Reporte Labelled Faces in the Wild (LFW)

Se considera el conjunto de datos Labelled Faces in the Wild (LFW), que consiste en fotografías de rostros recolectados de internet y contenido en sklearn. Algunos rostros identificados, tienen varias fotos incluÍdas en el dataset. Vamos a considerar solo aquellas personas que tienen al menos 70 fotografías de su rostro, tambien, vamos a considerar el tamaño original de la imágen (125×94)

```
from sklearn.datasets import fetch lfw people
lfw people = fetch lfw people( min faces per person =70, resize=1)
```

Esto resulta en 1288 imágenes que pertenecen a alguna de las etiquetas:

```
>>> for name in lfw people.target names:
>>> print(name)
Ariel Sharon
Colin Powell
Donald Rumsfeld
George W Bush
Gerhard Schroeder
Hugo Chavez
Tony Blair
```

Una muestra de las imágenes puede verse en la Figura2.
a) Separa un conjunto de entrenamiento (80 %) y prueba (puedes usar la función `train_test_split` de `sklearn.model_selection`, por ejemplo:

```
from sklearn.model selection import train test split
X train , X test , y train , y test , names train ,
names test = train test split(X, y, target names[y],
test size =0.2, random state =42)
```

donde antes, tuviste que declarar `X,y,target_names` (ve la documentación de `fetch_lfw_people`). Obtén las eigenfaces del conjunto de entrenamiento. Visualiza los scores de los primeros dos componentes principales ¿Encuentras patrones interesantes?
b) Proyecta los datos de prueba en los componentes principales y verifica si se ubican en su individuo correspondiente al graficarlos en los primeros dos componentes principales.
c) Usa el método del vecino más cercano para identificar a un sujeto de prueba en las imágenes de entrenamiento. Usa la distancia euclidiana en el espacio de los p componentes principales. Decide qué valor de p usar. El objetivo es obtener algo como lo que se muestra en la figura: ¿Puedes identificar correctamente a los sujetos usando éste criterio? ¿Qué tanto influye el valor de p?
d) Considera una(s) imágen(es) que no están la base de datos ¿Qué se te ocurre para prevenir casos como los que muestran en la Figura 4?

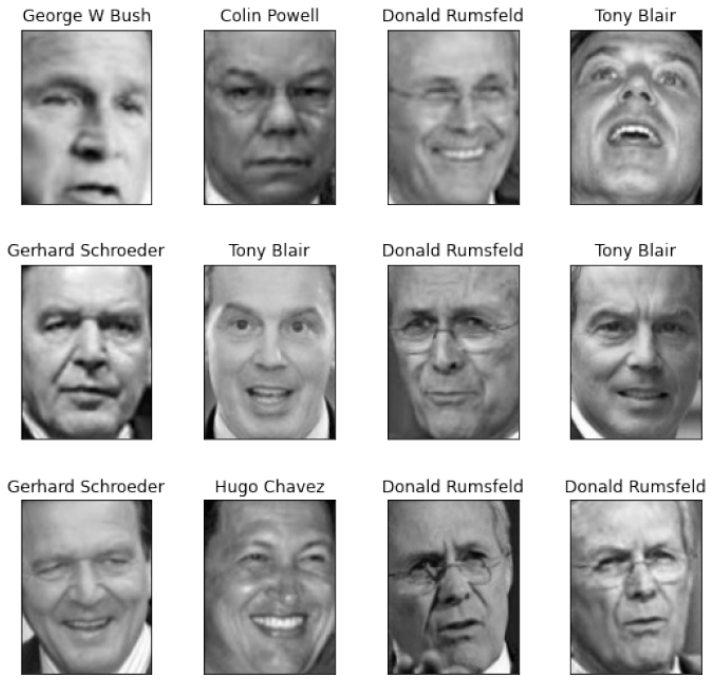

Figura 0.1: Ejemplo de los rostros del dataset LFW

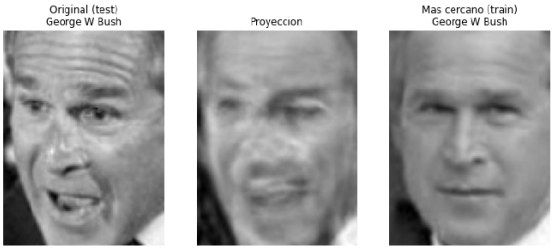

Figura 0.2: Identificación de un individuo de prueba usando el vecino más cercano en el espacio de los primeros p PC.

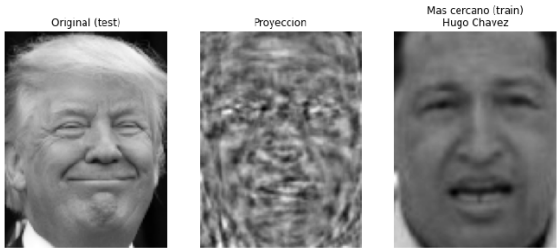

Figura 0.3: Identificación de un individuo de prueba usando el vecino más cercano en el espacio de los primeros p PC.

R E S P U E S T A:
Para empezar a responder este ejercicio nos centraremos en resolver inciso por inciso, para el inciso A, la idea que nos pide es primero separar nuestro conjunto de entrenamiento del de prueba y además usar la función train test split para ello.
Iniciaremos presentando todas las librerias que se usaron para resolver este ejercicio.

```
import matplotlib.pyplot as plt
from scipy.stats import loguniform
from sklearn.datasets import fetch_lfw_people
from sklearn.decomposition import PCA
from sklearn.metrics import ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, pairwise_distances_argmin_min
```

Después de las librerías completas, ahora tendremos que cargar correctamente nuestros datos con las reestricciones que nos solicita el problema. Además de separar nuestros datos en datos de entrenamiento y de prueba con el método que se nos indica.

```
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people ( min_faces_per_person =70, resize =1)
n_samples, h, w = lfw_people.images.shape
X = lfw_people.data
n_features = X.shape[1]
y = lfw_people.target
target_names = lfw_people.target_names
n_classes = target_names.shape[0]
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test,_names_train, names_test = train_test_split (X, y, target_names[y],
                                                    test_size = 0.2, random_state =42)
```

Para la parte que nos interesa en este inciso que es obtener las eigenfaces del conjunto de entrenamiento y después poder visualizar los scores de los primeros dos componentes principales es necesario realizar un pca.
Este fragmento de código realiza la reducción de dimensionalidad utilizando el análisis de componentes principales (PCA), primero definimos los n componentes se hace el ajuste pca y después se calculán lo eigenfaces.

```
n_components = 60
pca = PCA(n_components=n_components, svd_solver="randomized", whiten=True).fit(X_train)
eigenfaces = pca.components_.reshape((n_components, h, w))
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

Entonces los eigenfaces se ven de la siguiente forma:

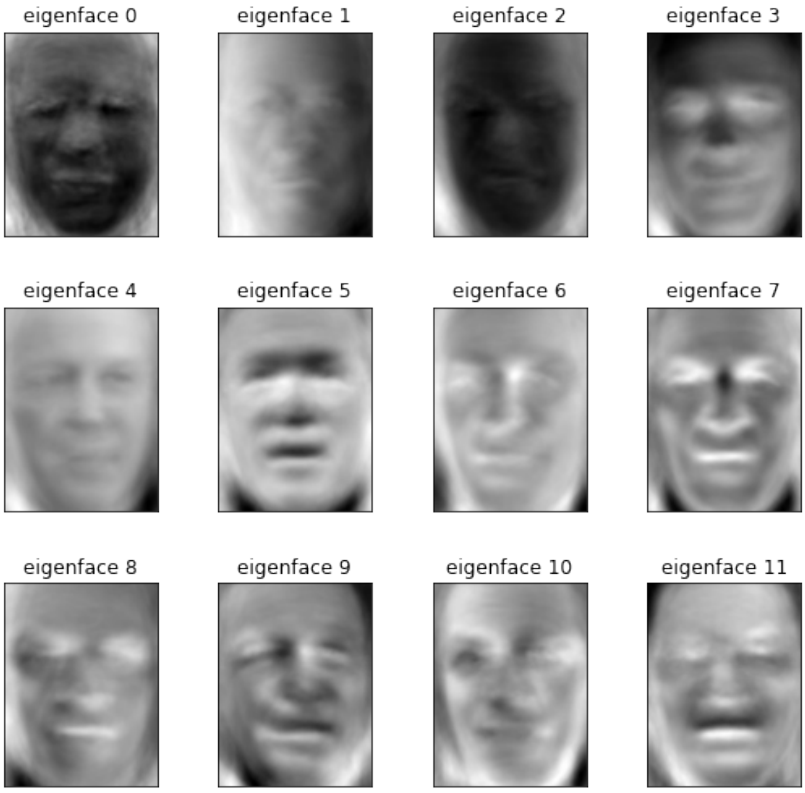

Figura 0.4: Eigenfaces del conjunto de entrenamiento

Además en esta parte podemos de igual forma mostrar la visualización de los scores. Para una mejor visualización podemos usar el siguiente tipo de gráfico.

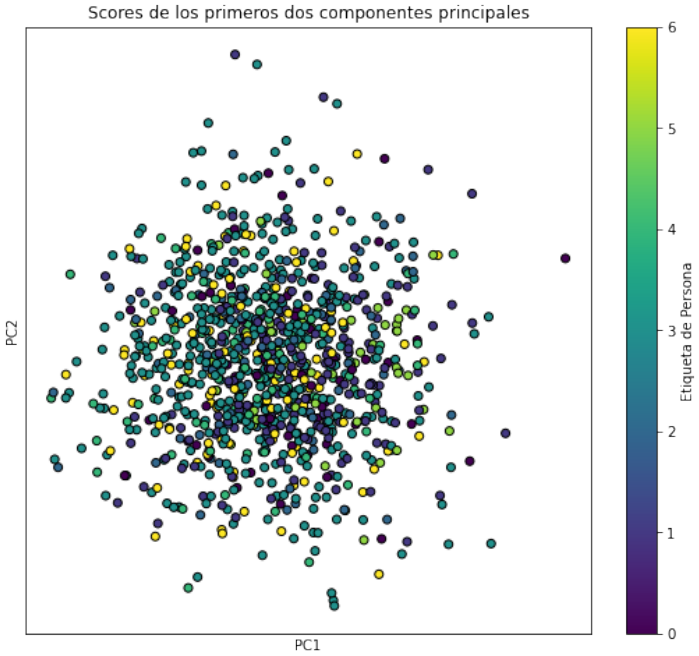

Figura 0.5: Visualización los scores de los primeros dos componentes principales del conjunto de entrenamiento

Al vere ste gráfico anterior podemos observar que en algunas imágenes si se logra ver que

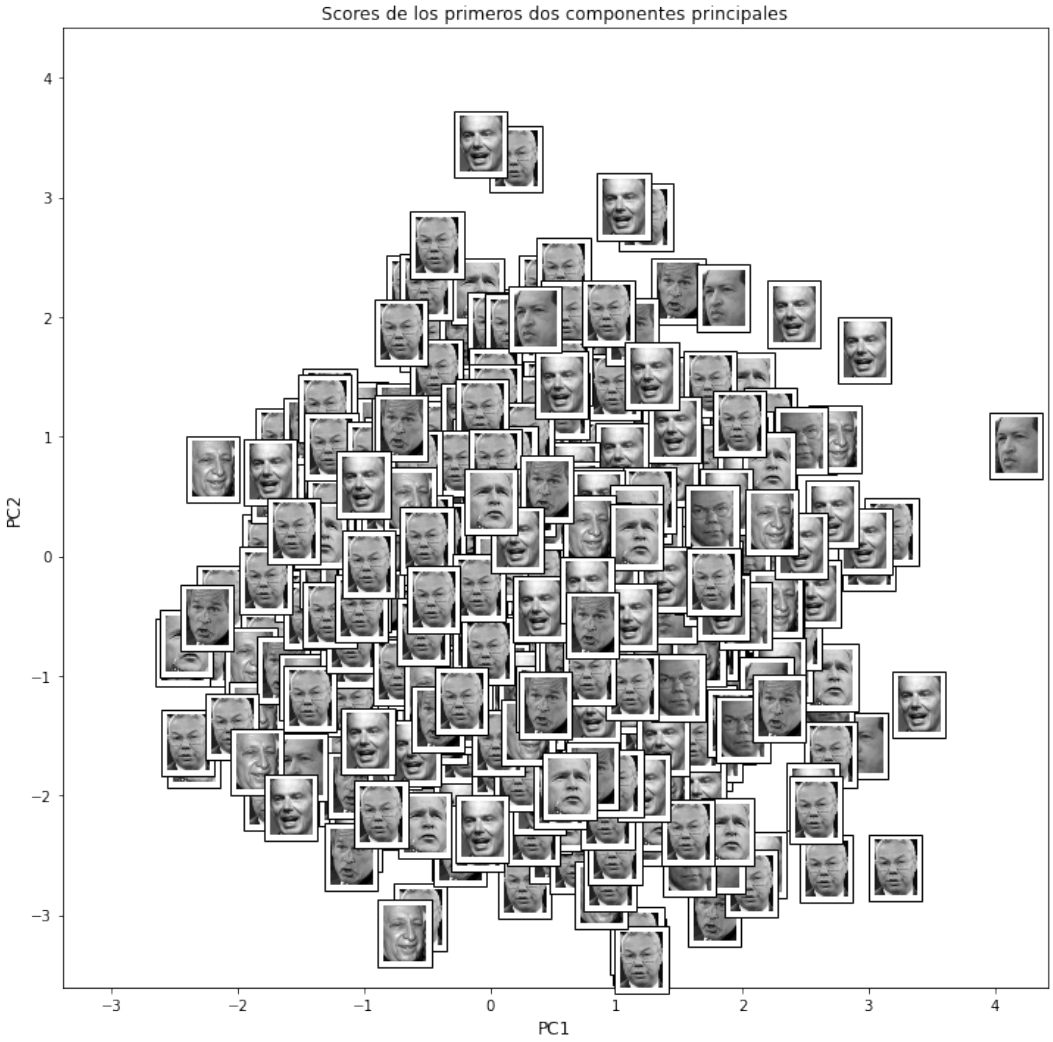

Figura 0.6: Visualización los scores de los primeros dos componentes principales del conjunto de entrenamiento con imágenes

están muy cercanas a otras que son la misma persona, pero puede inclusive haber imagenes de la misma persona muy lejana a otras. Muchas de estos datos están muy centrados por lo que resulta uy dificil porque encontrar patrones ahorita si queremos hacerlo visualmente, deberemos recurrir a otros métodos como el que se nos solicita en el inciso c de esté problema. Para el siguiente inciso, donde nos solicita proyectar los datos de prueba en los componentes principales, utilizaremos las mismas funciones que se usaron para hacer los gráficos anteriores, por lo que tenemos:

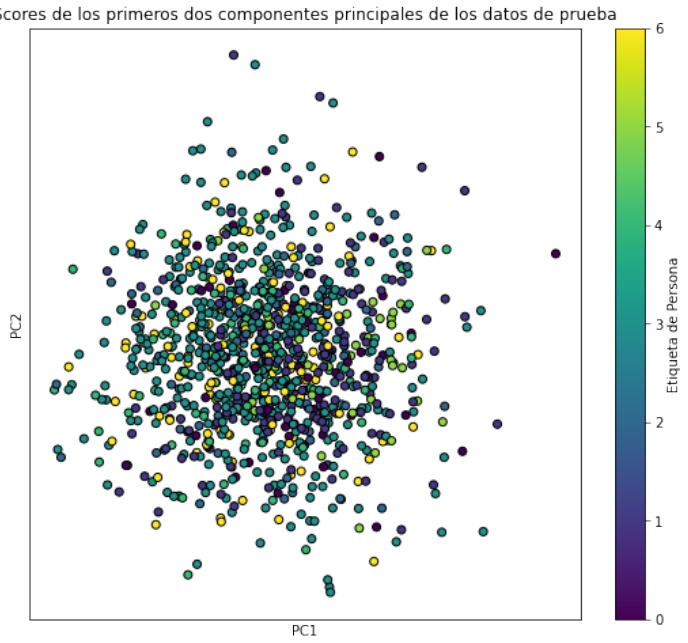

Figura 0.7: Visualización los scores de los primeros dos componentes principales del conjunto de prueba

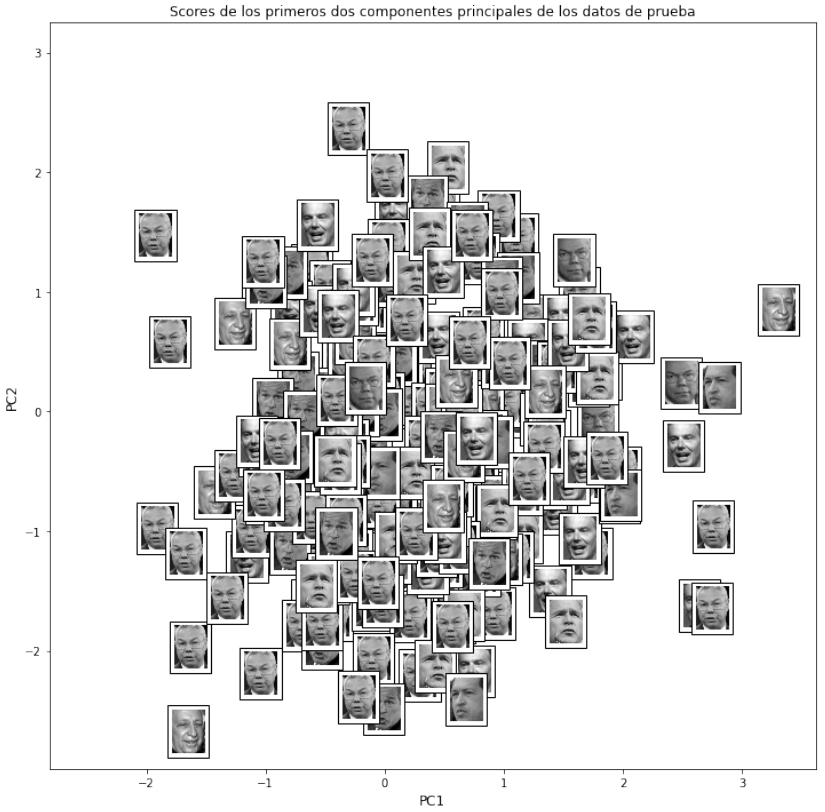

Figura 0.8: Visualización los scores de los primeros dos componentes principales del conjunto de prueba con las imágenes

En estas graficas, más en esta que esta representada por las imágenes, podemos decir que si

existe mas cercania entre los individuos podemos decir que estan mejores agrupados entre si. Ahora, bien nos toca trabajar con usar el método del vecino más cercano para identificar a un sujeto de prueba en las imágenes de entrenamiento entonces, primero tenemos que definir cúal es nuestro valor de p que mejor nos convenga para ello podemos apoyarnos de las siguientes graficas.

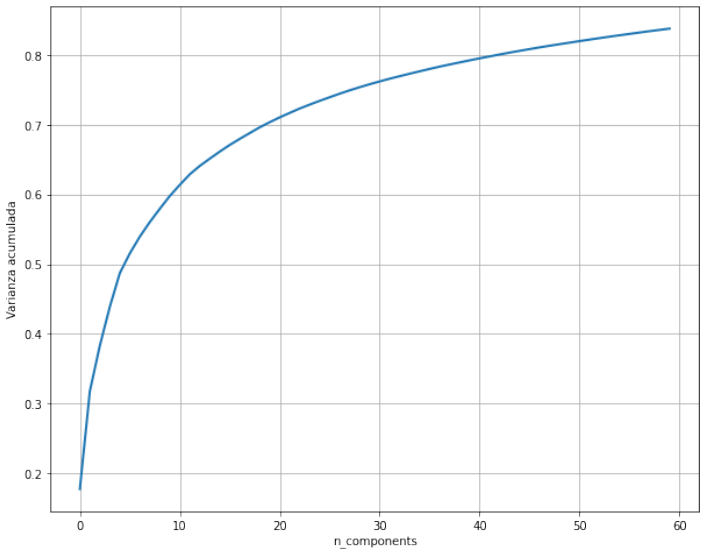

Figura 0.9: Visualización los scores de los primeros dos componentes principales del conjunto de prueba con las imágenes

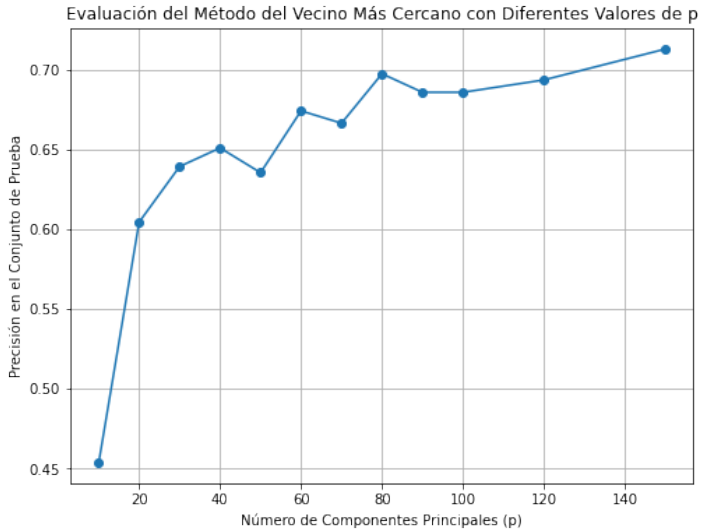

Figura 0.10: Visualización los scores de los primeros dos componentes principales del conjunto de prueba con las imágenes

En las dos básicamente lo que queremos lograr ver es que valor de p, que son el número de componentes principales vamos a usar para aplicar nuestro método del vecino más cercano, y graficamente podemos ver que un valor p=50, expresa la mayor varianza de los datos más de un 0.8 y tiene una buena precisión.
Ahora para aplicar el método con nuestra ya conocida p nos apoyamos del siguiente código:

```
# Elegir p
optimal_p = 50

# Entrenar el clasificador
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_pca, y_train)
random_index = np.random.randint(0, len(X_test))
sample_test_image = X_test[random_index]
sample_test_label = y_test[random_index]
sample_test_name = names_test[random_index]
# Proyectar la muestra
sample_test_pca = pca.transform([sample_test_image])

# Encontrar el vecino ms cercano
nearest_neighbor_index = pairwise_distances_argmin_min(sample_test_pca, X_train_pca)[0][0]
nearest_neighbor_image = X_train[nearest_neighbor_index]
nearest_neighbor_label = y_train[nearest_neighbor_index]
nearest_neighbor_name = _names_train[nearest_neighbor_index]
```

Que cómo hemos visto en el numero de componentes que usamos es un buen indicador y podemos decir que el modelo trabaja de buena forma. Aquí se proporcionan una de las pruebas hechas por el modelo.

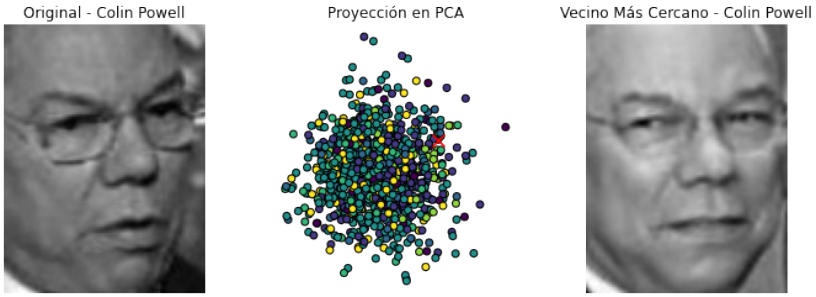

Figura 0.11: Visualización de Colin Powell entre los datos, su proyección y su vecino más cercano tomando en cuenta la distancia euclidiana

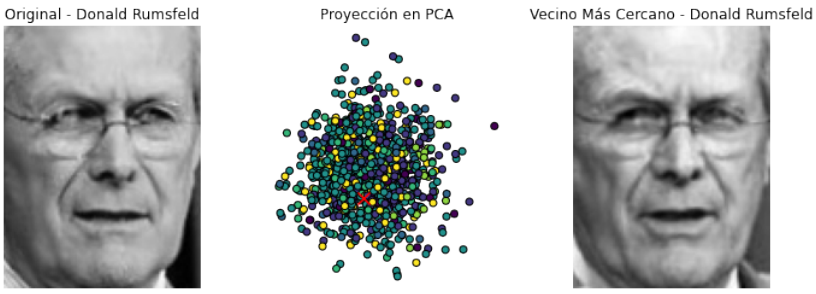

Figura 0.12: Visualización de Donald Rumsfeld entre los datos, su proyección y su vecino más cercano tomando en cuenta la distancia euclidiana

Por lo que podemos concluir que si hemos encontrado correctamente a las personas cone ste método, ya que se realizaron las pruebas que resultaron satisfactorias. Sobre la decisión de que valor de p tomar, hicimos análisis de los datos pero la influencia de p que representa la cantidad de componentes principales utilizados en el método de PCA, puede tener un

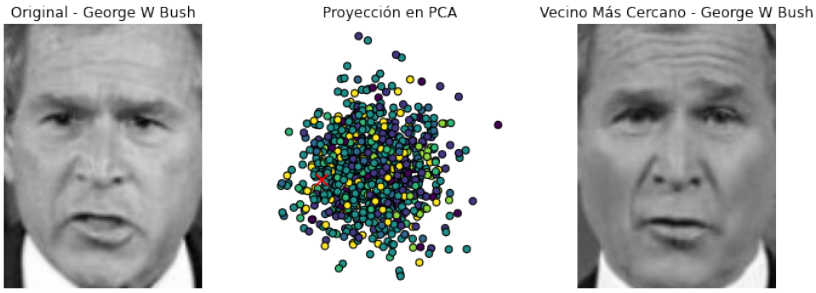

Figura 0.13: Visualización de George W Bush entre los datos, su proyección y su vecino más cercano tomando en cuenta la distancia euclidian

impacto significativo en el rendimiento del método del vecino más cercano y, por lo tanto, en la capacidad de reconocimiento de caras. Aquí hay algunas consideraciones sobre cómo el valor de p puede influir:
Información retenida: A medida que aumentas el valor de p, retienes más información de las características originales de las imágenes. Sin embargo, demasiadas dimensiones pueden llevar a un sobreajuste y a la incorporación de ruido.
Dimensionalidad del espacio de características: Un valor bajo de p puede reducir la dimensionalidad del espacio de características, pero podría perder información relevante para el reconocimiento facial. Un valor alto de p puede llevar a un espacio de características de alta dimensionalidad.
Como comentario final para está parte, respondiendo a la pregunta que se hace respecto a evitar problemas con imágenes que no están en nuestra base de datos sea incorrectamente representada como una de las personas de la base, se pueden implementar un umbral de Similaridad, es decir, establecer un umbral de similitud al comparar la imagen desconocida con las imágenes de la base. Si la similitud está por debajo de un cierto umbral, la imagen se considera como "no reconocida". Esto ayuda a controlar qué tan cercana debe ser la coincidencia para ser aceptada. De otra forma podemos utilizar técnicas de detección de anomalías para identificar si la imagen desconocida se desvía significativamente de las características típicas de las imágenes en la base de datos.