Importing Modules

In [39]:

```
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 [40]:

```
BASE_DIR = '../input/utkface-new/UTKFace/'
```

In [41]:

```
# 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)
```

In [42]:

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

Out[42]:

	image	age	gender
0	../input/utkface-new/UTKFace/26_0_2_2017010402...	26	0
1	../input/utkface-new/UTKFace/22_1_1_2017011223...	22	1
2	../input/utkface-new/UTKFace/21_1_3_2017010500...	21	1
3	../input/utkface-new/UTKFace/28_0_0_2017011718...	28	0
4	../input/utkface-new/UTKFace/17_1_4_2017010322...	17	1

In [43]:

```
# map labels for gender
```

```
gender_dict = {0:'Male', 1:'Female'}
```

Exploratory Data Analysis

In [44]:

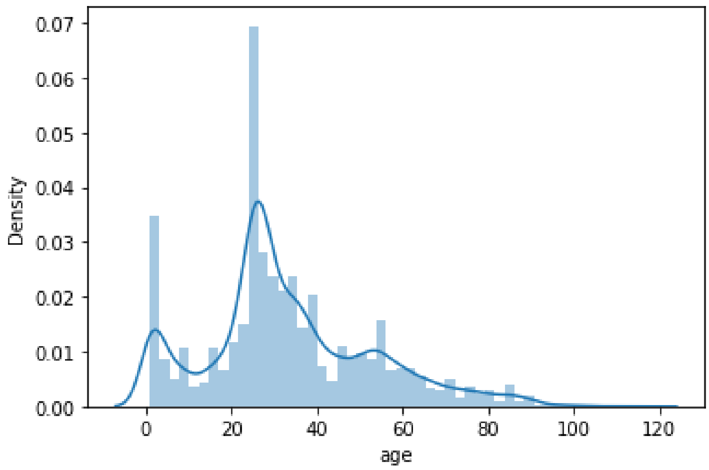
```
from PIL import Image
img = Image.open(df['image'][0])
plt.axis('off')
plt.imshow(img);
```

In [45]:

```
sns.distplot(df['age'])
```

Out[45]:

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```

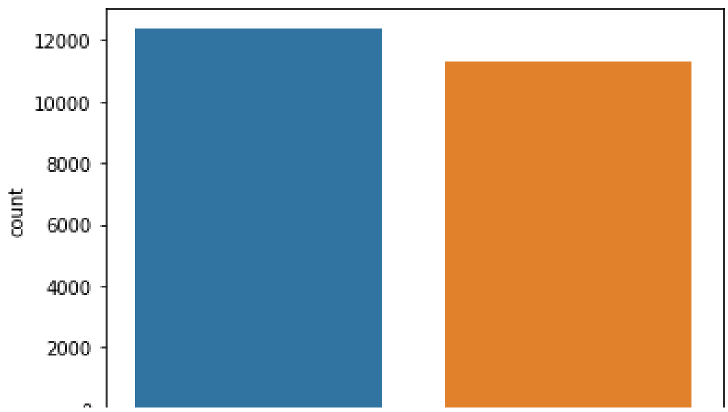


In [46]:

```
sns.countplot(df['gender'])
```

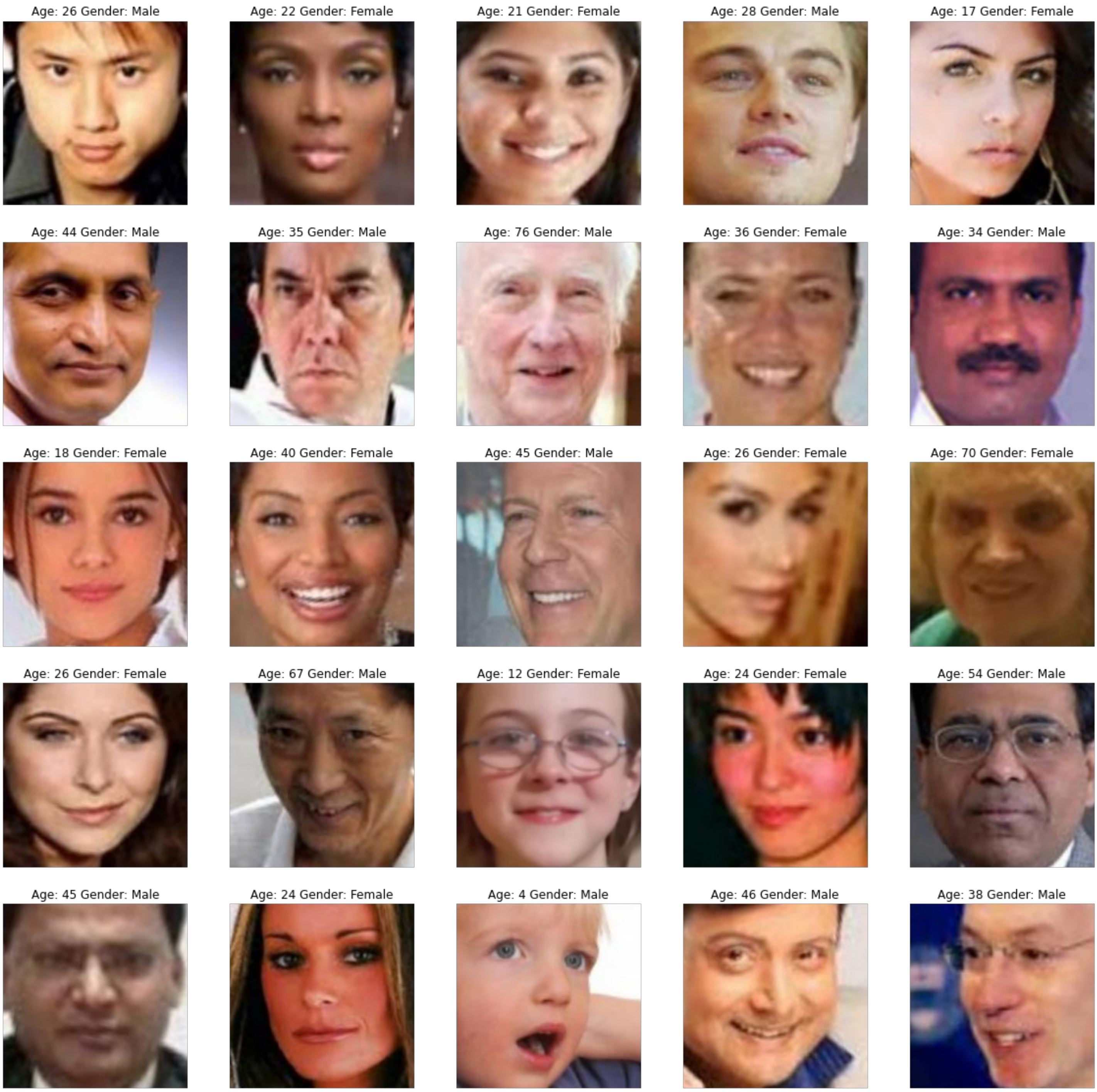
Out[46]:

```
<AxesSubplot:xlabel='gender', ylabel='count'>
```



In [47]:

```
# 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 [48]:

```
def extract_features(images):
```

```
    features = []
    for image in tqdm(images):
        img = load_img(image, grayscale=True)
        img = img.resize((128, 128), Image.ANTIALIAS)
        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 [49]:

```
X = extract_features(df['image'])
```

In [50]:

```
X.shape
```

Out[50]:

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

In [51]:

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

In [52]:

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

In [53]:

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

Model Creation

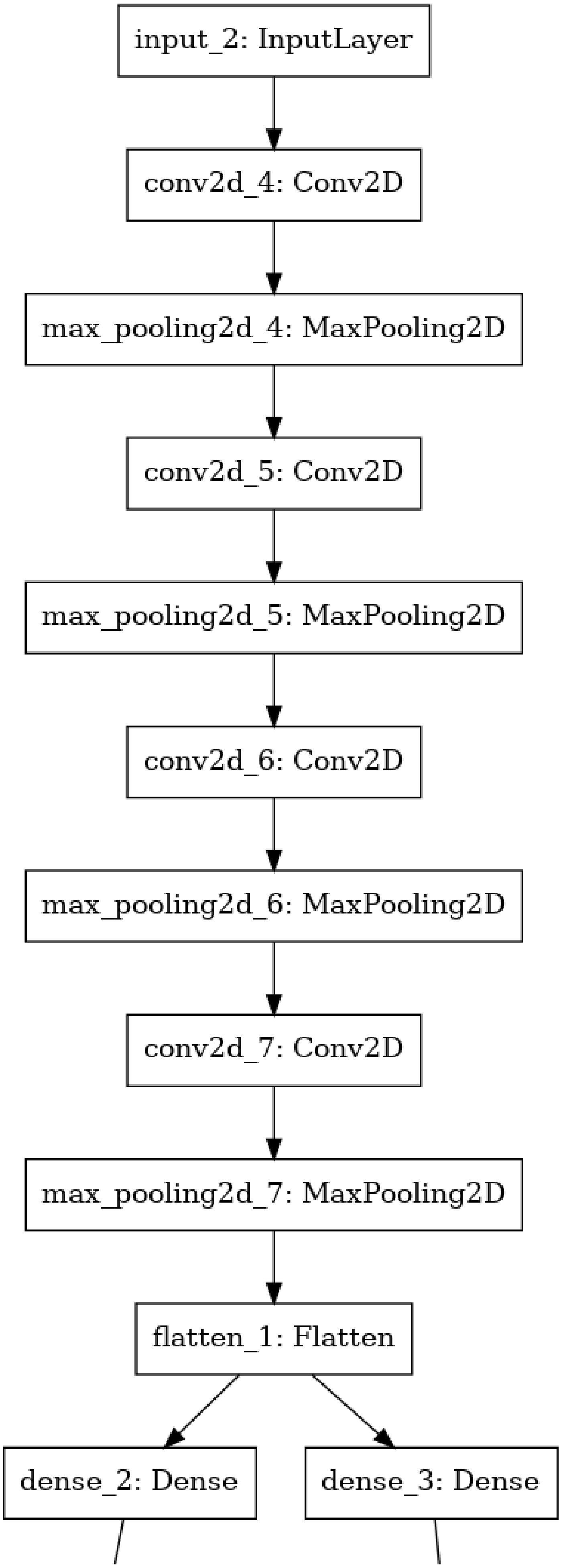
In [54]:

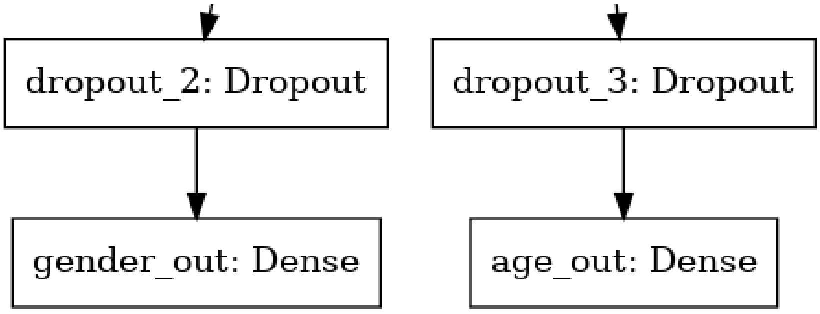
```
inputs = Input((input_shape))
# convolutional layers
conv_1 = Conv2D(32, kernel_size=(3, 3), activation='relu') (inputs)
maxp_1 = MaxPooling2D(pool_size=(2, 2)) (conv_1)
conv_2 = Conv2D(64, kernel_size=(3, 3), activation='relu') (maxp_1)
maxp_2 = MaxPooling2D(pool_size=(2, 2)) (conv_2)
conv_3 = Conv2D(128, 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(0.3) (dense_1)
dropout_2 = Dropout(0.3) (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'
])
```

In [55]:

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

Out[55]:





Training

In [56]:

```python
# train model
history = model.fit(x=X, y=[y_gender, y_age], batch_size=32, epochs=30, validation_split=0.2)
```

```
2022-11-06 13:07:57.551670: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 1242955776 exceeds 10% of free system memory.
2022-11-06 13:07:58.997097: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 1242955776 exceeds 10% of free system memory.
```

```
Epoch 1/30
593/593 [==============================] - 10s 15ms/step - loss: 15.6528 - gender_out_loss: 0.6753 - age_out_loss: 14.9775 - gender_out_accuracy: 0.5635 - age_out_accuracy: 0.0476 - val_loss: 13.1585 - val_gender_out_loss: 0.5430 - val_age_out_loss: 12.6155 - val_gender_out_accuracy: 0.7261 - val_age_out_accuracy: 0.0443
Epoch 2/30
593/593 [==============================] - 8s 14ms/step - loss: 11.5793 - gender_out_loss: 0.5037 - age_out_loss: 11.0756 - gender_out_accuracy: 0.7544 - age_out_accuracy: 0.0341 - val_loss: 10.0861 - val_gender_out_loss: 0.4533 - val_age_out_loss: 9.6328 - val_gender_out_accuracy: 0.7830 - val_age_out_accuracy: 0.0226
Epoch 3/30
593/593 [==============================] - 8s 14ms/step - loss: 9.8828 - gender_out_loss: 0.4340 - age_out_loss: 9.4488 - gender_out_accuracy: 0.7967 - age_out_accuracy: 0.0187 - val_loss: 8.9589 - val_gender_out_loss: 0.3848 - val_age_out_loss: 8.5741 - val_gender_out_accuracy: 0.8258 - val_age_out_accuracy: 0.0120
Epoch 4/30
593/593 [==============================] - 8s 13ms/step - loss: 8.6699 - gender_out_loss: 0.3753 - age_out_loss: 8.2946 - gender_out_accuracy: 0.8238 - age_out_accuracy: 0.0147 - val_loss: 8.3437 - val_gender_out_loss: 0.3498 - val_age_out_loss: 7.9939 - val_gender_out_accuracy: 0.8446 - val_age_out_accuracy: 0.0089
Epoch 5/30
593/593 [==============================] - 8s 14ms/step - loss: 7.9136 - gender_out_loss: 0.3324 - age_out_loss: 7.5812 - gender_out_accuracy: 0.8513 - age_out_accuracy: 0.0122 - val_loss: 7.5092 - val_gender_out_loss: 0.3220 - val_age_out_loss: 7.1872 - val_gender_out_accuracy: 0.8545 - val_age_out_accuracy: 0.0078
Epoch 6/30
593/593 [==============================] - 8s 14ms/step - loss: 7.4573 - gender_out_loss: 0.3079 - age_out_loss: 7.1494 - gender_out_accuracy: 0.8623 - age_out_accuracy: 0.0108 - val_loss: 7.1982 - val_gender_out_loss: 0.2940 - val_age_out_loss: 6.9042 - val_gender_out_accuracy: 0.8707 - val_age_out_accuracy: 0.0059
Epoch 7/30
593/593 [==============================] - 8s 14ms/step - loss: 7.1406 - gender_out_loss: 0.2883 - age_out_loss: 6.8523 - gender_out_accuracy: 0.8735 - age_out_accuracy: 0.0104 - val_loss: 7.0195 - val_gender_out_loss: 0.2805 - val_age_out_loss: 6.7390 - val_gender_out_accuracy: 0.8760 - val_age_out_accuracy: 0.0055
Epoch 8/30
593/593 [==============================] - 8s 13ms/step - loss: 6.9357 - gender_out_loss: 0.2766 - age_out_loss: 6.6591 - gender_out_accuracy: 0.8795 - age_out_accuracy: 0.0109 - val_loss: 7.5386 - val_gender_out_loss: 0.2722 - val_age_out_loss: 7.2664 - val_gender_out_accuracy: 0.8754 - val_age_out_accuracy: 0.0076
Epoch 9/30
593/593 [==============================] - 8s 14ms/step - loss: 6.6552 - gender_out_loss: 0.2603 - age_out_loss: 6.3950 - gender_out_accuracy: 0.8896 - age_out_accuracy: 0.0103 - val_loss: 7.0470 - val_gender_out_loss: 0.2819 - val_age_out_loss: 6.7650 - val_gender_out_accuracy: 0.8747 - val_age_out_accuracy: 0.0061
Epoch 10/30
593/593 [==============================] - 8s 14ms/step - loss: 6.3647 - gender_out_loss:
```

```
593/593 [==============================] - 8s 14ms/step - loss: 6.3647 - gender_out_loss:
0.2502 - age_out_loss: 6.1145 - gender_out_accuracy: 0.8906 - age_out_accuracy: 0.0091 -
val_loss: 6.9099 - val_gender_out_loss: 0.2694 - val_age_out_loss: 6.6405 - val_gender_ou
t_accuracy: 0.8859 - val_age_out_accuracy: 0.0057
Epoch 11/30
593/593 [==============================] - 8s 13ms/step - loss: 6.1276 - gender_out_loss:
0.2387 - age_out_loss: 5.8889 - gender_out_accuracy: 0.8989 - age_out_accuracy: 0.0088 -
val_loss: 7.1338 - val_gender_out_loss: 0.2665 - val_age_out_loss: 6.8672 - val_gender_ou
t_accuracy: 0.8758 - val_age_out_accuracy: 0.0042
Epoch 12/30
593/593 [==============================] - 8s 14ms/step - loss: 5.9246 - gender_out_loss:
0.2305 - age_out_loss: 5.6941 - gender_out_accuracy: 0.9023 - age_out_accuracy: 0.0099 -
val_loss: 6.8932 - val_gender_out_loss: 0.2580 - val_age_out_loss: 6.6351 - val_gender_ou
t_accuracy: 0.8889 - val_age_out_accuracy: 0.0042
Epoch 13/30
593/593 [==============================] - 8s 13ms/step - loss: 5.6747 - gender_out_loss:
0.2163 - age_out_loss: 5.4584 - gender_out_accuracy: 0.9078 - age_out_accuracy: 0.0096 -
val_loss: 6.9624 - val_gender_out_loss: 0.2962 - val_age_out_loss: 6.6662 - val_gender_ou
t_accuracy: 0.8777 - val_age_out_accuracy: 0.0046
Epoch 14/30
593/593 [==============================] - 8s 14ms/step - loss: 5.5883 - gender_out_loss:
0.2106 - age_out_loss: 5.3778 - gender_out_accuracy: 0.9093 - age_out_accuracy: 0.0095 -
val_loss: 7.0194 - val_gender_out_loss: 0.2613 - val_age_out_loss: 6.7581 - val_gender_ou
t_accuracy: 0.8874 - val_age_out_accuracy: 0.0046
Epoch 15/30
593/593 [==============================] - 8s 13ms/step - loss: 5.3596 - gender_out_loss:
0.2028 - age_out_loss: 5.1568 - gender_out_accuracy: 0.9137 - age_out_accuracy: 0.0097 -
val_loss: 7.1085 - val_gender_out_loss: 0.2752 - val_age_out_loss: 6.8333 - val_gender_ou
t_accuracy: 0.8893 - val_age_out_accuracy: 0.0051
Epoch 16/30
593/593 [==============================] - 8s 13ms/step - loss: 5.1682 - gender_out_loss:
0.1934 - age_out_loss: 4.9748 - gender_out_accuracy: 0.9183 - age_out_accuracy: 0.0089 -
val_loss: 6.8220 - val_gender_out_loss: 0.2744 - val_age_out_loss: 6.5475 - val_gender_ou
t_accuracy: 0.8855 - val_age_out_accuracy: 0.0067
Epoch 17/30
593/593 [==============================] - 8s 13ms/step - loss: 5.0024 - gender_out_loss:
0.1875 - age_out_loss: 4.8149 - gender_out_accuracy: 0.9216 - age_out_accuracy: 0.0098 -
val_loss: 7.0305 - val_gender_out_loss: 0.2734 - val_age_out_loss: 6.7571 - val_gender_ou
t_accuracy: 0.8878 - val_age_out_accuracy: 0.0076
Epoch 18/30
593/593 [==============================] - 8s 14ms/step - loss: 4.8957 - gender_out_loss:
0.1799 - age_out_loss: 4.7158 - gender_out_accuracy: 0.9254 - age_out_accuracy: 0.0111 -
val_loss: 6.7488 - val_gender_out_loss: 0.2810 - val_age_out_loss: 6.4679 - val_gender_ou
t_accuracy: 0.8865 - val_age_out_accuracy: 0.0097
Epoch 19/30
593/593 [==============================] - 8s 13ms/step - loss: 4.7517 - gender_out_loss:
0.1723 - age_out_loss: 4.5795 - gender_out_accuracy: 0.9271 - age_out_accuracy: 0.0180 -
val_loss: 6.8582 - val_gender_out_loss: 0.2935 - val_age_out_loss: 6.5647 - val_gender_ou
t_accuracy: 0.8897 - val_age_out_accuracy: 0.0164
Epoch 20/30
593/593 [==============================] - 8s 13ms/step - loss: 4.6152 - gender_out_loss:
0.1637 - age_out_loss: 4.4515 - gender_out_accuracy: 0.9321 - age_out_accuracy: 0.0256 -
val_loss: 6.7061 - val_gender_out_loss: 0.2848 - val_age_out_loss: 6.4213 - val_gender_ou
t_accuracy: 0.8842 - val_age_out_accuracy: 0.0346
Epoch 21/30
593/593 [==============================] - 8s 13ms/step - loss: 4.4771 - gender_out_loss:
0.1565 - age_out_loss: 4.3206 - gender_out_accuracy: 0.9317 - age_out_accuracy: 0.0345 -
val_loss: 6.7980 - val_gender_out_loss: 0.3328 - val_age_out_loss: 6.4652 - val_gender_ou
t_accuracy: 0.8827 - val_age_out_accuracy: 0.0352
Epoch 22/30
593/593 [==============================] - 8s 14ms/step - loss: 4.3761 - gender_out_loss:
0.1499 - age_out_loss: 4.2262 - gender_out_accuracy: 0.9369 - age_out_accuracy: 0.0370 -
val_loss: 7.0094 - val_gender_out_loss: 0.3167 - val_age_out_loss: 6.6927 - val_gender_ou
t_accuracy: 0.8861 - val_age_out_accuracy: 0.0407
Epoch 23/30
593/593 [==============================] - 8s 13ms/step - loss: 4.3791 - gender_out_loss:
0.1447 - age_out_loss: 4.2344 - gender_out_accuracy: 0.9405 - age_out_accuracy: 0.0382 -
val_loss: 6.9121 - val_gender_out_loss: 0.3121 - val_age_out_loss: 6.6000 - val_gender_ou
t_accuracy: 0.8794 - val_age_out_accuracy: 0.0354
Epoch 24/30
593/593 [==============================] - 8s 13ms/step - loss: 4.2377 - gender_out_loss:
0.1440 - age_out_loss: 4.0937 - gender_out_accuracy: 0.9392 - age_out_accuracy: 0.0404 -
val_loss: 6.7598 - val_gender_out_loss: 0.3431 - val_age_out_loss: 6.4167 - val_gender_ou
```

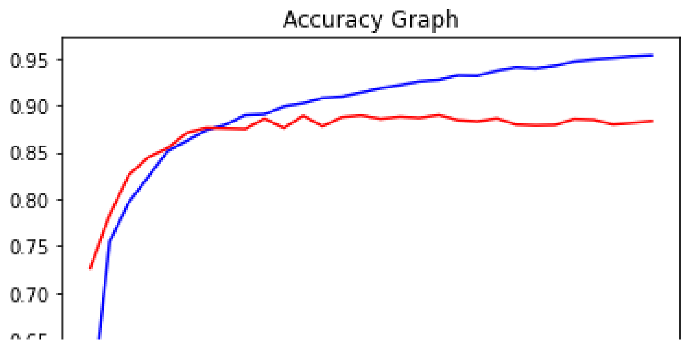
```
val_loss: 6.7598 - val_gender_out_loss: 0.3431 - val_age_out_loss: 6.4167 - val_gender_ou
t_accuracy: 0.8785 - val_age_out_accuracy: 0.0390
Epoch 25/30
593/593 [==============================] - 8s 13ms/step - loss: 4.0874 - gender_out_loss:
0.1338 - age_out_loss: 3.9535 - gender_out_accuracy: 0.9421 - age_out_accuracy: 0.0425 -
val_loss: 6.9762 - val_gender_out_loss: 0.3603 - val_age_out_loss: 6.6159 - val_gender_ou
t_accuracy: 0.8790 - val_age_out_accuracy: 0.0441
Epoch 26/30
593/593 [==============================] - 9s 15ms/step - loss: 4.0404 - gender_out_loss:
0.1259 - age_out_loss: 3.9145 - gender_out_accuracy: 0.9468 - age_out_accuracy: 0.0424 -
val_loss: 6.7752 - val_gender_out_loss: 0.3399 - val_age_out_loss: 6.4353 - val_gender_ou
t_accuracy: 0.8855 - val_age_out_accuracy: 0.0437
Epoch 27/30
593/593 [==============================] - 8s 13ms/step - loss: 4.0058 - gender_out_loss:
0.1188 - age_out_loss: 3.8870 - gender_out_accuracy: 0.9490 - age_out_accuracy: 0.0429 -
val_loss: 6.8348 - val_gender_out_loss: 0.3789 - val_age_out_loss: 6.4560 - val_gender_ou
t_accuracy: 0.8846 - val_age_out_accuracy: 0.0426
Epoch 28/30
593/593 [==============================] - 8s 13ms/step - loss: 3.9255 - gender_out_loss:
0.1187 - age_out_loss: 3.8068 - gender_out_accuracy: 0.9505 - age_out_accuracy: 0.0437 -
val_loss: 6.8525 - val_gender_out_loss: 0.3635 - val_age_out_loss: 6.4890 - val_gender_ou
t_accuracy: 0.8796 - val_age_out_accuracy: 0.0422
Epoch 29/30
593/593 [==============================] - 8s 13ms/step - loss: 3.8837 - gender_out_loss:
0.1111 - age_out_loss: 3.7726 - gender_out_accuracy: 0.9522 - age_out_accuracy: 0.0441 -
val_loss: 6.9580 - val_gender_out_loss: 0.4186 - val_age_out_loss: 6.5394 - val_gender_ou
t_accuracy: 0.8813 - val_age_out_accuracy: 0.0434
Epoch 30/30
593/593 [==============================] - 8s 14ms/step - loss: 3.8749 - gender_out_loss:
0.1100 - age_out_loss: 3.7648 - gender_out_accuracy: 0.9533 - age_out_accuracy: 0.0437 -
val_loss: 6.9042 - val_gender_out_loss: 0.3818 - val_age_out_loss: 6.5225 - val_gender_ou
t_accuracy: 0.8832 - val_age_out_accuracy: 0.0437
```

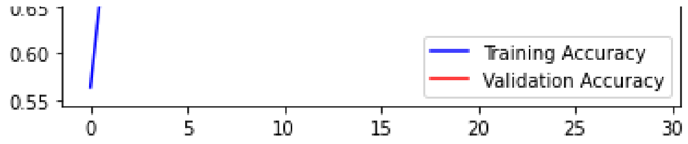
Plot The Results

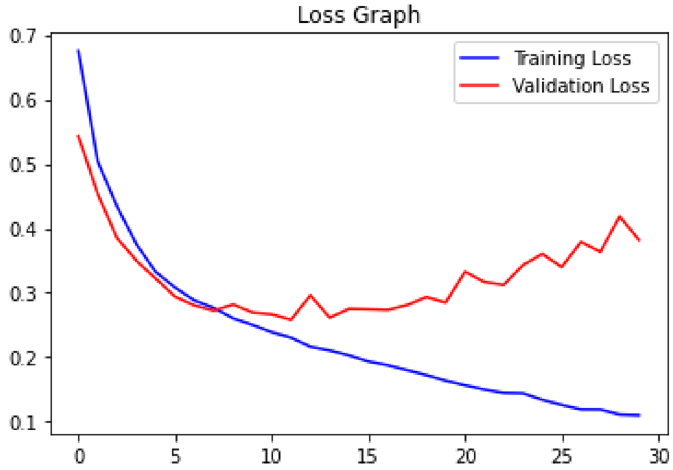
Results for Gender

In [57]:

```python
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['gender_out_loss']
val_loss = history.history['val_gender_out_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()
```



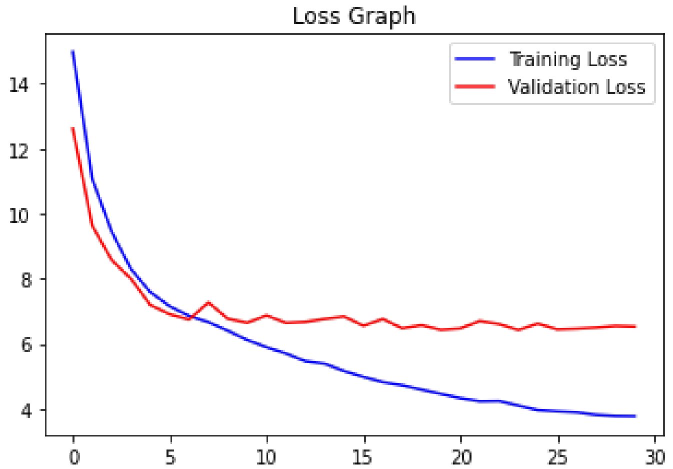




Results for Age

In [58]:

```
loss = history.history['age_out_loss']
val_loss = history.history['val_age_out_loss']
epochs = range(len(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()
```



Final Results

In [59]:

```
image_index = 100
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[imag
e_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: 3
```

Predicted Gender: Female Predicted Age: 2

In [60]:

```
image_index = 3000
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[imag
e_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: 28
Predicted Gender: Male Predicted Age: 22

In [61]:

```
image_index = 10000
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[imag
e_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: 42
Predicted Gender: Male Predicted Age: 42

In [62]:

```
image_index = 4663
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[imag
e_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: 27
Predicted Gender: Male Predicted Age: 24
```

In [63]:

```
image_index = 12345
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[imag
e_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: 26
Predicted Gender: Male Predicted Age: 29
```

In [64]:

```
image_index = 23707
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:", y_age[imag
e_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: 66
Predicted Gender: Male Predicted Age: 66
```