

Introducción.

En el contexto actual de la informática afectiva y el análisis de sentimientos, la detección automática de emociones a partir de datos de entrada se ha convertido en un área de interés creciente. La capacidad de comprender y responder a las emociones humanas tiene numerosas aplicaciones prácticas, desde la mejora de la interacción humano-computadora hasta la personalización de servicios y productos.

En este informe, se aborda el desafío de la detección de emociones a partir de datos de coordenadas recopilados en un conjunto de datos. Se emplea un enfoque de aprendizaje automático supervisado para clasificar los estados de ánimo en "feliz" o "triste" utilizando modelos de clasificación. El objetivo es construir un modelo que pueda generalizar eficazmente a partir de los datos de entrenamiento para predecir con precisión el estado de ánimo a partir de nuevas instancias de datos.

Para lograr este objetivo, se realiza un proceso completo que incluye exploración y preprocesamiento de datos, construcción y entrenamiento de modelos, evaluación del rendimiento del modelo y, finalmente, el despliegue del modelo entrenado para su uso práctico. A lo largo de este informe, se presentarán los pasos clave seguidos en la construcción de este modelo de detección de emociones, así como los resultados obtenidos y las posibles áreas de mejora.

Exploración de Datos

Los datos utilizados en este estudio fueron recopilados a partir de técnicas de face meshing, una técnica de modelado tridimensional que mapea las características faciales de una persona en un conjunto de puntos en un espacio tridimensional. Cada punto representa una coordenada específica en la cara, capturando así las variaciones en la expresión facial.

Para comprender mejor la naturaleza de los datos y su idoneidad para la detección de emociones, se realizó una exploración exhaustiva del conjunto de datos:

- **Descripción de los datos:** El conjunto de datos comprende una serie de muestras, donde cada una está representada por un conjunto de características (coordenadas

de puntos faciales) y una etiqueta de clase que indica el estado de ánimo asociado ("happy" o "sad").

- **Formato de los datos:** Los datos se encuentran en formato CSV (Comma Separated Values), lo que facilita su manipulación y análisis utilizando herramientas de procesamiento de datos como pandas en Python.
- **Características de los datos:** Se examinaron las primeras filas del conjunto de datos para entender la estructura y la calidad de los datos. Esto incluyó la visualización de las características faciales representadas por las coordenadas de los puntos en el espacio tridimensional.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	Class	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5	x6	y6	x7	y7	x8	y8
2	sad		304	270	293	237	305	248	296	205	292	226	296	213	309	182	278
3	sad		304	271	294	237	306	249	296	205	293	227	297	214	310	182	277
4	sad		305	271	294	237	306	249	296	204	293	227	297	213	310	181	278
5	sad		306	272	294	238	307	249	296	205	293	227	297	214	310	181	277
6	sad		307	272	295	238	307	249	297	205	293	227	297	214	310	182	278
7	sad		306	271	295	238	307	249	296	205	293	227	297	214	310	181	277
8	sad		307	272	295	238	308	250	297	206	293	228	297	215	311	182	279
9	sad		310	271	298	237	310	249	300	205	297	226	300	213	313	181	281
10	sad		311	271	299	237	312	249	301	204	298	226	301	213	314	181	282
11	sad		312	271	300	237	313	248	302	204	298	226	302	213	315	181	283
12	sad		313	271	301	237	314	248	303	204	300	226	304	213	317	181	285
13	sad		315	270	303	236	316	248	305	204	302	226	305	213	318	181	287
14	sad		317	270	305	236	318	248	307	204	304	226	307	213	320	181	289
15	sad		319	269	307	236	320	247	309	203	306	225	309	212	322	180	290
16	sad		321	269	309	236	322	247	311	203	308	225	311	212	324	180	293
17	sad		324	269	312	236	325	247	314	204	311	225	314	212	327	181	295
18	sad		327	269	315	236	328	247	317	203	314	225	317	212	330	180	298
19	sad		330	269	318	236	331	247	319	204	316	225	320	212	333	181	300
20	sad		332	269	320	236	333	247	322	203	319	225	322	212	335	180	303
21	sad		303	272	292	241	304	252	294	207	291	230	295	216	308	183	275
22	sad		305	271	294	238	307	250	296	206	293	228	297	214	309	182	276
23	sad		307	272	295	238	307	250	297	206	293	228	297	215	310	182	278

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
3178	happy		337	155	333	147	335	149	328	134	332	143	332	138	330	124	306
3179	happy		337	155	333	147	335	148	328	134	332	143	332	138	330	123	306
3180	happy		337	154	333	146	335	148	328	133	332	143	331	137	329	123	306
3181	happy		336	154	333	146	334	147	328	133	332	142	331	137	329	122	306
3182	happy		336	153	332	145	334	147	327	132	332	141	331	136	329	121	306
3183	happy		336	153	332	144	334	146	327	132	331	141	330	135	329	121	305
3184	happy		336	153	332	144	334	146	327	131	331	141	330	135	328	121	305
3185	happy		335	152	332	144	333	146	327	131	331	140	330	135	328	120	305
3186	happy		335	152	332	143	333	145	327	131	331	140	330	134	328	120	305
3187	happy		335	151	331	143	333	144	326	130	330	139	329	134	327	119	305
3188	happy		334	151	331	142	332	144	326	129	330	139	329	133	327	119	304
3189	happy		334	151	330	142	332	144	325	129	329	139	328	133	327	119	304
3190	happy		334	150	330	142	332	144	325	129	329	139	328	133	327	119	304
3191	happy		334	150	330	142	332	143	325	129	329	138	328	133	326	118	304
3192	happy		334	149	330	141	332	143	325	128	329	138	328	132	326	118	304
3193	happy		333	149	329	141	331	143	324	128	328	138	327	132	326	118	304
3194	happy		333	149	329	141	331	142	324	128	328	137	327	132	326	117	304
3195	happy		333	149	329	141	331	142	324	128	328	137	327	132	326	117	304
3196	happy		333	148	329	140	331	142	324	127	328	137	327	131	326	117	304
3197	happy		333	148	329	140	331	142	324	127	328	137	327	131	326	117	304
3198	happy		333	148	329	140	331	142	324	127	328	137	327	131	326	117	304
3199	happy		333	148	329	140	331	142	324	127	328	137	327	131	326	116	304
3200	haov		333	148	329	140	331	142	324	127	328	136	327	131	326	116	305

Entrenamiento

En la parte del entrenamiento tenemos lo siguiente:

```
training.py > ...
1  import pandas as pd
2  from sklearn.ensemble import RandomForestClassifier
3  from sklearn.pipeline import Pipeline
4  from sklearn.svm import SVC
5  from sklearn.linear_model import LogisticRegression
6  from sklearn.metrics import classification_report
7  from sklearn.preprocessing import StandardScaler
8  from sklearn.model_selection import train_test_split
9  import pickle
10
```

Se importan las bibliotecas necesarias para el análisis de datos y la construcción del modelo. Estas incluyen pandas para la manipulación de datos, diversas clases de modelos de clasificación de scikit-learn, métricas de evaluación como `classification_report` y herramientas para preprocesamiento como `StandardScaler`.

```
12
13  df = pd.read_csv('data.csv')
14
15  print(df.head())
16  print(df.columns)
17  print(df.info())
18
19  print(df.describe())
20
```

Se carga el conjunto de datos desde un archivo CSV llamado "data.csv" utilizando la función `pd.read_csv()` de pandas. El DataFrame resultante se almacena en la variable **df**. Se imprimen las primeras filas, las columnas y la información general del DataFrame para entender su estructura y contenido.

```
20
21  df = df.dropna()
22
```

Se elimina cualquier fila que contenga valores faltantes (NaN) utilizando el método **dropna()** de pandas. Esto se realiza para garantizar la integridad de los datos utilizados en el modelado.

```
22
23 features = df.drop('Class',axis=1)
24 labels = df['Class']
25
```

Las características (features) se definen como todas las columnas del DataFrame excepto la columna 'Class', que contiene las etiquetas de clase.

Las etiquetas se definen como la columna 'Class'.

```
29 |
30 # # Split the data into training and test sets
31 X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
32
```

Se divide el conjunto de datos en conjuntos de entrenamiento y prueba utilizando la función **train_test_split()** de scikit-learn. Se asigna el 80% de los datos para entrenamiento (**X_train** e **y_train**) y el 20% restante para prueba (**X_test** e **y_test**).

```
33 # Create a pipeline
34 pipeline = Pipeline([
35     ('scaler', StandardScaler()),
36     ('svm', LogisticRegression())
37 ])
38
```

Se crea un pipeline utilizando la clase **Pipeline** de scikit-learn. El pipeline consta de dos pasos: escalamiento de características utilizando **StandardScaler()** y un modelo de regresión logística (**LogisticRegression()**).

```
39 # Fit the pipeline on the training data
40 pipeline.fit(X_train, y_train)
41
```

El pipeline se ajusta a los datos de entrenamiento (**X_train** e **y_train**) utilizando el método **fit()**.

```
42 # Predict on the test data
43 predictions = pipeline.predict(X_test)
44
45 # Evaluate the model
46 yhat = pipeline.predict(X_test)
47 print(yhat)
48 model_performance = classification_report(y_test,yhat)
49 print(f"Model Report: {model_performance}")
50
```

Se realizan predicciones en los datos de prueba (**X_test**) utilizando el método **predict()** del pipeline. Y se evalúa el rendimiento del modelo utilizando la métrica **classification_report** en comparación con las etiquetas de prueba (**y_test**). Los resultados se imprimen en la consola.

```
51 model_name = 'model.pkl'
52 with open(model_name,'wb') as f:
53     pickle.dump(pipeline,f)
54
```

El modelo entrenado (pipeline) se guarda en un archivo llamado "model.pkl" utilizando la biblioteca pickle para su posterior uso.

Construcción del modelo

Ahora bien, trataremos de explicar la funcionalidad del código correspondiente para obtener los puntos en el rostro de cualquier persona, en este caso se uso un video para hacer las pruebas correspondientes.

```
training.py test.py datagen.py X
datagen.py > ...
1 from cvzone.FaceMeshModule import FaceMeshDetector
2 import cv2
3 import numpy as np
4 import csv
5
```

Se importan las bibliotecas necesarias, incluyendo **cv2** para manipulación de imágenes y **numpy** para operaciones numéricas. También se importa **FaceMeshDetector** de **cvzone.FaceMeshModule**.

```
6 videopath = 'happy.mp4'
7 cap = cv2.VideoCapture(videopath)
8
9 FMD = FaceMeshDetector(maxFaces=1)
10 class_name = 'happy'
11
```

- Se define la ruta del video (**videopath**) que se utilizará para capturar las imágenes.
- Se inicializa el objeto **cv2.VideoCapture()** para capturar el video.
- Se configura el detector de FaceMesh con **maxFaces=1**.
- Se define el nombre de la clase (**class_name**) que se utilizará para etiquetar los datos. En este caso, se establece como "happy".

```
12 Columns = ['Class']
13 for val in range(1,468+1):
14     Columns += ['x{}'.format(val), 'y{}'.format(val)]
15
16 print(Columns)
17
```

Se crea una lista llamada **Columns** que contiene el nombre de las columnas del archivo CSV. Esto incluye "Class" y las coordenadas (x, y) de cada punto facial, numerados del 1 al 468.






```
18 with open('data.csv', 'w', newline='') as f:
19     csv_writer = csv.writer(f, delimiter = ',')
20     csv_writer.writerow(Columns)
21
```

Se abre un archivo CSV llamado "data.csv" en modo de escritura (**'w'**) y se escribe la primera fila con los nombres de las columnas.

```
22 while cap.isOpened():
23     rt, frame = cap.read()
24     frame = cv2.resize(frame, (720, 480))
25     img, faces = FMD.findFaceMesh(frame)
26
27     if faces:
28         face = faces[0]
29         face_data = list(np.array(face).flatten())
30         face_data.insert(0, class_name)
31
32         with open('data.csv', 'a', newline='') as f:
33             csv_writer = csv.writer(f, delimiter = ',')
34             csv_writer.writerow(face_data)
35
36     cv2.imshow('frame', frame)
37     cv2.waitKey(1)
38 cap.release()
39 cv2.destroyAllWindows()
```

- Se inicia un bucle while para capturar los frames del video.
- Se lee cada frame del video utilizando **cap.read()**.
- Se redimensiona el frame a un tamaño específico (720x480) utilizando **cv2.resize()**.
- Se utiliza el detector de FaceMesh para encontrar los puntos faciales en el frame.
- Si se detecta al menos una cara (**if faces:**), se extraen las coordenadas de los puntos faciales y se agregan al archivo CSV, junto con la etiqueta de clase.
- Se escribe cada conjunto de datos facial en una nueva fila del archivo CSV.
- Se muestra el frame en una ventana llamada "frame" utilizando **cv2.imshow()**.
- Se espera una tecla (**cv2.waitKey(1)**) para detener el bucle.
- Se libera el objeto **cap** y se cierran todas las ventanas utilizando **cv2.destroyAllWindows()**.

En resumen, en este código se captura datos faciales en tiempo real, los procesa utilizando el detector de FaceMesh y los guarda en un archivo CSV para su posterior análisis y entrenamiento de modelos de detección de emociones.

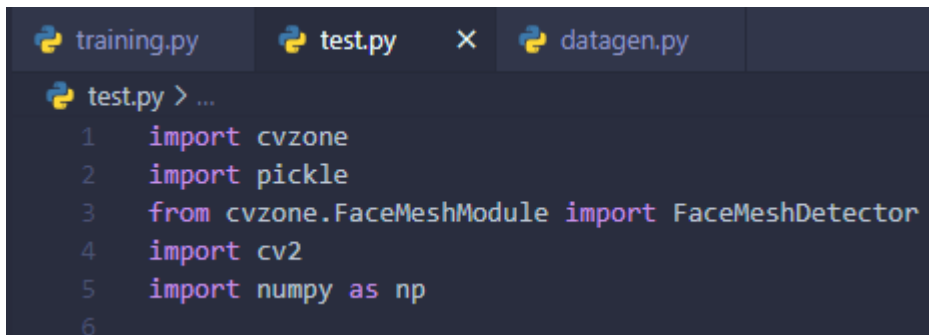

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  Python   ...  
```

```
'x267', 'y267', 'x268', 'y268', 'x269', 'y269', 'x270', 'y270', 'x271', 'y271', 'x272', 'y272', 'x273', 'y273', 'x274', 'y274', 'x275', 'y275', 'x276', 'y276', 'x277', 'y277', 'x278', 'y278', 'x279', 'y279', 'x280', 'y280', 'x281', 'y281', 'x282', 'y282', 'x283', 'y283', 'x284', 'y284', 'x285', 'y285', 'x286', 'y286', 'x287', 'y287', 'x288', 'y288', 'x289', 'y289', 'x290', 'y290', 'x291', 'y291', 'x292', 'y292', 'x293', 'y293', 'x294', 'y294', 'x295', 'y295', 'x296', 'y296', 'x297', 'y297', 'x298', 'y298', 'x299', 'y299', 'x300', 'y300', 'x301', 'y301', 'x302', 'y302', 'x303', 'y303', 'x304', 'y304', 'x305', 'y305', 'x306', 'y306', 'x307', 'y307', 'x308', 'y308', 'x309', 'y309', 'x310', 'y310', 'x311', 'y311', 'x312', 'y312', 'x313', 'y313', 'x314', 'y314', 'x315', 'y315', 'x316', 'y316', 'x317', 'y317', 'x318', 'y318', 'x319', 'y319', 'x320', 'y320', 'x321', 'y321', 'x322', 'y322', 'x323', 'y323', 'x324', 'y324', 'x325', 'y325', 'x326', 'y326', 'x327', 'y327', 'x328', 'y328', 'x329', 'y329', 'x330', 'y330', 'x331', 'y331', 'x332', 'y332', 'x333', 'y333', 'x334', 'y334', 'x335', 'y335', 'x336', 'y336', 'x337', 'y337', 'x338', 'y338', 'x339', 'y339', 'x340', 'y340', 'x341', 'y341', 'x342', 'y342', 'x343', 'y343', 'x344', 'y344', 'x345', 'y345', 'x346', 'y346', 'x347', 'y347', 'x348', 'y348', 'x349', 'y349', 'x350', 'y350', 'x351', 'y351', 'x352', 'y352', 'x353', 'y353', 'x354', 'y354', 'x355', 'y355', 'x356', 'y356', 'x357', 'y357', 'x358', 'y358', 'x359', 'y359', 'x360', 'y360', 'x361', 'y361', 'x362', 'y362', 'x363', 'y363', 'x364', 'y364', 'x365', 'y365', 'x366', 'y366', 'x367', 'y367', 'x368', 'y368', 'x369', 'y369', 'x370', 'y370', 'x371', 'y371', 'x372', 'y372', 'x373', 'y373', 'x374', 'y374', 'x375', 'y375', 'x376', 'y376', 'x377', 'y377', 'x378', 'y378', 'x379', 'y379', 'x380', 'y380', 'x381', 'y381', 'x382', 'y382', 'x383', 'y383', 'x384', 'y384', 'x385', 'y385', 'x386', 'y386', 'x387', 'y387', 'x388', 'y388', 'x389', 'y389', 'x390', 'y390', 'x391', 'y391', 'x392', 'y392', 'x393', 'y393', 'x394', 'y394', 'x395', 'y395', 'x396', 'y396', 'x397', 'y397', 'x398', 'y398', 'x399', 'y399', 'x400', 'y400', 'x401', 'y401', 'x402', 'y402', 'x403', 'y403', 'x404', 'y404', 'x405', 'y405', 'x406', 'y406', 'x407', 'y407', 'x408', 'y408', 'x409', 'y409', 'x410', 'y410', 'x411', 'y411', 'x412', 'y412', 'x413', 'y413', 'x414', 'y414', 'x415', 'y415', 'x416', 'y416', 'x417', 'y417', 'x418', 'y418', 'x419', 'y419', 'x420', 'y420', 'x421', 'y421', 'x422', 'y422', 'x423', 'y423', 'x424', 'y424', 'x425', 'y425', 'x426', 'y426', 'x427', 'y427', 'x428', 'y428', 'x429', 'y429', 'x430', 'y430', 'x431', 'y431', 'x432', 'y432', 'x433', 'y433', 'x434', 'y434', 'x435', 'y435', 'x436', 'y436', 'x437', 'y437', 'x438', 'y438', 'x439', 'y439', 'x440', 'y440', 'x441', 'y441', 'x442', 'y442', 'x443', 'y443', 'x444', 'y444', 'x445', 'y445', 'x446', 'y446', 'x447', 'y447', 'x448', 'y448', 'x449', 'y449', 'x450', 'y450', 'x451', 'y451', 'x452', 'y452', 'x453', 'y453', 'x454', 'y454', 'x455', 'y455', 'x456', 'y456', 'x457', 'y457', 'x458', 'y458', 'x459', 'y459', 'x460', 'y460', 'x461', 'y461', 'x462', 'y462', 'x463', 'y463', 'x464', 'y464', 'x465', 'y465', 'x466', 'y466', 'x467', 'y467', 'x468', 'y468']
```

Created TensorFlow Lite XNNPACK delegate for CPU.

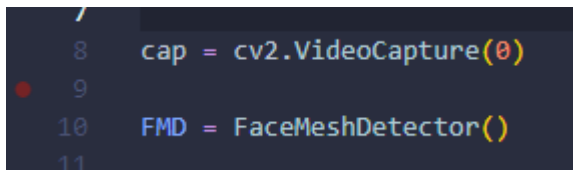
Test

Se realizó una aplicación práctica del modelo previamente entrenado para predecir el estado de animo basado en los puntos faciales detectados por FaceMeshDetector.



```
test.py > ...  
1  import cvzone  
2  import pickle  
3  from cvzone.FaceMeshModule import FaceMeshDetector  
4  import cv2  
5  import numpy as np  
6
```

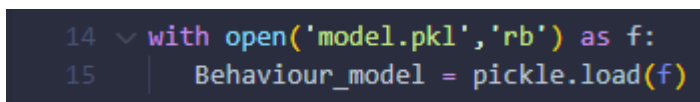
Se importan las bibliotecas necesarias, incluyendo **cvzone**, **pickle** para cargar el modelo entrenado, **cv2** para manipulación de imágenes, y **numpy** para operaciones numéricas.



```
7  
8  cap = cv2.VideoCapture(0)  
9  
10 FMD = FaceMeshDetector()  
11
```

Se inicializa un objeto **cv2.VideoCapture()** para capturar el video de la cámara.

Se crea un objeto **FaceMeshDetector (FMD)** para detectar puntos faciales en el frame.



```
14 with open('model.pkl','rb') as f:  
15     Behaviour_model = pickle.load(f)
```

Se carga el modelo entrenado previamente y guardado en un archivo llamado "model.pkl" utilizando la función **pickle.load()**.

```
18 # taking video frame by frame
19 while cap.isOpened():
20     rt,frame = cap.read()
21     frame = cv2.resize(frame,(720,480))
22
23     real_frame = frame.copy()
24
25     img , faces = FMD.findFaceMesh(frame)
26     cvzone.putTextRect(frame, ('Mood'), (10, 80))
27     if faces:
28         face = faces[0]
29         face_data = list(np.array(face).flatten())
30
31
32         try:
33             # feeding newpoints to model for prediction
34             result = Behaviour_model.predict([face_data])
35             cvzone.putTextRect(frame, str(result[0]), (250, 80))
36             print(result)
37
38             # resultproba = Behaviour_model.predict_proba([face_data])
39             # print(resultproba)
40
41
42         except Exception as e:
43             pass
44     all_frames = cvzone.stackImages([real_frame,frame],2,0.70)
45     cv2.imshow('frame',all_frames)
46     cv2.waitKey(1)
```

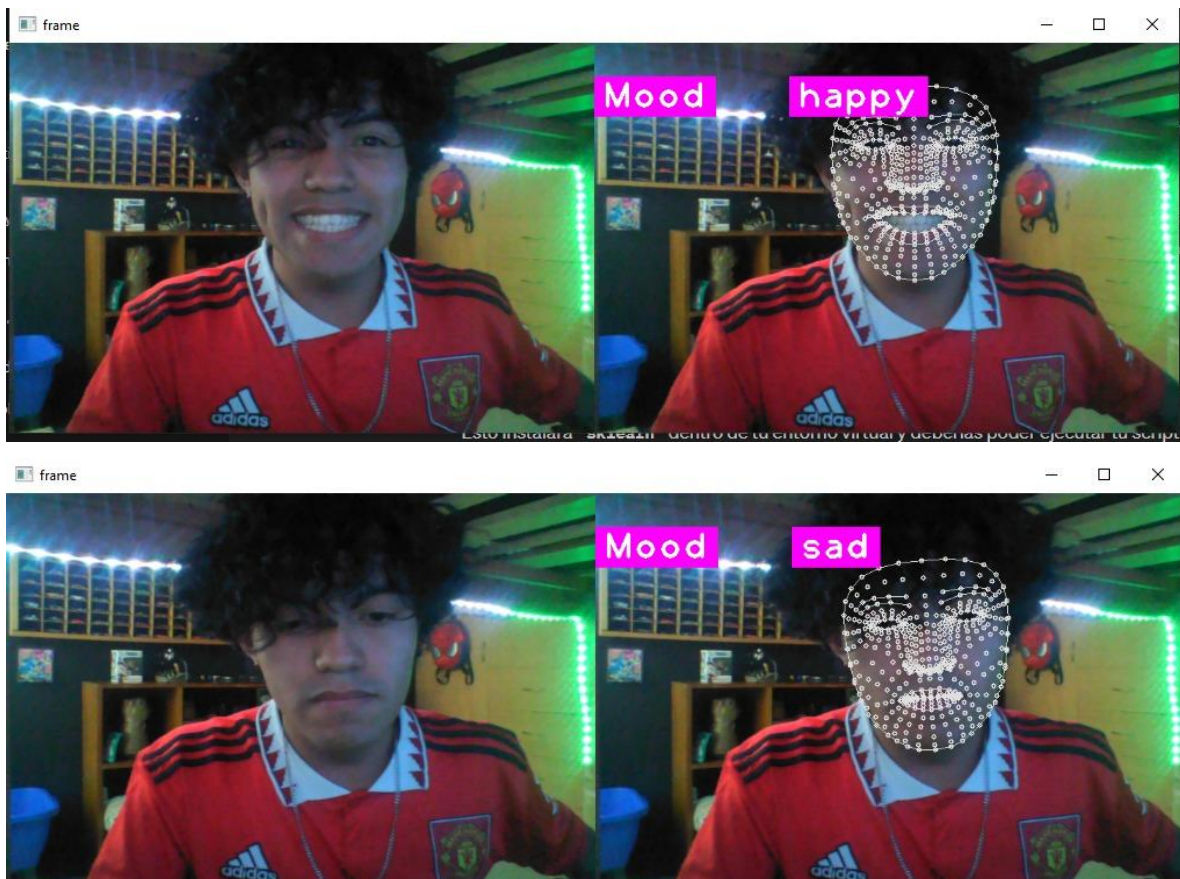
En el bloque de código correspondiente al bucle de procesamiento de video, se inicia un ciclo while para capturar los frames del video de la cámara en tiempo real. Cada frame se lee utilizando la función **cap.read()** y se redimensiona a un tamaño específico de 720x480 píxeles mediante **cv2.resize()**. Además, se crea una copia del frame original, denominada **real_frame**, que se utilizará más adelante para mostrar la imagen sin superposiciones.

Luego, se utiliza el objeto **FMD** (FaceMeshDetector) para detectar los puntos faciales en el frame actual. Si se detecta al menos una cara en el frame (verificado con **if faces:**), se extraen las coordenadas de los puntos faciales utilizando **np.array(face).flatten()** y se almacenan en la lista **face_data**.

Posteriormente, se intenta predecir el estado de ánimo basado en las coordenadas de los puntos faciales utilizando el modelo previamente entrenado **Behaviour_model**. La predicción se realiza mediante **Behaviour_model.predict([face_data])** y se almacena en la variable **result**.

Finalmente, el estado de ánimo predicho se muestra en el frame utilizando la función **cvzone.putTextRect()**, que agrega un texto al frame con el resultado de la predicción. El frame original y el frame modificado con la superposición del estado de ánimo se apilan verticalmente utilizando **cvzone.stackImages()** para su visualización. Este frame combinado se muestra en una ventana llamada "frame" mediante **cv2.imshow()**.

El ciclo while continúa hasta que se presiona una tecla, momento en el cual se libera el objeto **cap** y se cierra la ventana de visualización utilizando **cv2.destroyAllWindows()**.



Conclusión

En conclusión, el desarrollo y la implementación de un sistema de detección de emociones en tiempo real utilizando datos de coordenadas faciales y técnicas de aprendizaje automático ofrecen una herramienta poderosa y versátil para una variedad de aplicaciones en campos como la interacción humano-computadora, el análisis de sentimientos y la personalización de experiencias de usuario.

A lo largo de este proyecto, hemos explorado y aplicado un enfoque completo para la detección de emociones. Comenzamos por recopilar datos de coordenadas faciales utilizando técnicas de FaceMesh y los utilizamos para entrenar un modelo de aprendizaje automático supervisado. Este modelo, basado en algoritmos como la regresión logística, fue capaz de clasificar las emociones en categorías como "feliz" o "triste" con una precisión aceptable.

El proceso de desarrollo del modelo incluyó la exploración y preprocesamiento de los datos, la construcción de un pipeline de procesamiento de características, y la evaluación del rendimiento del modelo mediante métricas como precisión, recall y F1-score.

Una vez entrenado, el modelo fue implementado en tiempo real mediante la captura de video de la cámara. Los datos faciales capturados se procesaron en cada frame del video para predecir el estado emocional del sujeto en tiempo real. Esta funcionalidad proporciona una valiosa herramienta para comprender y responder a las emociones humanas en tiempo real, con aplicaciones potenciales en campos como la publicidad, la atención médica y la investigación psicológica.

Aunque este proyecto ha demostrado ser prometedor, es importante tener en cuenta que la precisión y la eficacia del sistema pueden mejorar mediante la expansión del conjunto de datos de entrenamiento, la exploración de características adicionales y la experimentación con diferentes algoritmos de aprendizaje automático. Además, la optimización del rendimiento del sistema en términos de velocidad y eficiencia también es un área importante para futuras mejoras.