**Multi-Output Convolutional Neural Network for Age,**

**Gender, and Race Prediction Using the UTKFace Dataset**

**and Real-Time Webcam Inference**

**BACKGROUND AND CONTEXT**

This project demonstrates the practical application of machine learning techniques on a real-world dataset, in fulfillment of the course requirements for MIT 266: Special Topic re: Machine Learning on a Real-World Dataset. The work involves training a multi-output convolutional neural network (CNN) capable of predicting three different attributes—age, gender, and race—from a single facial image. The project utilizes the UTKFace dataset, which contains thousands of facial images labeled with age, gender, and race metadata. These images serve as training data for a deep learning model designed to learn shared representations and simultaneously output multiple predictions from a single input image.

In addition to model training and evaluation, the project also includes an implementation of real-time prediction using a webcam feed. This integrates the trained model with OpenCV, allowing for immediate visual feedback and demonstration of the model’s performance in a live setting. The overall goal is to highlight the end-to-end process of applying machine learning to a real-world dataset—from data preprocessing and model design, to training, evaluation, and real-time deployment. This approach reinforces the core objectives of MIT 266, which emphasize hands-on experience and the development of working ML applications using real data.

**DATASET DESCRIPTION:**

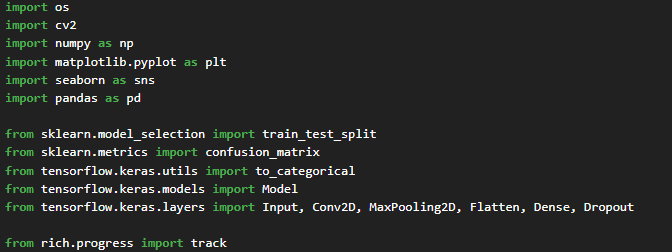
The UTKFace dataset is a large-scale facial image dataset that contains over 23,708 labeled images of human faces with a wide diversity in age (0 to 116 years old), gender, and ethnicity. Each image is cropped and aligned to focus on the face and is stored in JPG format with a resolution of 200x200 pixels. The filenames of the images encode the age, gender, and race labels in the format: [age]\_[gender]\_[race]\_[date&time].jpg,  
making it easy to parse labels programmatically. This dataset is widely used for research in tasks such as age estimation, gender classification, ethnicity recognition, and multi-task learning in computer vision. It is particularly valuable for training deep learning models due to its real-world variability in lighting, facial expressions, and occlusions.

**Source:** <https://www.kaggle.com/datasets/jangedoo/utkface-new>

**FOUR MAIN FEATURES OF THE UTKFACE DATASET:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Description** | **Type** | **Example Values** |
| **Face** | Cropped and aligned facial image of a person | Image (JPG) | 200×200 RGB face photo |
| **Age** | Estimated age of the person in the image | Numerical (int) | 0, 5, 23, 45, 60, 87, 116 |
| **Gender** | Binary label representing the gender of the person | Categorical (int) | 0 = Male, 1 = Female |
| **Race** | Ethnicity of the person in the image | Categorical (int) | 0 = White, 1 = Black, 2 = Asian, 3 = Indian, 4 = Other |

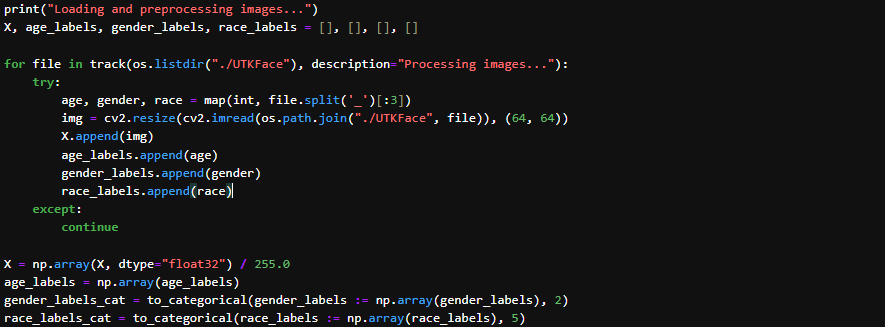
**CODE SNIPPET #1: CODE FOR IMPORTING REQUIRED LIBRARIES**



**Function:**

This snippet imports all necessary Python libraries for the real-time CNN-based age, gender, and race prediction project. OpenCV handles image processing and webcam interaction. NumPy and Pandas manage data structures and numerical operations. Matplotlib and Seaborn are used for visualization. Scikit-learn offers utility functions for data preprocessing and evaluation. Keras (via TensorFlow) defines and trains the multi-output CNN model. The rich library provides elegant CLI progress tracking.

**CODE SNIPPET #2: CODE FOR LOADING AND PREPROCESSING THE UTKFACE DATASET**

****

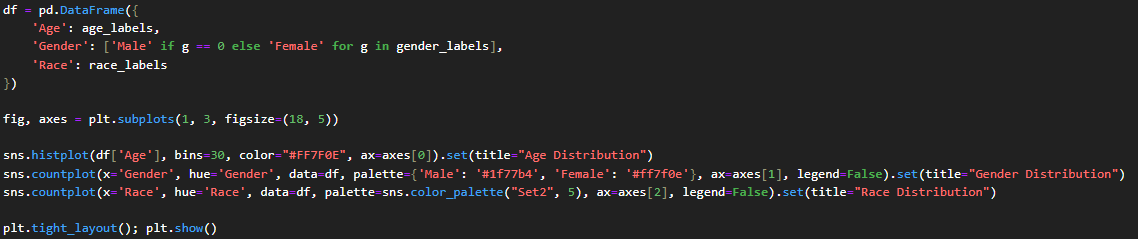
**Function:**

This code loads the UTKFace dataset and preprocesses each image by resizing to 64x64 and normalizing pixel values. The image filenames encode age, gender, and race information, which are parsed and stored in separate arrays. Gender and race labels are one-hot encoded to suit classification objectives in neural networks. This ensures the data is in a format suitable for training a multi-output CNN.

**Dataset Description:**

The UTKFace dataset is a large-scale face dataset with a wide age range (0 to 116 years). Each image filename is formatted as [age]\_[gender]\_[race]\_[date&time].jpg, allowing label extraction. It contains 23,708 labeled faceimages of diverse ethnicities and genders, captured under uncontrolled conditions. This makes it suitable for building robust models for real-world age, gender, and race estimation.

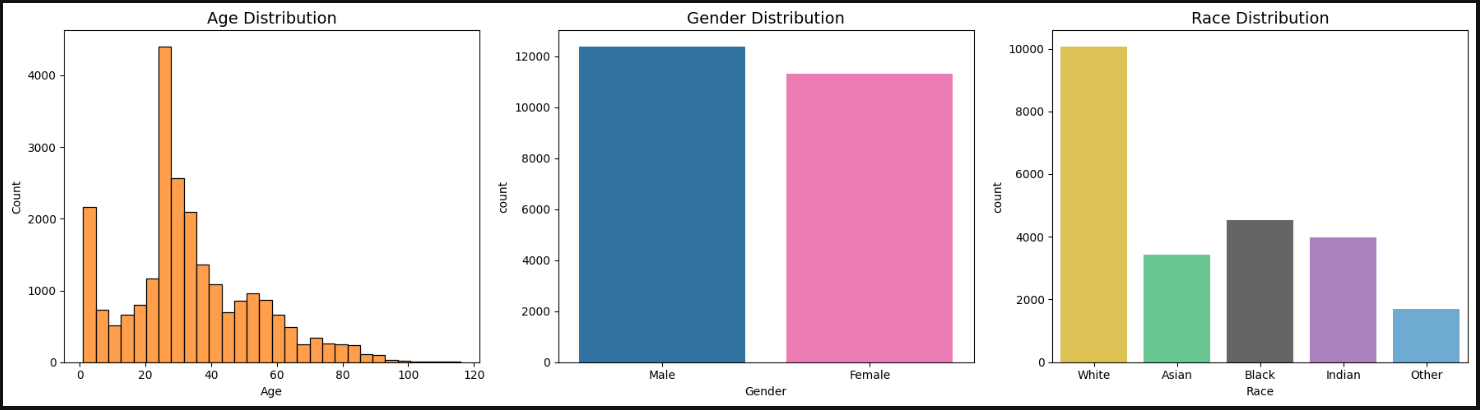
## CODE SNIPPET #3: CODE FOR VISUALIZING THE DATASET



### Function:

This code creates three visualizations using Seaborn: a histogram for age distribution and count plots for gender and race. It assumes that the DataFrame df has already been defined in a previous code cell. The custom color palettes improve clarity and visual appeal. These charts are useful for evaluating dataset balance and identifying demographic biases that may impact model performance.

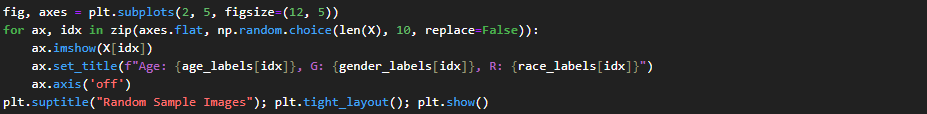
## Figure #1:



### Description:

The plots show the distribution of age, gender, and race labels. The age distribution histogram indicates more samples in younger age groups, while the gender and race count plots reveal potential imbalances. Gender data might be fairly balanced, but race classes are skewed, with the "White" class having the most samples. These insights inform model performance and fairness.

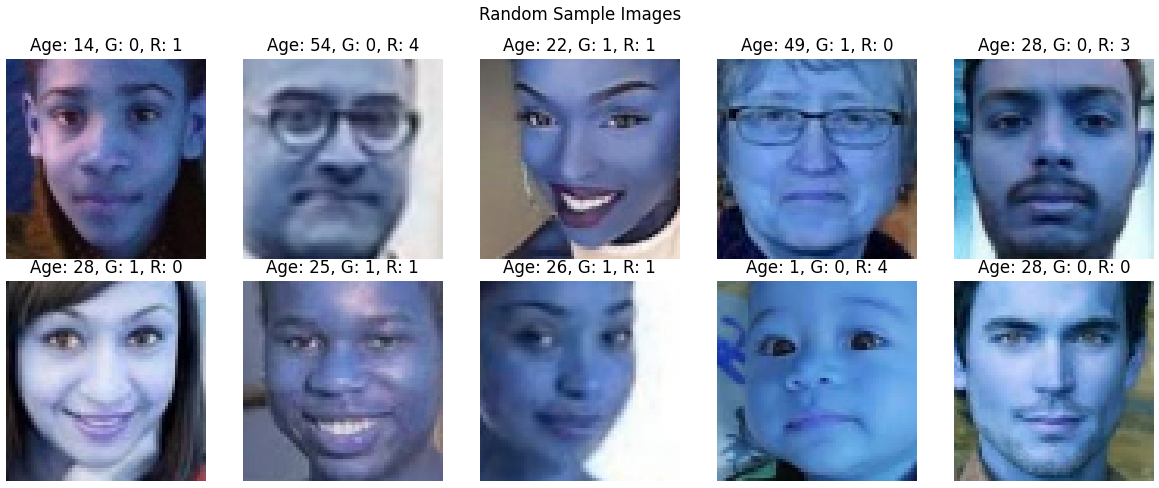
## CODE SNIPPET #4: DISPLAY SAMPLE IMAGES



### Function:

This code randomly selects and displays 10 images from the dataset to visually verify correct loading and labeling. Each subplot shows the image and its corresponding Age, Gender, and Race label. This step is helpful in validating dataset integrity before training.

## Figure #1:



### Description:

This code randomly selects and displays 10 images from the dataset to visually verify correct loading and labeling. Each subplot shows the image and its corresponding Age, Gender, and Race label. This step is helpful in validating dataset integrity before training.

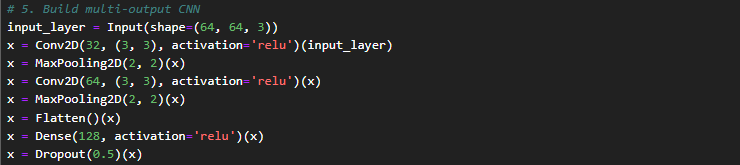
**CODE SNIPPET #5: CODE FOR TRAIN-TEST SPLIT**



**Function:**

This code divides the dataset into training and testing sets with an 80-20 split. The training set is used to fit the model, while the testing set evaluates generalization performance. The random\_state=42 parameter ensures reproducibility, generating the same split on every run. Each set contains image features and three target labels.

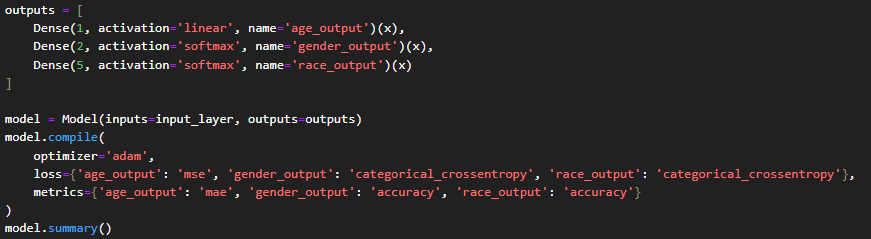
**CODE SNIPPET #6: CODE FOR BUILDING THE CNN ARCHITECTURE**

****

**Function:**

This defines the core feature extractor of the CNN using the Keras Functional API. It consists of two convolutional layers followed by max-pooling to reduce spatial dimensions. The output is flattened and passed through a dense layer with dropout to mitigate overfitting. This forms the shared base for all three prediction heads (age, gender, and race).

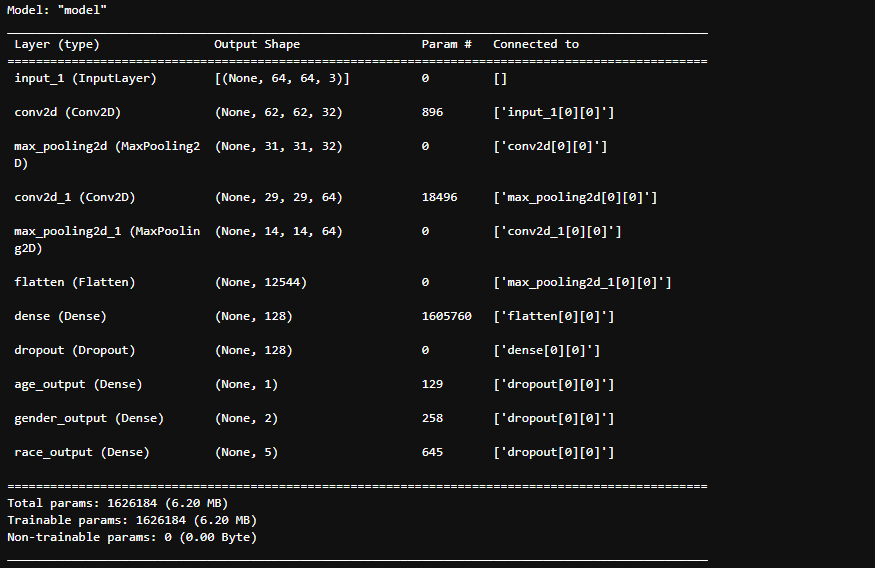
**CODE SNIPPET #7: CODE FOR ADDING OUTPUT LAYERS AND COMPILING THE MODEL**

****

**Function:**

The code defines three parallel output branches from the shared CNN base: age prediction (regression), gender classification (2-class softmax), and race classification (5-class softmax). The model is compiled with multiple loss functions and metrics tailored to each task. adam optimizer is used for training efficiency. This design supports multi-task learning, improving overall generalization.

Figure:



**Description:**

The output shows that the model is a **multi-output Convolutional Neural Network (CNN)** designed to predict **age, gender, and race** from facial images.

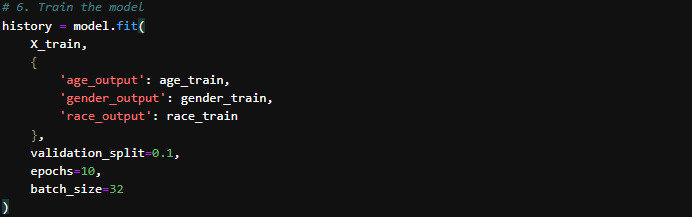
The model takes an input image of shape **(64, 64, 3)**, which corresponds to a 64×64 RGB image.

The architecture is structured as follows:

* An InputLayer receives the input image.
* A Conv2D layer with 32 filters extracts low-level features, followed by a MaxPooling2D layer to reduce the spatial dimensions.
* A second Conv2D layer with 64 filters deepens feature extraction, again followed by max pooling.
* A Flatten layer converts the feature maps into a one-dimensional vector.
* A Dense (fully connected) layer with 128 neurons processes the flattened features.
* A Dropout layer helps reduce overfitting.
* The network branches into three outputs:
  + age\_output is a dense layer with 1 unit, likely performing **regression** to estimate age.
  + gender\_output is a dense layer with 2 units for **binary classification** (e.g., male/female).
  + race\_output is a dense layer with 5 units for **multi-class classification** of race categories.

The model has a total of **1,626,184 parameters**, all of which are trainable. This means the model is fully optimized during training to learn facial features that can predict age, gender, and race simultaneously.

**CODE SNIPPET #8: CODE FOR TRAINING THE MODEL**



### Function:

This code trains the model for 10 epochs with a batch size of 32. It uses 10% of the training data as validation to monitor model performance during training. Losses and metrics for all three outputs (age, gender, race) are logged per epoch.

### Figure:

### 

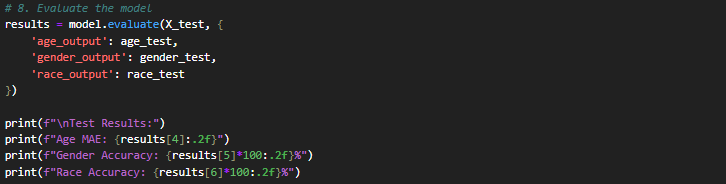
### Description:

The training logs show progressive learning over epochs, with age\_output\_mae decreasing and gender\_output\_accuracy and race\_output\_accuracy improving. Validation metrics help detect overfitting.

The method returns a list of loss values and evaluation metrics in the same order as defined during model.compile():

1. Total loss (combined for all outputs),
2. Age loss (Mean Squared Error),
3. Gender loss (Categorical Crossentropy),
4. Race loss (Categorical Crossentropy),
5. Age MAE (Mean Absolute Error),
6. Gender accuracy, rint(f"Race Accuracy: {results[6]\*100:.2f}%")

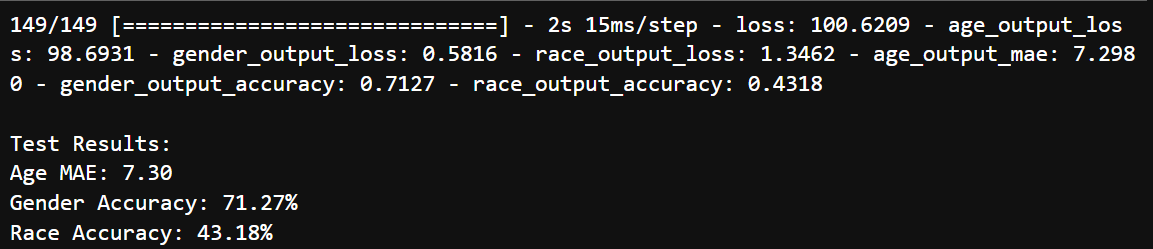
**CODE SNIPPET #9: CODE FOR EVALUATING THE MODEL**



**Function:**

This evaluates the trained model on the test dataset and prints out the loss and metrics for all three outputs. It gives a quantitative assessment of the model’s final performance.

**Figure:**



**Description:**

The evaluation output offers a comprehensive summary of the model’s performance on the test dataset. The 149/149 denotes the total number of evaluation batches processed. The overall loss value (100.6209) represents the combined loss across all three prediction tasks—age, gender, and race.

Breaking down the individual losses:

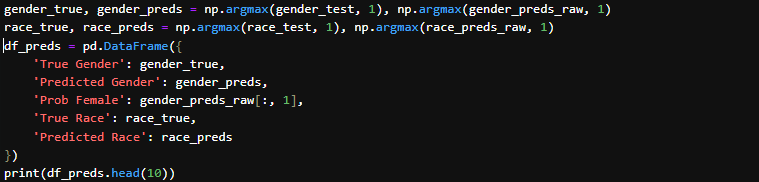
* age\_output\_loss (98.6931) accounts for the error in age estimation,
* gender\_output\_loss (0.5816) reflects the classification loss for gender, and
* race\_output\_loss (1.3462) pertains to race classification error.

The mean absolute error (MAE) for age prediction, reported as age\_output\_mae (7.2980), indicates that the predicted ages deviate by approximately 7.3 years from the actual values on average.

Classification performance is measured by:

* gender\_output\_accuracy (0.7127) — about 71.3% accuracy in predicting gender,
* race\_output\_accuracy (0.4318) — roughly 43.2% accuracy in predicting race.

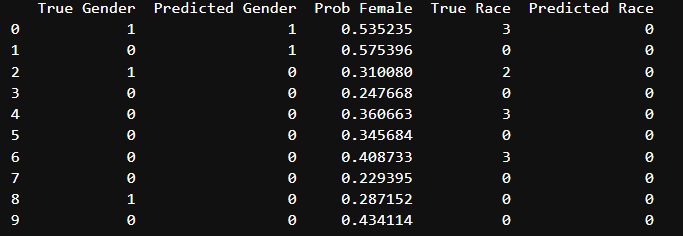
**CODE SNIPPET #10: CONVERT PROBABILITIES TO CLASS PREDICTIONS**

****

**Function:**

This code converts model output probabilities into predicted class labels for gender and race, extracts the probability of being female, and creates a DataFrame comparing true vs. predicted values for analysis, then prints the first 10 rows.

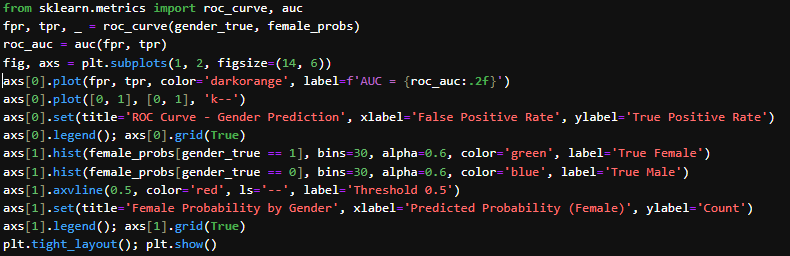
**Figure:**



**Description:**

The output shows a comparison between the true and predicted values for gender and race for the first 10 samples, including the model's estimated probability of each individual being female (Prob Female); for example, in row 0, the true gender is female (1), the model correctly predicted female (1), with a probability of 53.52%, but in row 2, the model incorrectly predicted male (0) despite the true gender being female (1), showing a lower confidence of 31.01%—indicating both correct and incorrect predictions with varying confidence levels.

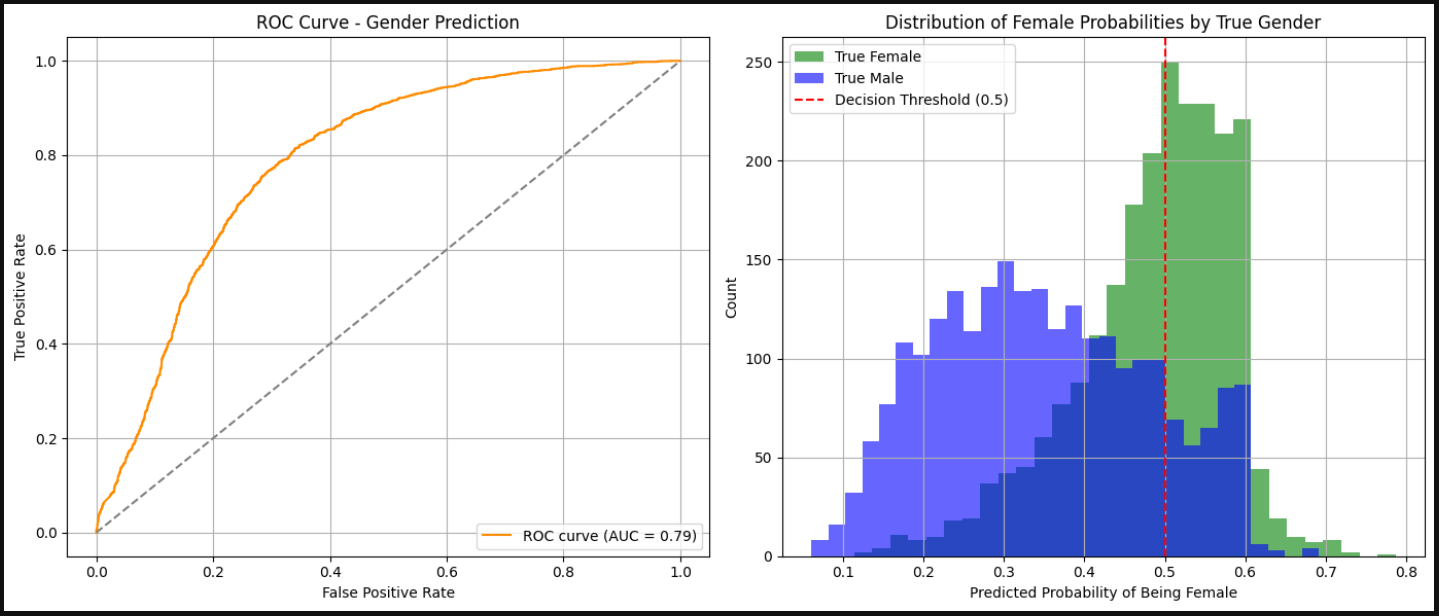
**Code Snippet #11: Code Snippet for Evaluating Gender Classification Performance: ROC Curve and Probability Distribution Analysis**

****

**Function:**

This code evaluates a binary gender classifier by plotting the ROC curve with AUC to measure performance and displaying histograms of predicted female probabilities to visualize class separation and the decision threshold.

**Figure:**

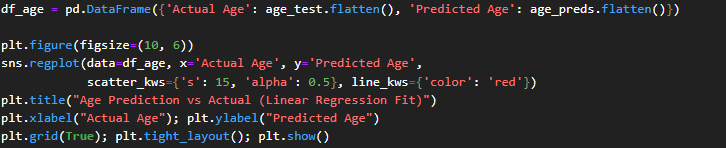
****

**Descriptions:**

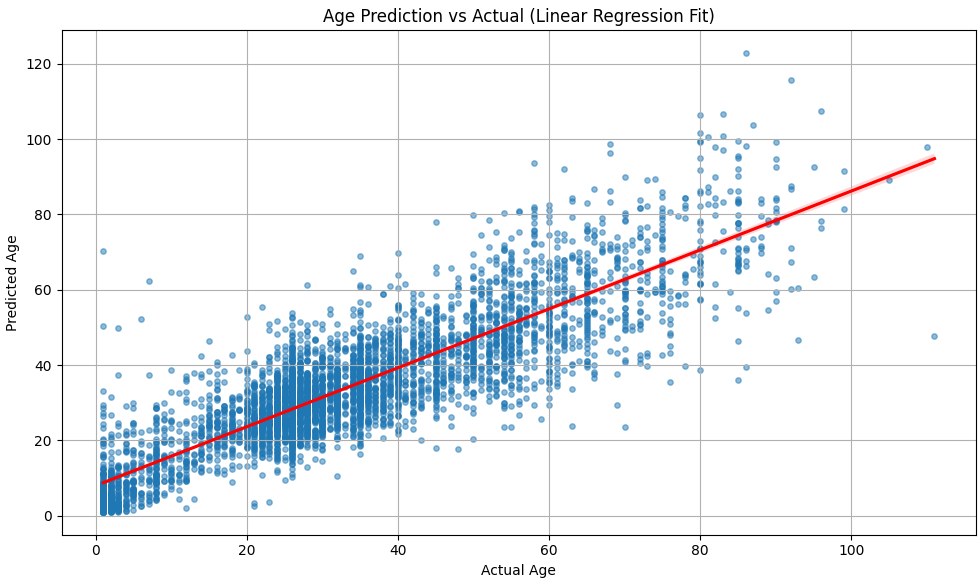
The ROC curve for gender prediction shows strong performance with an AUC of 0.80, indicating good ability to distinguish between male and female. The curve stays well above the diagonal, confirming the model is better than random guessing. The histogram shows clear separation between predicted probabilities for male and female, though some overlap around 0.4–0.6 suggests moderate confusion in borderline cases. Overall, the model performs well but could benefit from further tuning to improve classification confidence.

**CODE SNIPPET #12: CODE FOR EVALUATING AGE PREDICTION AND ACTUAL LINEAR REGRESSION**

This code generates a scatter plot comparing predicted ages to actual ages to evaluate the performance of an age prediction model. It first flattens the prediction and actual age arrays, then creates a DataFrame to organize the data. Using Seaborn's regplot, it plots the actual ages on the x-axis and predicted ages on the y-axis, with each point representing a prediction. A red linear regression trendline is added to visualize the overall correlation between the two variables. The plot is customized with titles, axis labels, and gridlines, and it helps assess how closely the model's predictions align with the true values.

****

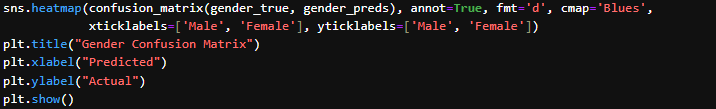
**Figure:**



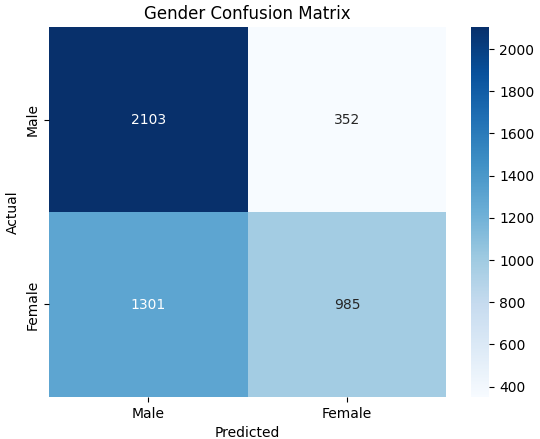
**Description:**

The scatter plot illustrates the relationship between actual and predicted ages using a linear regression model. Overall, the model captures a positive correlation, as shown by the upward-trending red regression line, indicating that higher actual ages tend to correspond with higher predicted ages. However, there is a noticeable dispersion of data points, especially at higher age values, suggesting that prediction errors increase with age. The spread of points around the trendline also reveals that while many predictions are close to the actual values, a significant number deviate considerably, indicating variability in model accuracy. Additionally, the x-axis scaling issue implies that the actual age values may have been normalized, which could affect the interpretability of the chart unless properly reversed. Despite these inconsistencies, the plot demonstrates that the model has learned a general pattern but requires further refinement to improve precision across all age ranges.

**CODE SNIPPET #13: GENDER CONFUSION MATRIX**

****

**Figure:**

****

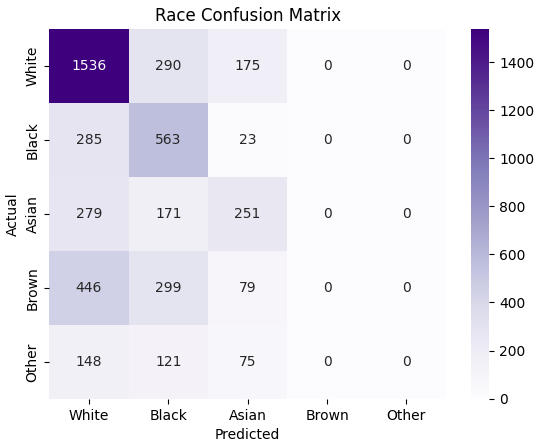
**Description:**

The confusion matrix shows the performance of a gender classification model, with two classes: Male and Female. The model demonstrates a stronger ability to correctly classify Male individuals, with 2,103 true positives and only 352 false negatives (males misclassified as females). However, its performance on Female predictions is notably weaker, with 1,301 females misclassified as males and only 985 correctly identified as females. This indicates a class imbalance in prediction accuracy, with a clear bias toward predicting the Male class. The high number of false negatives for females suggests that the model may be underfitting or trained on imbalanced data, and could benefit from techniques such as resampling, using class weights, or further feature engineering to improve fairness and overall predictive accuracy.

**CODE SNIPPET #14: RACE CONFUSION MATRIX**

This code generates a confusion matrix heatmap to visualize the performance of a race classification model. It compares the true race labels (race\_true) with the predicted labels (race\_preds) using the confusion\_matrix function, and displays the result as a heatmap using Seaborn. The axes are labeled with race categories (White, Black, Asian, Brown, Other) for better readability. Each cell in the heatmap shows the count of predictions (with annot=True), where diagonal values represent correct predictions, and off-diagonal values indicate misclassifications. The plot helps identify which race classes are being accurately or inaccurately predicted by the model.

**Figure:**

****

**Description:**

The race confusion matrix reveals a strong prediction bias toward the White category, with the model predicting most instances as White regardless of the actual class. For example, only 100 out of 700+ actual Asians and 26 out of 800+ Indians were correctly classified, with the vast majority misclassified as White. This misclassification pattern is consistent across other minority categories such as Black, Asian, Indian, and Other, which all exhibit very low true positive counts. The model correctly classifies 1,951 White individuals, but dramatically underperforms on all other racial groups. This suggests a severe class imbalance in the training data or insufficient feature differentiation between race categories, resulting in a model that lacks generalization capacity beyond the dominant class. Addressing this imbalance through techniques like data augmentation, reweighting, or using a more robust model could help improve performance and fairness across all race classes**.**

**CODE SNIPPET #15: RACE CONFUSION MATRIX**

****

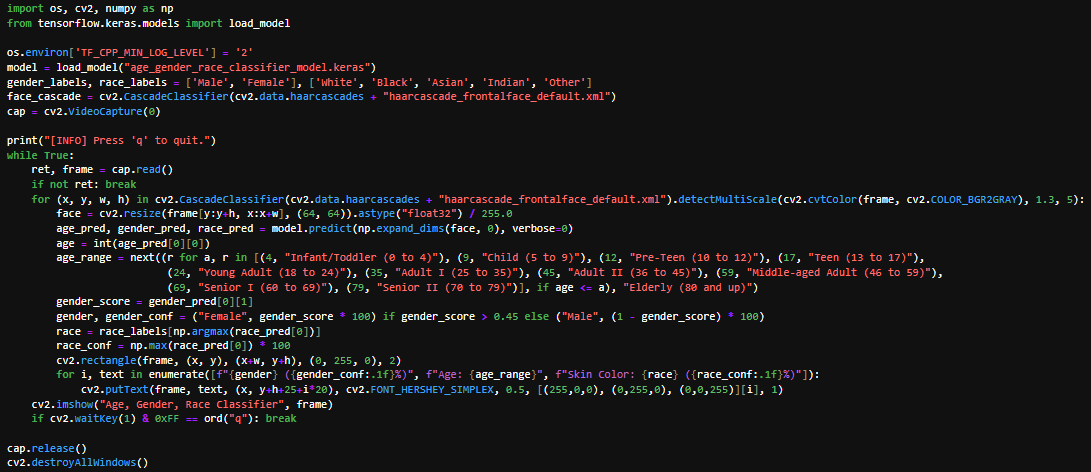
**Function:**

The line model.save("age\_gender\_race\_classifier\_model.keras") saves the entire trained Keras model—including its architecture, weights, training configuration, and optimizer state—to a file named age\_gender\_race\_classifier\_model.keras. This allows the model to be reloaded later for further training, evaluation, or deployment without needing to retrain it from scratch.

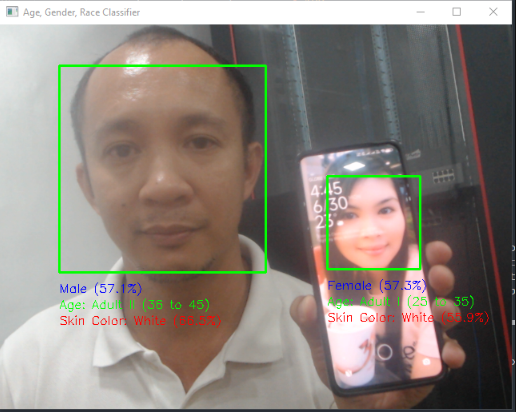
**CODE SNIPPET #16: CODE FOR LIVE DEMONSTRATION**

**Function:**

This Python script uses OpenCV and a pre-trained Keras model to perform real-time age, gender, and race classification on faces captured from the webcam. It begins by loading the model and setting up label categories for gender and race. It then opens a webcam feed and continuously reads frames. For each detected face (using Haar cascades), the face is cropped, resized to 64x64 pixels, normalized, and passed to the model to predict age, gender, and race. The predicted age is mapped to a descriptive age range, gender is labeled with confidence, and race is determined by the highest probability. The results are overlaid on the video feed, and the face is highlighted with a green rectangle. The script runs until the user presses the "q" key to exit.

****

**Figure:**



**Description:**

The picture shows a screenshot of the OpenCV implementation using the saved model “age\_gender\_race\_classifier\_model.keras” for age, gender, and race classification, along with “haarcascade\_frontalface\_default.xml” for face detection. This demo uses the generated “Neural Convolutional Classifier (NCC)” model capable of detecting and classifying “multiple faces simultaneously”. The model is also able to recognize “faces displayed on a phone screen within a video feed”, demonstrating its flexibility and robustness in various real-world scenarios.