Introduction

In this project, we aim to develop a robust image processing pipeline that extracts features from a dataset of images. The primary focus is on leveraging deep learning techniques to analyze and understand visual data, specifically in a grayscale format. Objectives

Feature Extraction: Utilizing the Pillow library to load and preprocess images, we resize and convert them into a standardized format suitable for further analysis.

Data Preparation: Organizing image data into a structured array, facilitating the application of machine learning algorithms.

Model Training: (If applicable) Preparing to train a model on the extracted features to achieve tasks such as classification, detection, or segmentation.

Evaluation: Assessing the performance of the model and refining the approach based on the results.

Importance

Image processing and feature extraction play a crucial role in various applications, including computer vision, healthcare imaging, and autonomous systems. This project aims to provide foundational insights and methods that can be scaled and adapted for more complex tasks in the field.

By: Kanishk Karam

Import Modules

In [2]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from tqdm.notebook import tqdm
warnings.filterwarnings('ignore')
%matplotlib inline

import tensorflow as tf
from keras.preprocessing.image import load_img
from keras.models import Sequential, Model
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D, Input
```

Load the Dataset

In [3]:
```
BASE_DIR = 'UTKFace'
```

In [4]:
```
# labels - age, gender, ethnicity
image_paths = []
age_labels = []
gender_labels = []

for filename in tqdm(os.listdir(BASE_DIR)):
    image_path = os.path.join(BASE_DIR, filename)
    temp = filename.split('_')
    age = int(temp[0])
    gender = int(temp[1])
    image_paths.append(image_path)
    age_labels.append(age)
    gender_labels.append(gender)
```

```
0%|          | 0/23708 [00:00<?, ?it/s]
```

In [5]:
```
# convert to dataframe
df = pd.DataFrame()
df['image'], df['age'], df['gender'] = image_paths, age_labels, gender_labels
df.head()
```

Out[5]:

	image	age	gender
0	UTKFace\100_0_0_20170112213500903.jpg.chip.jpg	100	0
1	UTKFace\100_0_0_20170112215240346.jpg.chip.jpg	100	0
2	UTKFace\100_1_0_20170110183726390.jpg.chip.jpg	100	1
3	UTKFace\100_1_0_20170112213001988.jpg.chip.jpg	100	1
4	UTKFace\100_1_0_20170112213303693.jpg.chip.jpg	100	1

In [6]:
```
# map labels for gender
gender_dict = {0:'Male', 1:'Female'}
```

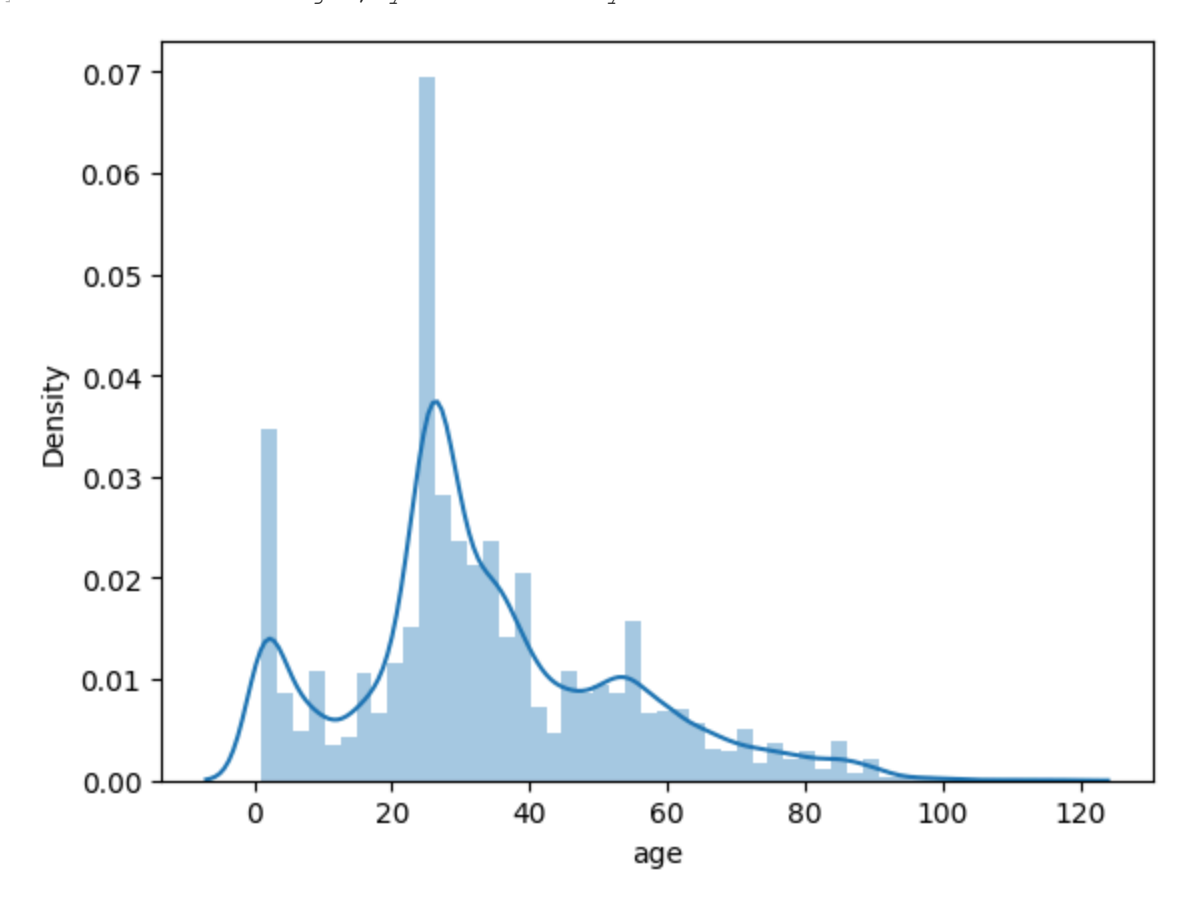
Exploratory Data Analysis

In [7]:
```
from PIL import Image
img = Image.open(df['image'][0])
plt.axis('off')
plt.imshow(img)
```

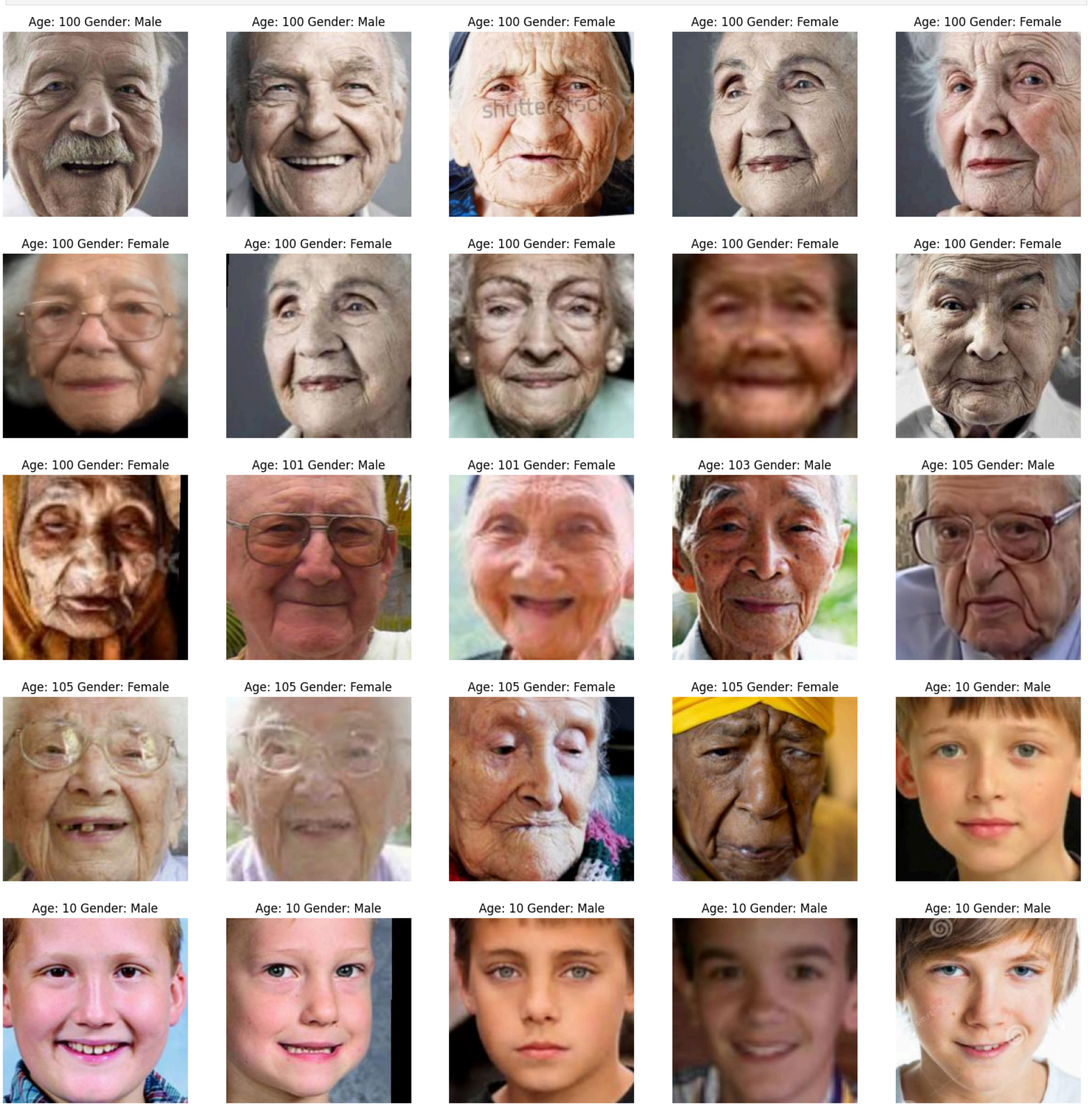
Out[7]: <matplotlib.image.AxesImage at 0x162fcb82270>

In [8]:
```
sns.distplot(df['age'])
```

<Axes: xlabel='age', ylabel='Density'>



In [9]:
```
# to display grid of images
plt.figure(figsize=(20, 20))
files = df.iloc[0:25]

for index, file, age, gender in files.itertuples():
    plt.subplot(5, 5, index+1)
    img = load_img(file)
    img = np.array(img)
    plt.imshow(img)
    plt.title(f"Age: {age} Gender: {gender_dict[gender]}")
    plt.axis('off')
```



Feature Extraction

In [25]:
```
def extract_features(images):
    features = []
    for image in tqdm(images):
        img = load_img(image, color_mode="grayscale")
        img = img.resize((128, 128), Image.LANCZOS)
        img = np.array(img)
        features.append(img)

    features = np.array(features)
    # ignore this step if using RGB
    features = features.reshape(len(features), 128, 128, 1)
    return features
```

In [26]:
```
X = extract_features(df['image'])
```

```
0%|          | 0/23708 [00:00<?, ?it/s]
```

```
X.shape
```

```
(23708, 128, 128, 1)
```

```
# normalize the images
X = X/255.0
```

```
y_gender = np.array(df['gender'])
y_age = np.array(df['age'])
```

```
input_shape = (128, 128, 1)
```

Model Creation

```
inputs = Input((input_shape))
# convolutional layers
conv_1 = Conv2D(32, kernel_size=(3, 3), activation='relu') (inputs)
maxp_1 = MaxPooling2D(pool_size=(2, 2)) (conv_1)
conv_2 = Conv2D(64, kernel_size=(3, 3), activation='relu') (maxp_1)
maxp_2 = MaxPooling2D(pool_size=(2, 2)) (conv_2)
conv_3 = Conv2D(128, kernel_size=(3, 3), activation='relu') (maxp_2)
maxp_3 = MaxPooling2D(pool_size=(2, 2)) (conv_3)
conv_4 = Conv2D(256, kernel_size=(3, 3), activation='relu') (maxp_3)
maxp_4 = MaxPooling2D(pool_size=(2, 2)) (conv_4)

flatten = Flatten() (maxp_4)

# fully connected layers
dense_1 = Dense(256, activation='relu') (flatten)
dense_2 = Dense(256, activation='relu') (flatten)

dropout_1 = Dropout(0.4) (dense_1)
dropout_2 = Dropout(0.4) (dense_2)

output_1 = Dense(1, activation='sigmoid', name='gender_out') (dropout_1)
output_2 = Dense(1, activation='relu', name='age_out') (dropout_2)

model = Model(inputs=[inputs], outputs=[output_1, output_2])

model.compile(loss=['binary_crossentropy', 'mae'], optimizer='adam', metrics=['accuracy', 'mae'])
```

```
# plot the model
from tensorflow.keras.utils import plot_model
plot_model(model)
```

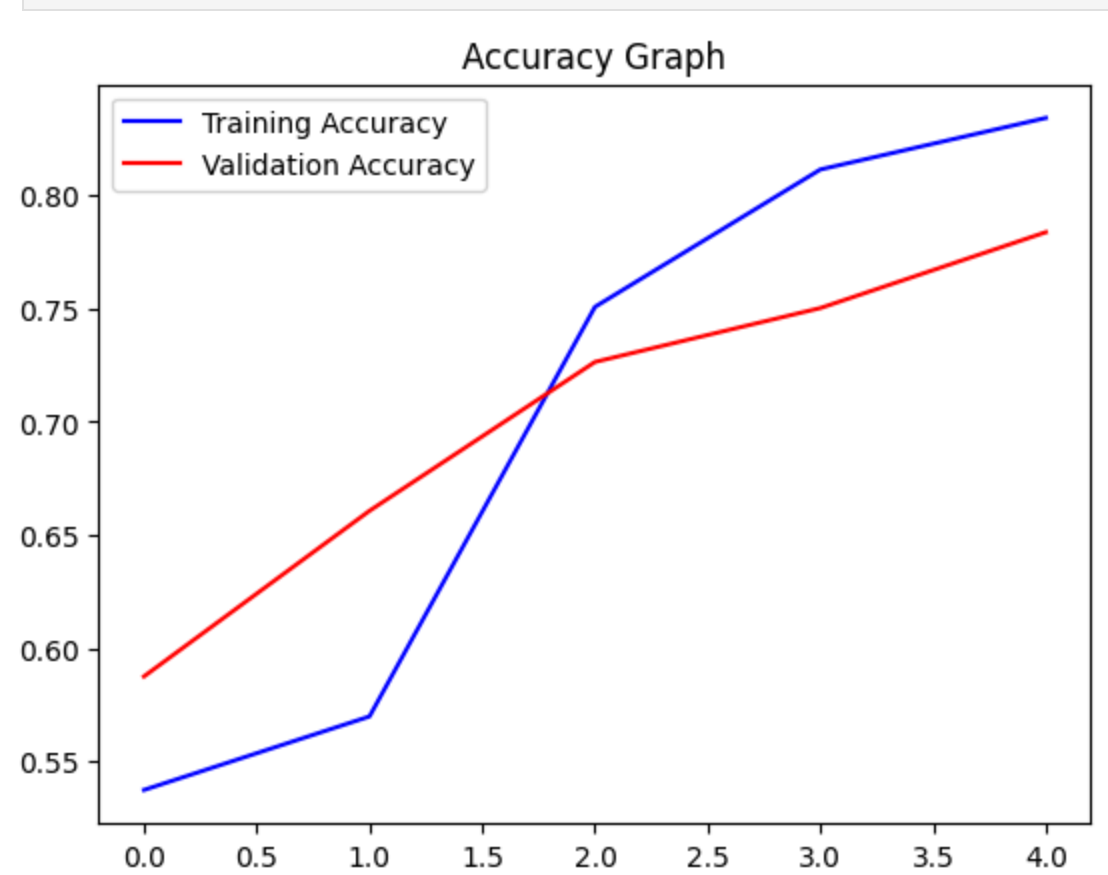
You must install pydot (`pip install pydot`) for `plot_model` to work.

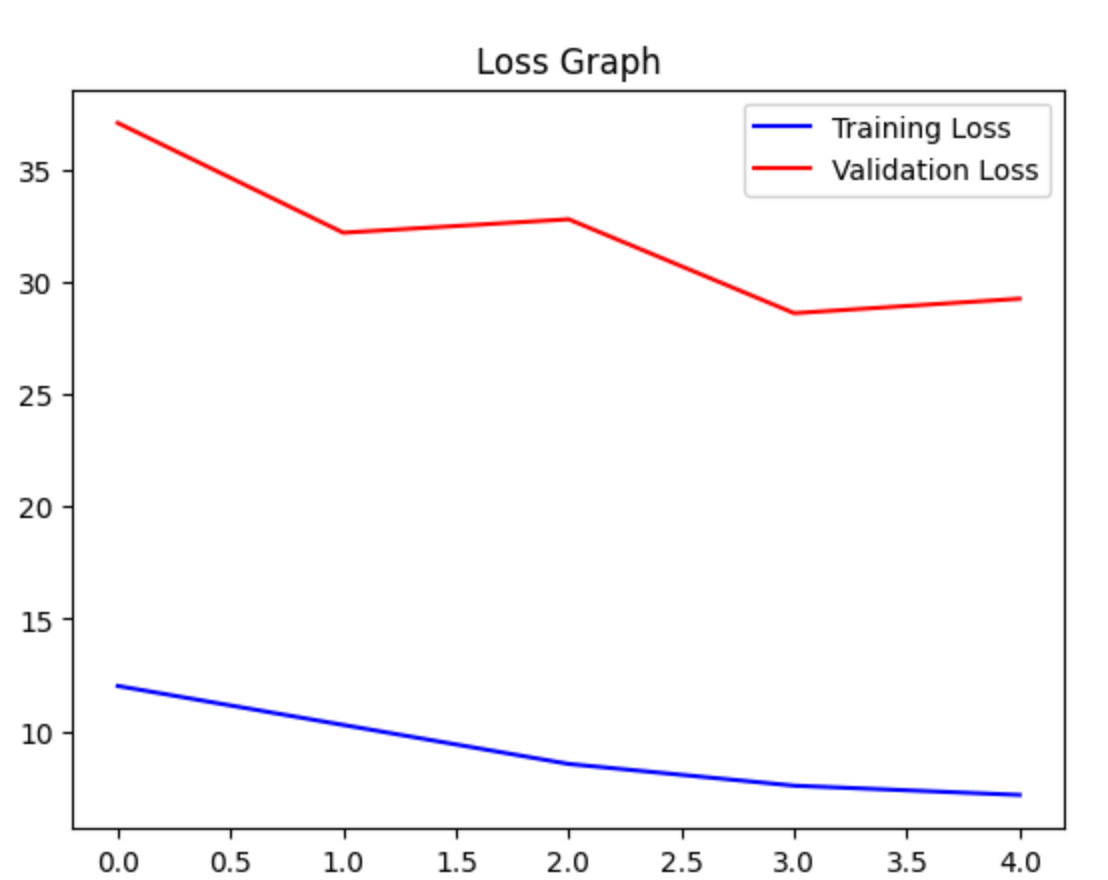
```
# train model
history = model.fit(x=X, y=[y_gender, y_age], batch_size=128, epochs=5, validation_split=0.2)
```

```
Epoch 1/5
149/149 ━━━━━━━━━━━━━━━━━━━━ 233s 1s/step - age_out_loss: 13.2983 - age_out_mae: 13.2983 - gender_out_accuracy: 0.5487 - gender_out_loss: 0.7085 - loss: 14.0068 - val_age_out_loss: 35.4122 - val_age_out_mae: 36.3634 - val_gender_out_accuracy: 0.5875 - val_gender_out_loss: 0.6893 - val_loss: 37.0522
Epoch 2/5
149/149 ━━━━━━━━━━━━━━━━━━━━ 211s 1s/step - age_out_loss: 9.6597 - age_out_mae: 9.6597 - gender_out_accuracy: 0.5329 - gender_out_loss: 0.6866 - loss: 10.3463 - val_age_out_loss: 30.9589 - val_age_out_mae: 31.5554 - val_gender_out_accuracy: 0.6607 - val_gender_out_loss: 0.6138 - val_loss: 32.1696
Epoch 3/5
149/149 ━━━━━━━━━━━━━━━━━━━━ 210s 1s/step - age_out_loss: 8.2597 - age_out_mae: 8.2599 - gender_out_accuracy: 0.7304 - gender_out_loss: 0.5475 - loss: 8.8074 - val_age_out_loss: 31.5906 - val_age_out_mae: 32.2245 - val_gender_out_accuracy: 0.7265 - val_gender_out_loss: 0.5423 - val_loss: 32.7671
Epoch 4/5
149/149 ━━━━━━━━━━━━━━━━━━━━ 215s 1s/step - age_out_loss: 7.3046 - age_out_mae: 7.3046 - gender_out_accuracy: 0.8003 - gender_out_loss: 0.4281 - loss: 7.7326 - val_age_out_loss: 27.5389 - val_age_out_mae: 28.0981 - val_gender_out_accuracy: 0.7503 - val_gender_out_loss: 0.4990 - val_loss: 28.5899
Epoch 5/5
149/149 ━━━━━━━━━━━━━━━━━━━━ 215s 1s/step - age_out_loss: 6.8312 - age_out_mae: 6.8312 - gender_out_accuracy: 0.8216 - gender_out_loss: 0.3821 - loss: 7.2133 - val_age_out_loss: 28.1614 - val_age_out_mae: 28.7494 - val_gender_out_accuracy: 0.7838 - val_gender_out_loss: 0.4796 - val_loss: 29.2331
```

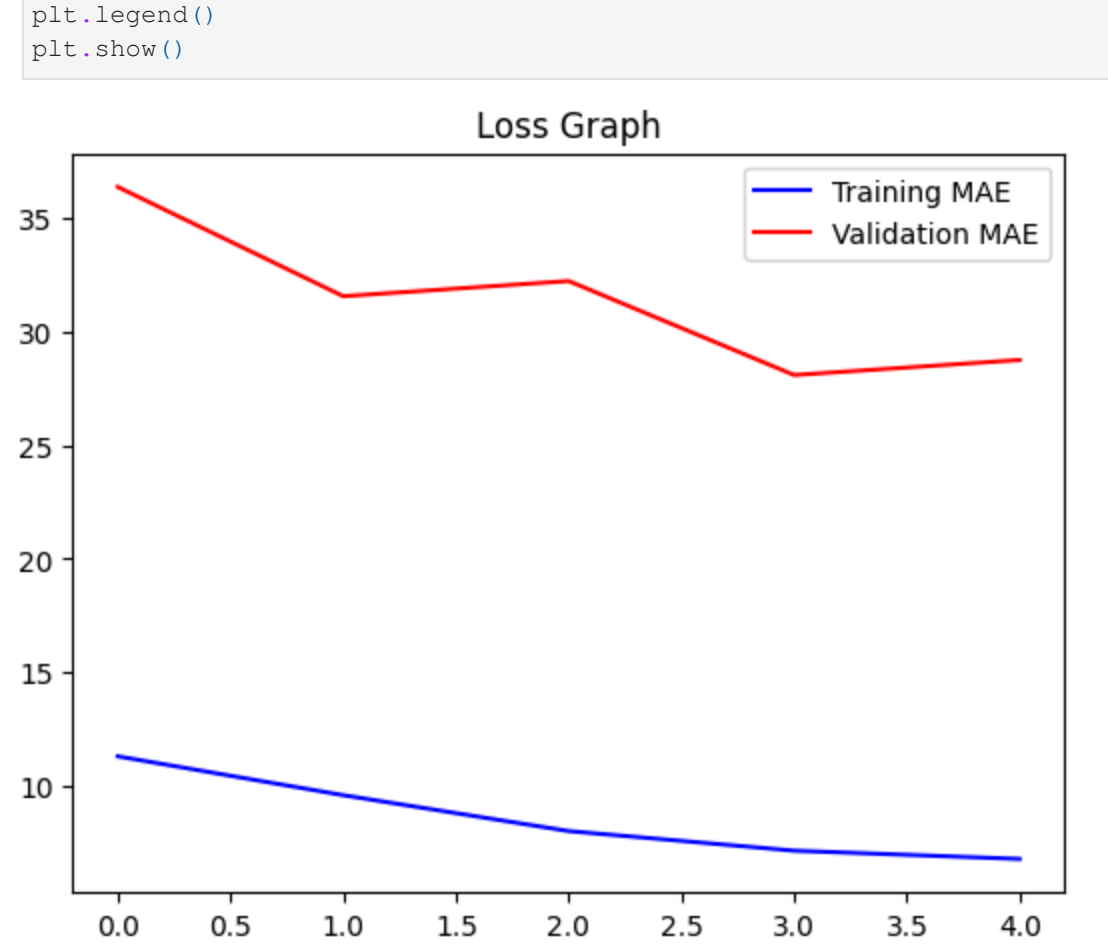
Plot the Results

```
# plot results for gender
acc = history.history['gender_out_accuracy']
val_acc = history.history['val_gender_out_accuracy']
epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training Accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation Accuracy')
plt.title('Accuracy Graph')
plt.legend()
plt.figure()

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.plot(epochs, loss, 'b', label='Training Loss')
plt.plot(epochs, val_loss, 'r', label='Validation Loss')
plt.title('Loss Graph')
plt.legend()
plt.show()
```






```
# plot results for age
loss = history.history['age_out_mae']
val_loss = history.history['val_age_out_mae']
epochs = range(len(loss))

plt.plot(epochs, loss, 'b', label='Training MAE')
plt.plot(epochs, val_loss, 'r', label='Validation MAE')
plt.title('Loss Graph')
plt.legend()
plt.show()
```



Prediction

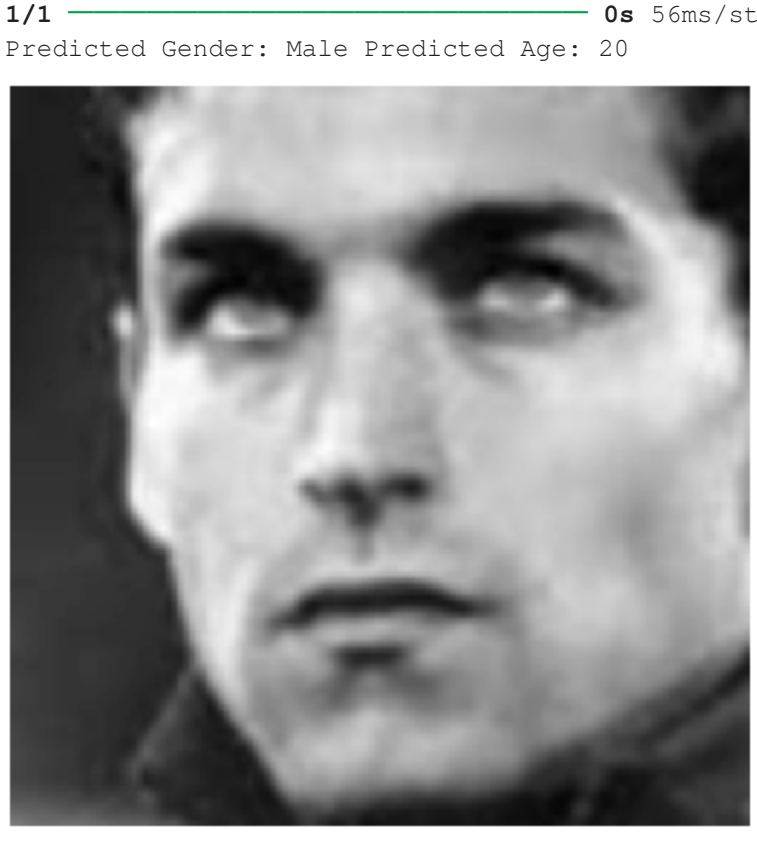
```
image_index = 100
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = round(pred[1][0][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

```
Original Gender: Male Original Age: 10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 430ms/step
Predicted Gender: Female Predicted Age: 25
```

<matplotlib.image.AxesImage at 0x162c93a97c0>

```
image_index = 3000
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = round(pred[1][0][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

```
Original Gender: Male Original Age: 21
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
Predicted Gender: Male Predicted Age: 20
```



```
image_index = 10000
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = round(pred[1][0][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray')
```

```
Original Gender: Female Original Age: 26
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
Predicted Gender: Female Predicted Age: 26
```

<matplotlib.image.AxesImage at 0x162c924db80>

Conclusion

In this project, we successfully implemented an image processing pipeline to extract features from a dataset of grayscale images. By leveraging the capabilities of the Pillow library for image manipulation and preprocessing, we established a robust framework that prepares visual data for analysis. Key Outcomes

Effective Feature Extraction: The implementation demonstrated an efficient way to resize and standardize images, ensuring consistency across the dataset.

Foundation for Further Analysis: The extracted features can serve as input for various machine learning models, opening avenues for tasks such as classification and pattern recognition.

Insights into Image Processing: This project provided valuable insights into the importance of preprocessing in computer vision applications, highlighting how quality data preparation can significantly impact model performance.

Future Work

Moving forward, this project can be expanded to include more complex models and techniques, such as convolutional neural networks (CNNs) for enhanced accuracy in image classification tasks. Additionally, incorporating data augmentation strategies could improve the model's robustness and generalization capabilities.

Overall, this work lays the groundwork for further exploration in the field of image analysis, contributing to advancements in automated visual understanding.