 0 | 30 24 21 23 25 25 49... |
| 12 | 6 | 39 75 78 58 58 45 49... |
| 13 | 6 | 219 213 206 202 209 ... |

사용 데이터셋 : Facial Expression Recognition Challenge

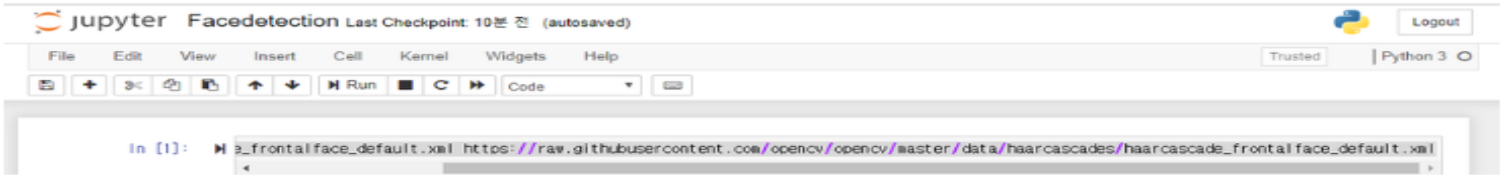



참고 코드 : DRCAPA - Facial_Expression_EDA_CNN

# 얼굴인식 참고 코드 및 모델

## 2. Jupyter Notebook에서 얼굴 인식하기 (이미지)

먼저, 얼굴 인식을 위해서 얼굴 데이터를 불러옵니다.




haarcascade_frontalface_default.xml이라는 파일을 불러서 저장합니다.

```
xml https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_frontalface_default.xml
```

위와 같이 입력한 다음에 컨트롤+엔터를 입력하면, 파일을 다운로드 받았음을 확인할 수 있습니다.

얼굴인식 참고 코드 : 오픈랩 – OpenCV 설치와 얼굴 인식

# 02

## 프로젝트 설명

프로젝트 수행 과정 및 코드 등 설명

# 프로젝트 수행 과정

01

## 코드 분석

참고할 코드,
사용할 모듈들을
분석하고 이해하기

02

## 모델 분석

사용할 모델을
분석해보기

03

## 코드 사용

참고한 코드,
모델을 주제에
맞게 변경하고
활용해보기

04

## 테스트

활용한 코드를
테스트해보기

# 주요 코드 설명

```python
def prepare_data(data):
    image_array = np.zeros(shape=(len(data), 48, 48))
    image_label = np.array(list(map(int, data['emotion'])))

    for i, row in enumerate(data.index):
        image = np.fromstring(data.loc[row, ' pixels'], dtype=int, sep=' ')
        image = np.reshape(image, (48, 48))
        image_array[i] = image

    return image_array, image_label
```

```python
train_image_array, train_image_label = prepare_data(data[data[' Usage']=='Training'])
val_image_array, val_image_label = prepare_data(data[data[' Usage']=='PrivateTest'])
test_image_array, test_image_label = prepare_data(data[data[' Usage']=='PublicTest'])
```

사용할 데이터를 준비 → 훈련용, 검증용, 테스트용으로 분리

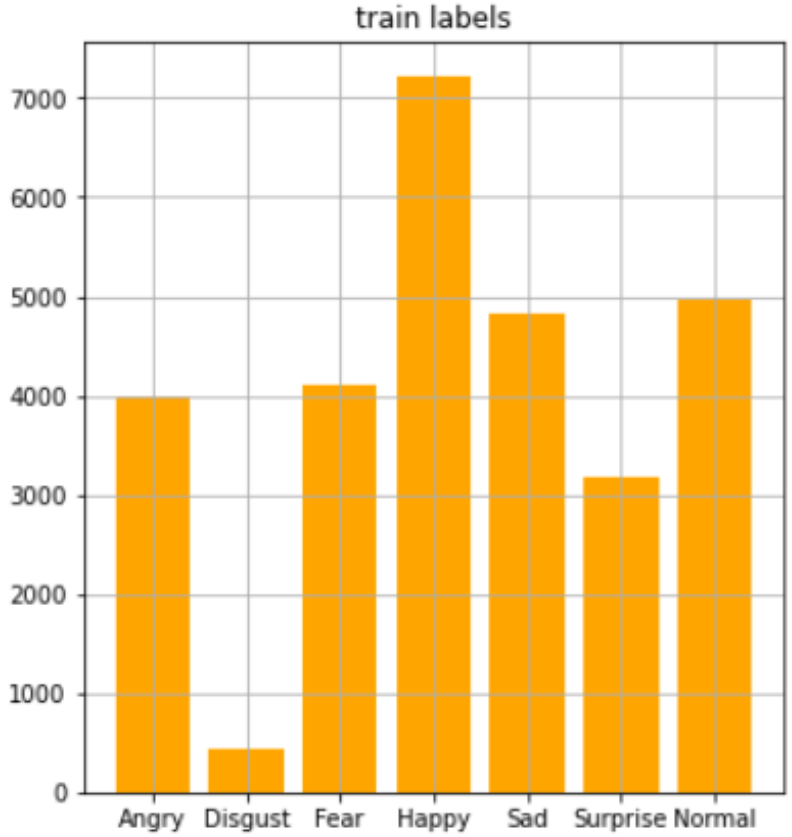
# 주요 코드 설명

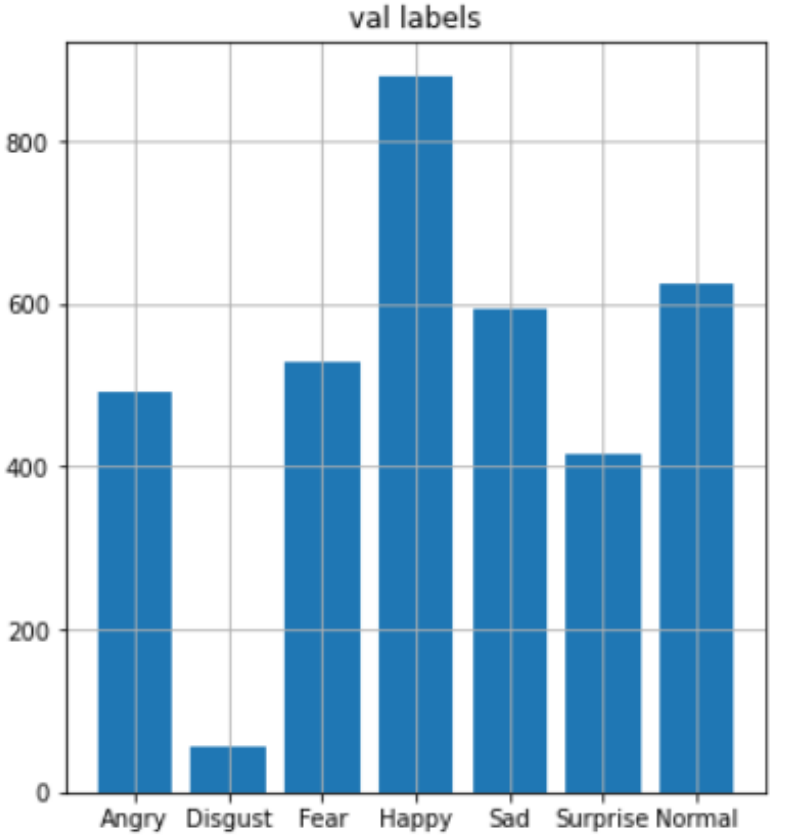
```
train_images = train_image_array.reshape((train_image_array.shape[0], 48, 48, 1))
train_images = train_images.astype('float32')/255
val_images = val_image_array.reshape((val_image_array.shape[0], 48, 48, 1))
val_images = val_images.astype('float32')/255
test_images = test_image_array.reshape((test_image_array.shape[0], 48, 48, 1))
test_images = test_images.astype('float32')/255
```

```
train_labels = to_categorical(train_image_label)
val_labels = to_categorical(val_image_label)
test_labels = to_categorical(test_image_label)
```

```
emotions = {0: 'Angry', 1: 'Disgust', 2: 'Fear', 3: 'Happy', 4: 'Sad', 5: 'Surprise', 6: 'Normal'}
```

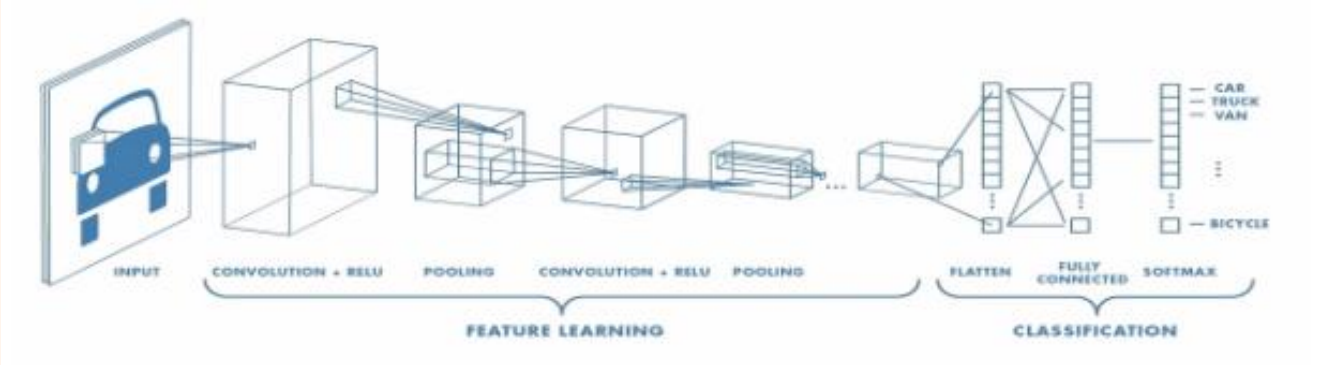
데이터의 shape과 type을 학습시키기 적절하게 변경,
데이터셋의 label들을 원-핫 인코딩




학습용 데이터 약 30,000개
검증용 데이터 약 3,600개

# 주요 코드 설명




## CNN(Convolutional Neural Network)이란?

- 인간의 시신경 구조를 모방한 기술
- 이미지 인식 위한 패턴 찾는데 유용
- 불변하는 특징 찾고 특징을 입력데이터로 신경망에 보내 분류 수행

# 주요 코드 설명

```
model = models.Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(7, activation='softmax'))
```

```
model.compile(optimizer=Adam(learning_rate=1e-3), loss='categorical_crossentropy', metrics=['accuracy'])
```

## CNN 학습 모델 만들기

- Conv2D : 필터로 특징 뽑기
- MaxPool2D : 정해진 크기 내에서 가장 큰 값 뽑기
- Flatten : 다차원 데이터를 1차원 데이터로 변경
- Dense : 추출된 정보 모아 원하는 차원으로 표현
- Adam : CNN에서 적은 시간에 효율적으로 신경망 훈련 가능

# 주요 코드 설명

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 46, 46, 32)        320

 max_pooling2d (MaxPooling2D  (None, 23, 23, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 64)          36928

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 64)                262208

 dense_1 (Dense)             (None, 7)                 455

=================================================================
Total params: 318,407
Trainable params: 318,407
Non-trainable params: 0
_________________________________________________________________
```

# 주요 코드 설명

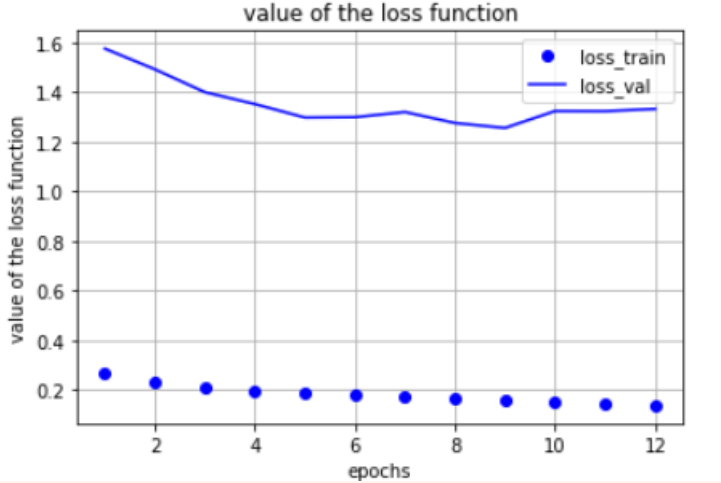
```
history = model.fit(train_images, train_labels,
                    validation_data=(val_images, val_labels),
                    class_weight = class_weight,
                    epochs=12,
                    batch_size=64)
```

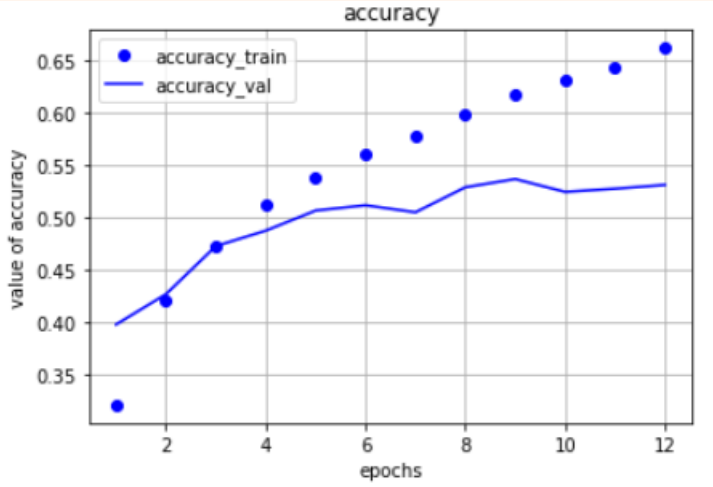
```
Epoch
449/4                                                                    :979
Epoch
449/4                                                                    :269
Epoch
449/4                                                                    :728
Epoch
449/4                                                                    :876
Epoch
449/449 [==============================] - 41s 92ms/step - loss: 0.1675 - accuracy: 0.5576 - val_loss: 1.2970 - val_accuracy: 0.5068
Epoch 6/12
449/449 [==============================] - 40s 89ms/step - loss: 0.1786 - accuracy: 0.5600 - val_loss: 1.2991 - val_accuracy: 0.5118
Epoch 7/12
449/449 [==============================] - 37s 83ms/step - loss: 0.1711 - accuracy: 0.5776 - val_loss: 1.3198 - val_accuracy: 0.5052
Epoch 8/12
449/449 [==============================] - 37s 83ms/step - loss: 0.1639 - accuracy: 0.5986 - val_loss: 1.2755 - val_accuracy: 0.5291
Epoch 9/12
449/449 [==============================] - 39s 87ms/step - loss: 0.1568 - accuracy: 0.6180 - val_loss: 1.2558 - val_accuracy: 0.5369
Epoch 10/12
449/449 [==============================] - 41s 91ms/step - loss: 0.1507 - accuracy: 0.6312 - val_loss: 1.3237 - val_accuracy: 0.5247
Epoch 11/12
449/449 [==============================] - 39s 88ms/step - loss: 0.1442 - accuracy: 0.6444 - val_loss: 1.3233 - val_accuracy: 0.5277
Epoch 12/12
449/449 [==============================] - 41s 92ms/step - loss: 0.1378 - accuracy: 0.6622 - val_loss: 1.3321 - val_accuracy: 0.5313
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test accuracy:', test_acc)
```

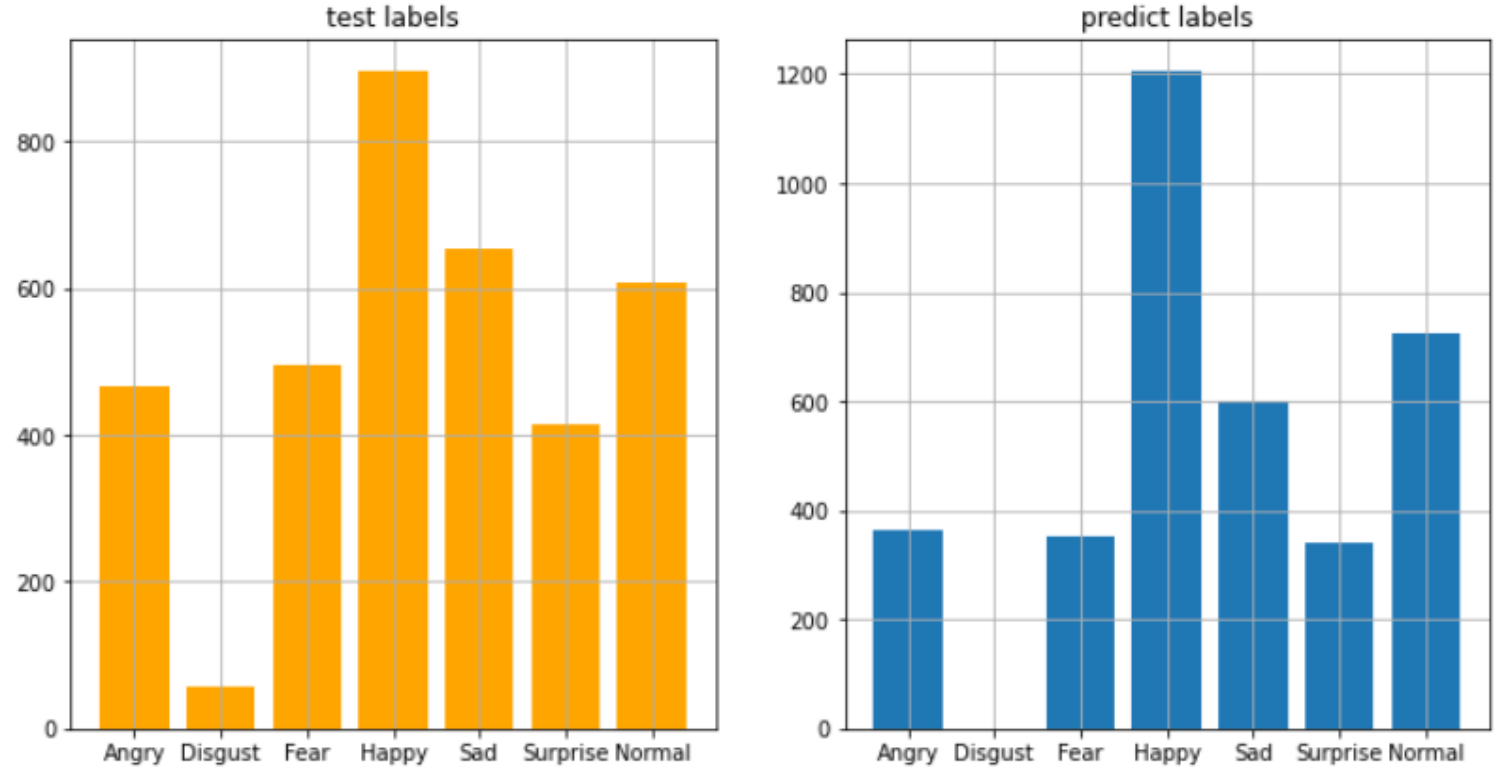
```
113/113 [==============================] - 2s 15ms/step - loss: 1.3329 - accuracy: 0.5408
test accuracy: 0.5408191680908203
```

# 주요 코드 설명


value of the loss function
loss_train
loss_val
value of the loss function
epochs
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
2
4
6
8
10
12


accuracy
accuracy_train
accuracy_val
value of accuracy
epochs
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2
4
6
8
10
12

# 주요 코드 설명


test labels
0
200
400
600
800
Angry
Disgust
Fear
Happy
Sad
Surprise
Normal
predict labels
0
200
400
600
800
1000
1200
Angry
Disgust
Fear
Happy
Sad
Surprise
Normal

# 주요 코드 설명

```
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

t_gray = cv2.cvtColor(t_img, cv2.COLOR_BGR2GRAY)

t_faces = faceCascade.detectMultiScale(t_gray, scaleFactor = 1.1, minNeighbors = 5, minSize = (30, 30))
```

```
t_img_cut = t_img[t_faces[0][0] : t_faces[0][0] + t_faces[0][2], t_faces[0][1] : t_faces[0][1] + t_faces[0][3]] #
```

```
t_img_gray = t_gray[t_faces[0][0] : t_faces[0][0] + t_faces[0][2], t_faces[0][1] : t_faces[0][1] + t_faces[0][3]]
t_img_gray = cv2.resize(255 - t_img_gray, (48, 48)) # grayscale이 적용된 사진을 48 * 48의 크기로 맞춤
```

**실제로 모델을 사용하기 위한 데이터 준비**

- 사진 파일에서 얼굴을 인식하여 인식한 부분을 자름
- 그 후 사진에 grayscale을 적용, 모델에 적합한 크기로 조절
- 조절된 이미지 픽셀 값들을 저장해 DataFrame으로 만듦

```
t_height, t_width = t_img_gray.shape[:2]

t_data_pixels = ''
for y in range(t_height) :
    for x in range(t_width) :
        t_data_pixels = t_data_pixels + str(t_img_gray[y, x]) + ' '

t_data_array = np.array([t_data_pixels])
t_data_array.shape
```

```
(1,)
```

```
t_data = pd.DataFrame(t_data_array, columns = ['pixels'])
t_data
```

# 주요 코드 설명

```
t_image_array = np.zeros(shape=(len(t_data), 48, 48))

for i, row in enumerate(t_data.index):
    t_image = np.fromstring(t_data.loc[row, 'pixels'], dtype = int, sep = ' ')
    t_image = np.reshape(t_image, (48, 48))
    t_image_array = t_image
```

```
t_test_image = t_image_array.reshape((48, 48, 1))
t_test_image = t_test_image.astype('float32')/255
```

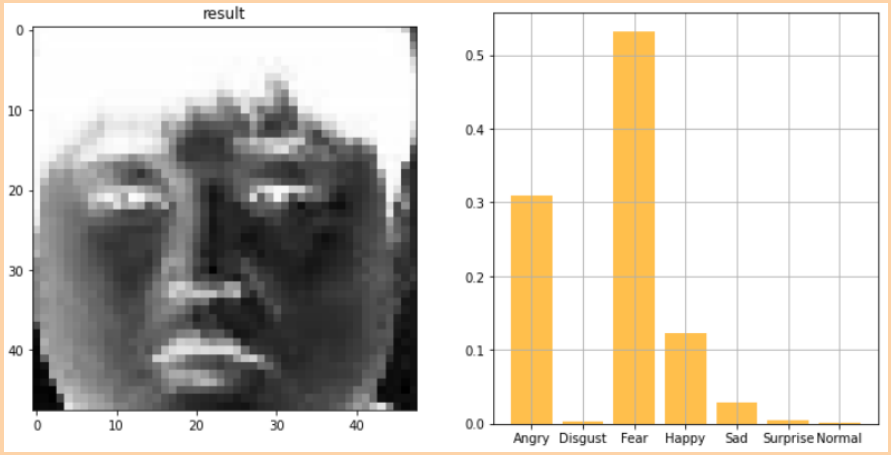
```
t_new_image = tf.expand_dims(t_test_image,0)
pred_test_labels = model.predict(t_new_image)
```

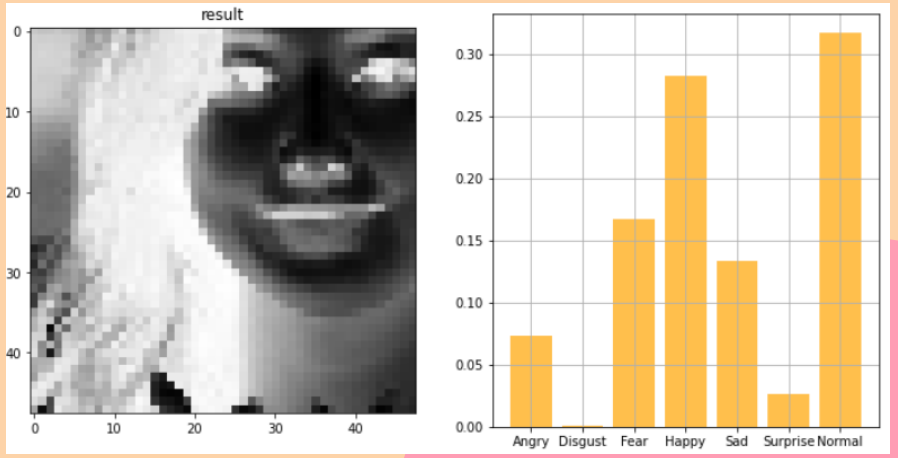
```
1/1 [==============================] - 0s 36ms/step
```

## 데이터 준비 및 모델에 적용해보기

- DataFrame에 저장된 픽셀 값들을 tensor 형태로 저장
- tensor 형태로 저장된 픽셀 값들을 모델에 적용하기 적당한 크기로 reshape
- 모델은 데이터셋에 맞게 만들어져 있어 사용할 데이터 차원보다 1차원이 큼
  → 사용할 데이터의 차원을 강제로 1늘려준 후 모델에 적용

# 주요 코드 설명


result
Angry
Disgust
Fear
Happy
Sad
Surprise
Normal


result
Angry
Disgust
Fear
Happy
Sad
Surprise
Normal

# 주요 코드 설명

```
kor = gTTS('당신의 기분은 ' + t_result + '인 것 같네요! 장르를 선택해주시면 음악을 추천해드릴게요', lang = 'ko', slow = False)
kor.save('result.mp3')
display(Audio('result.mp3', autoplay = True))

print('당신의 기분은 ' + t_result + '인 것 같네요')
print()

print("당신에게 추천해주고 싶은 음악을 찾았어요! 듣고 싶은 장르를 선택해주세요")
print("1. K-POP   2. POP   3. HIPHOP   4. 가사 없는 음악")
print()
music = input("듣고 싶은 장르의 숫자 번호만 입력해주세요! 다른 키를 입력하시면 프로그램이 종료됩니다 : ")
```

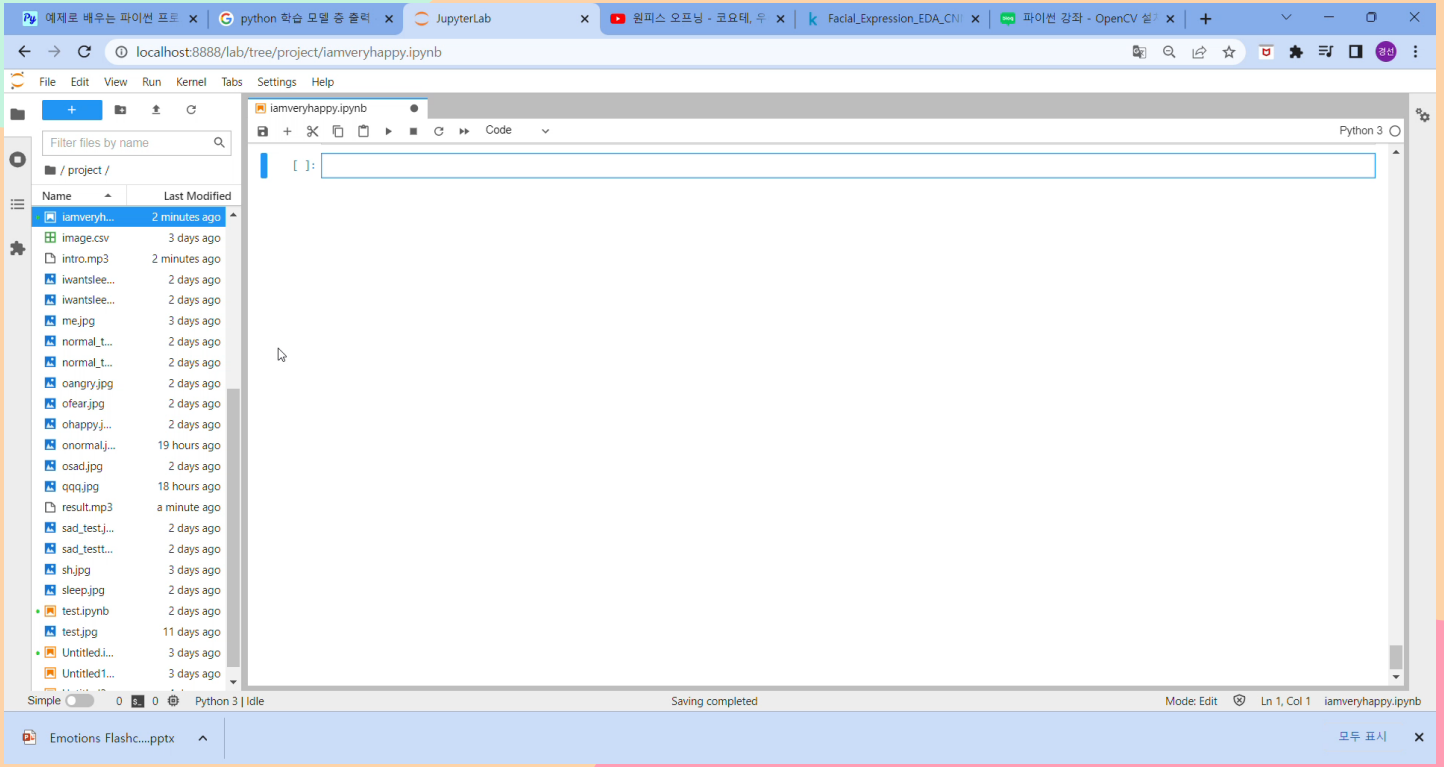
**결과에 따른 노래 추천 프로그램 완성**

- 모델을 활용한 코드들을 함수로 묶기
- gTTS와 display를 사용해서 사용자에게 결과 등을 소리로 알려줌
- 결과에 따라 리스트로 만들어 둔 노래 중 한 곡이 댄덤을 추천됨

```
def recommend(t_result, music) :
    if (t_result == "Happy" or t_result == "Surprise") and music == '1':
        happy_kpop = ["르세라핌 - Antifragile (https://youtu.be/pyf8cbqyfPs)",
                      "뉴진스 - hype boy (https://youtu.be/9wUKhEgnllc)",
                      "코요테 - 우리의꿈 (https://youtu.be/junvn7qKikc)"]
        ran_num = random.randint(0, 2)
        print(happy_kpop[ran_num])

    elif (t_result == "Happy" or t_result == "Surprise") and music == '2':
        happy_pop = ["Christopher - Bad(https://youtu.be/T6tTohWiu5A)",
                     "Imagine Dragons - Believer(https://youtu.be/7wtfhZwyrcc)",
                     "Justin Bieber - Peaches (https://youtu.be/tQ0yjYUFKAE)"]
        ran_num = random.randint(0, 2)
        print(happy_pop[ran_num])
```

# 시연영상


예제로 배우는 파이썬 프로
python 학습 모델 층 출력
JupyterLab
원피스 오프닝 - 코요테, 우
Facial_Expression_EDA_CNN
파이썬 강좌 - OpenCV 설
localhost:8888/lab/tree/project/iamveryhappy.ipynb
File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ project /
Name
Last Modified
iamveryh... 2 minutes ago
image.csv 3 days ago
intro.mp3 2 minutes ago
iwantslee... 2 days ago
iwantslee... 2 days ago
me.jpg 3 days ago
normal_t... 2 days ago
normal_t... 2 days ago
oangry.jpg 2 days ago
ofear.jpg 2 days ago
ohappy.j... 2 days ago
onormal.j... 19 hours ago
osad.jpg 2 days ago
qqq.jpg 18 hours ago
result.mp3 a minute ago
sad_test.j... 2 days ago
sad_testt... 2 days ago
sh.jpg 3 days ago
sleep.jpg 2 days ago
test.ipynb 2 days ago
test.jpg 11 days ago
Untitled.i... 3 days ago
Untitled1... 3 days ago
iamveryhappy.ipynb
Code
Python 3
[ ]:
Simple 0 0 Python 3 | Idle
Saving completed
Mode: Edit Ln 1, Col 1 iamveryhappy.ipynb
Emotions Flashc....pptx
모두 표시

# 03

## 프로젝트 후기

추후 계획 및 프로젝트 한줄평

# 추후 계획

**정확도 향상**

표정 분석 모델을
더 깊게 이해하고
정확도 향상시켜보기

**모델 추가**

노래 추천을 AI가
사용자의 플레이리스트를
분석하여 할 수 있도록
AI 모델 만들어보기

**실제 사용**

실제로 일상생활에서도
사용할 수 있도록
만들어보기

# 프로젝트 한줄평

문성욱

생각보다
어려웠지만,
생각보다
재미있었다.

김경선

에러가 뜨고 막힐 때는
정말 그만하고 싶을
때도 많았으나 해결을
한 후에 정말 행복했다.

서주완

프로젝트 진행을
하면서 막혔던
부분이 많아
공부의 필요성을
느꼈다.


감사합니다
질문이 있으시다면 편하게 해주세요!