Age Verification For Healthy Online Environment

Rachel Yu, Olivia Zhang, Jerry Shi, Alissa Xiang

Outline

1. Problem - Presented by Rachel Yu

- Our motivation and the function of our model

2. Data Processing - Presented by Alissa Xiang

- Data Repurpose
- Data Resize and Normalization
- VGG Feature Maps Extraction
- Our Own Data for Testing

3. Our Model-Presented by Jerry Shi

- Baseline model
- Primary model

4. Results - Presented by Olivia Zhang

- The Way We Measure
- Quantitative and Qualitative Results
- Comparison between Baseline and Primary Model
- Sample Predictions Using Our Own Data

If you are under 16...

Figure 1: WhatsApp [1]

R

RESTRICTED

UNDER 17 REQUIRES ACCOMPANYING
PARENT OR ADULT GUARDIAN

VIOLENCE, GRISLY IMAGES, LANGUAGE,
SOME NUDITY AND SEXUALITY

Figure 2: Restricted Movie [2]

This prompt us to develop an efficient and reliable age verification system using facial recognition technology, and uses 16yo as the threshold.

Our Purpose and Context

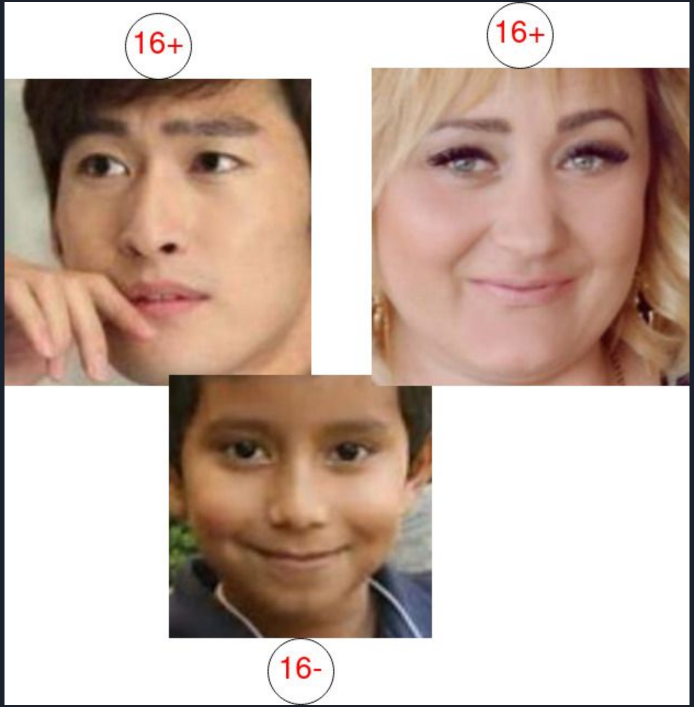
Need: Develop a facial recognition system capable of classifying individuals into two distinct categories based on age

- Two classes: above 16, below 16
- Based on a full face photo of the user

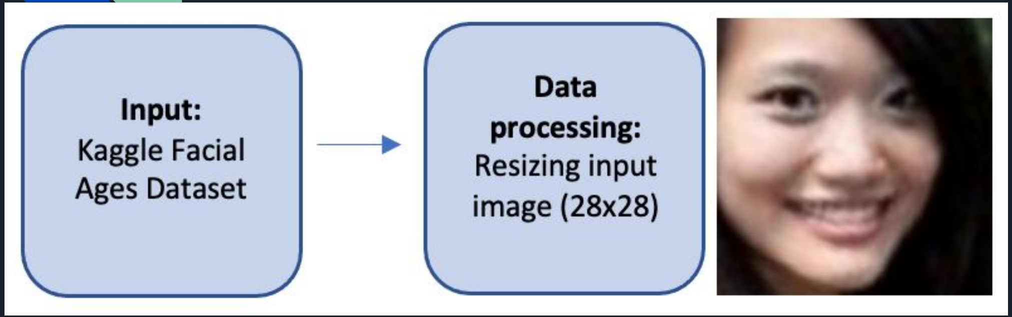
Challenge

These properties will make people look younger [3]

- Lighter skin tone
- Smiling face



So how does our model work...

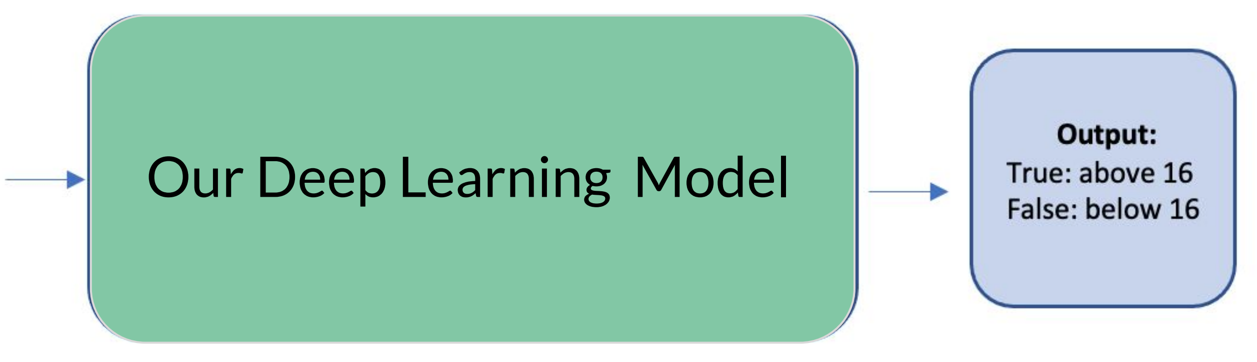


Input: Kaggle Facial Ages Dataset

Data processing: Resizing input image (28x28)

Input: full face picture of the user

Output: Boolean predicted output of whether the user is under 16 or not

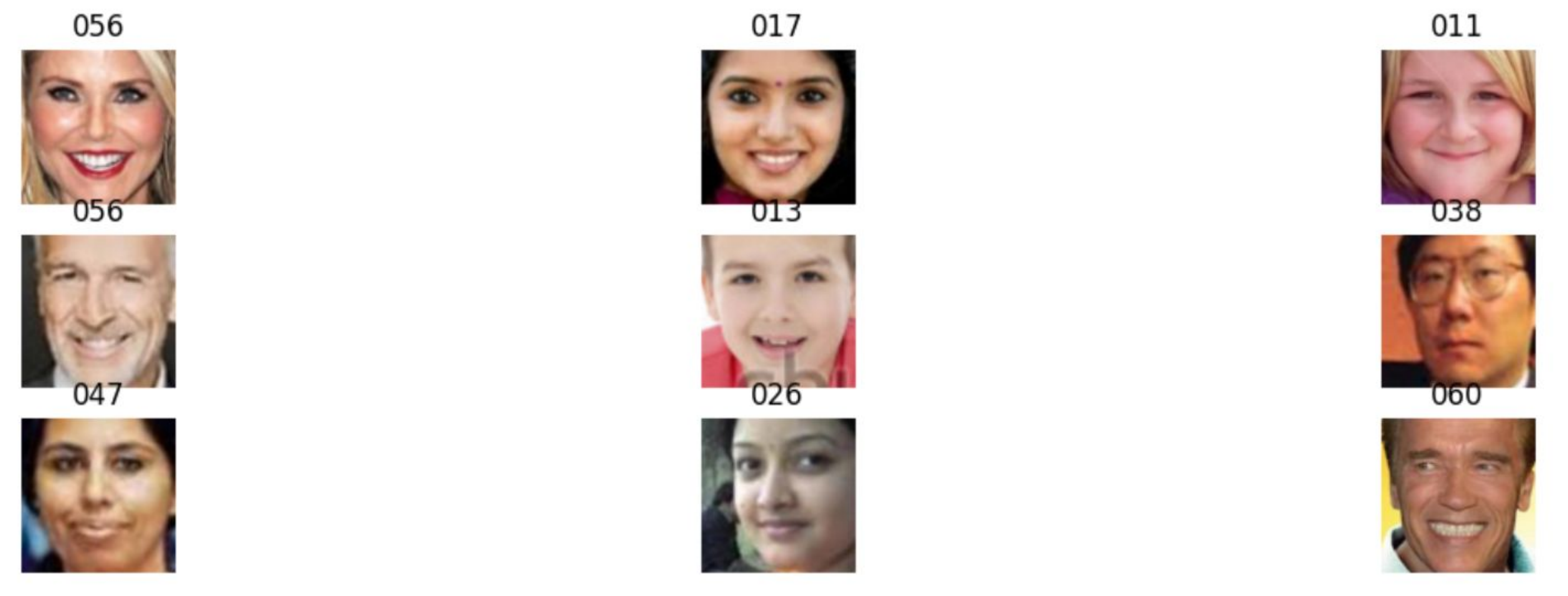


Our Deep Learning Model

Output: True: above 16 False: below 16

Data Processing

- Raw sample data from the Kaggle’s Facial Age Dataset



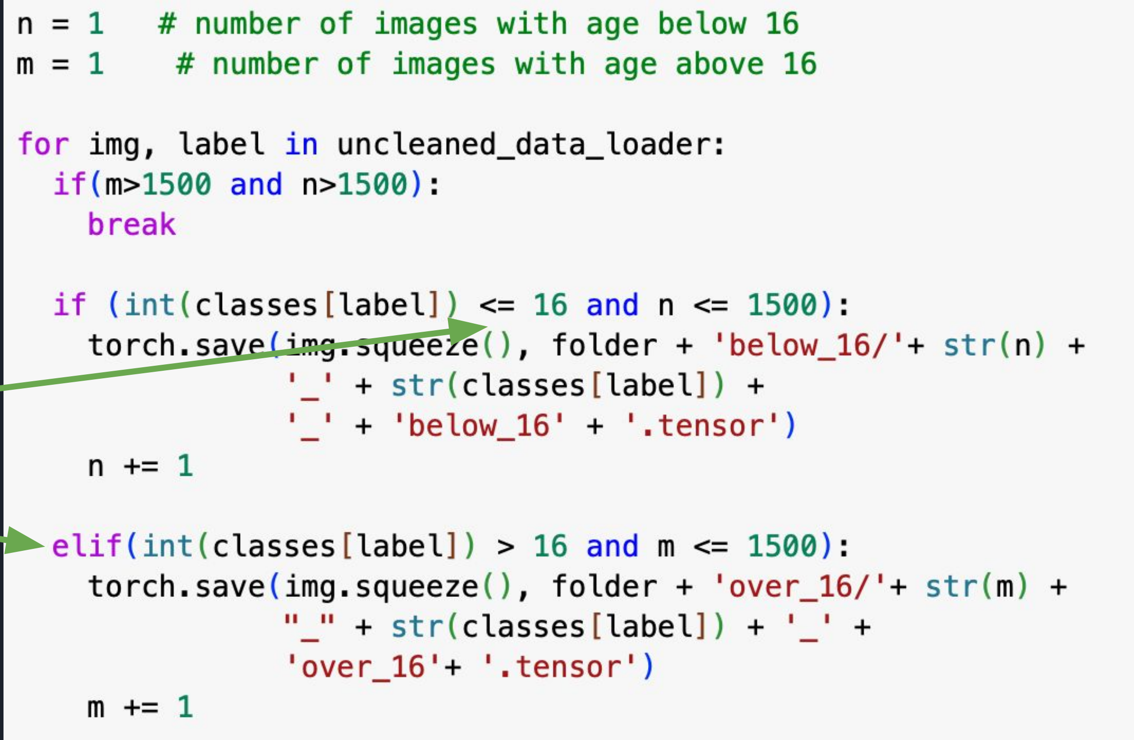
Data Repurpose

- **99 folders:** 9778 files → **53 folders**

  (manually)

- **53 folders → two classes:**
  - Below 16
  - Over 16

  (Code-based)



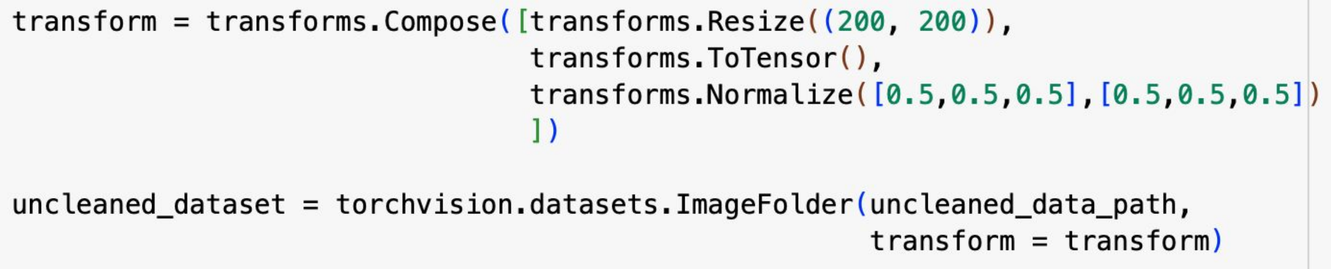
```
n = 1   # number of images with age below 16
m = 1    # number of images with age above 16

for img, label in uncleaned_data_loader:
  if(m>1500 and n>1500):
    break

  if (int(classes[label]) <= 16 and n <= 1500):
    torch.save(img.squeeze(), folder + 'below_16/'+ str(n) +
               '_' + str(classes[label]) +
               '_' + 'below_16' + '.tensor')
    n += 1

  elif(int(classes[label]) > 16 and m <= 1500):
    torch.save(img.squeeze(), folder + 'over_16/'+ str(m) +
               "_" + str(classes[label]) + '_' +
               'over_16'+ '.tensor')
    m += 1
```

Data Normalization and Resize

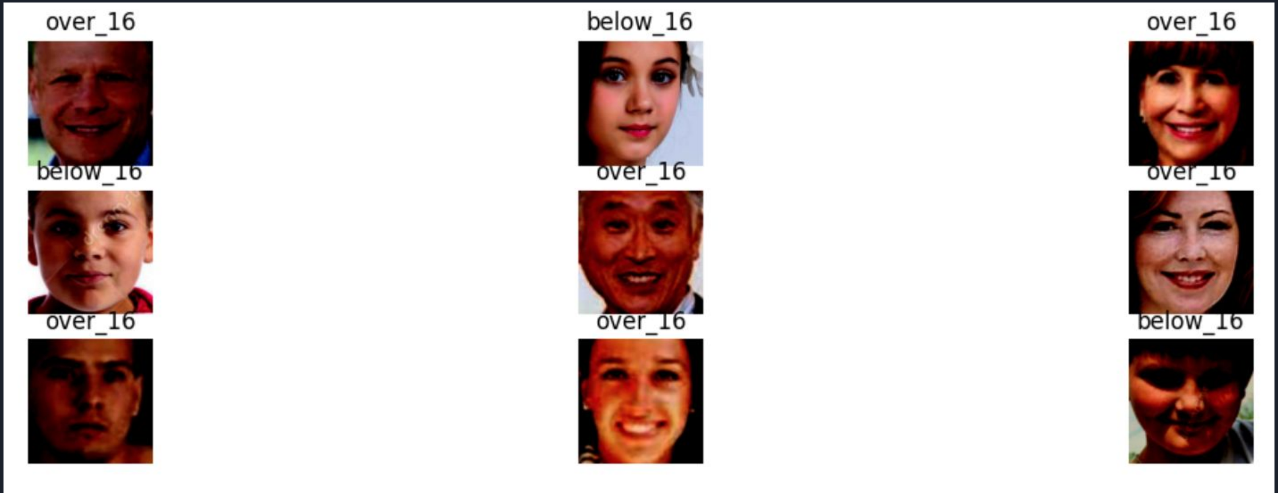


```
transform = transforms.Compose([transforms.Resize((200, 200)),
                                transforms.ToTensor(),
                                transforms.Normalize([0.5,0.5,0.5],[0.5,0.5,0.5])
                                ])

uncleaned_dataset = torchvision.datasets.ImageFolder(uncleaned_data_path,
                                                     transform = transform)
```

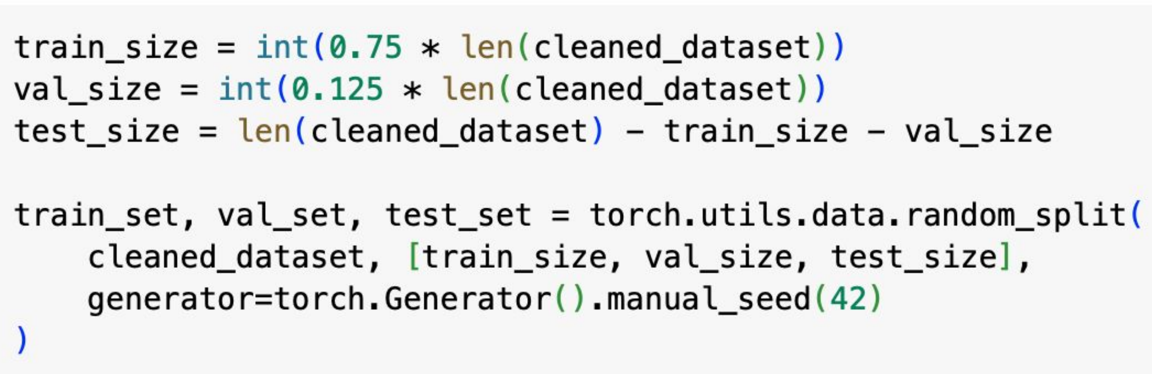
- Resize and Normalization

- Clean Sample Data



Splitting to Train, Validation and Test Sets

- **Training set:** 75% of original data
    - 2078 Training Images
- **Validation set:** 12.5% of original data
    - 346 Validation Images
- **Test set:** 12.5% of original data
    - 347 Testing Images

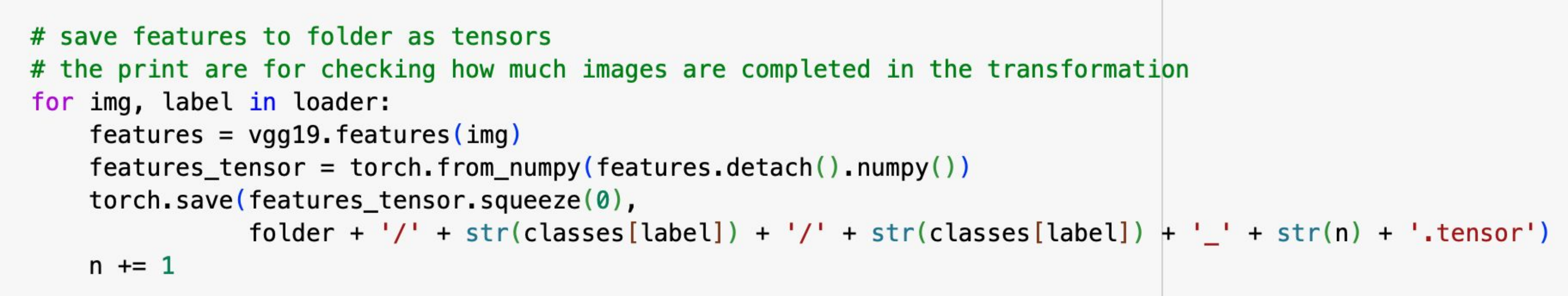


```
train_size = int(0.75 * len(cleaned_dataset))
val_size = int(0.125 * len(cleaned_dataset))
test_size = len(cleaned_dataset) - train_size - val_size

train_set, val_set, test_set = torch.utils.data.random_split(
    cleaned_dataset, [train_size, val_size, test_size],
    generator=torch.Generator().manual_seed(42)
)
```

VGG Feature Maps

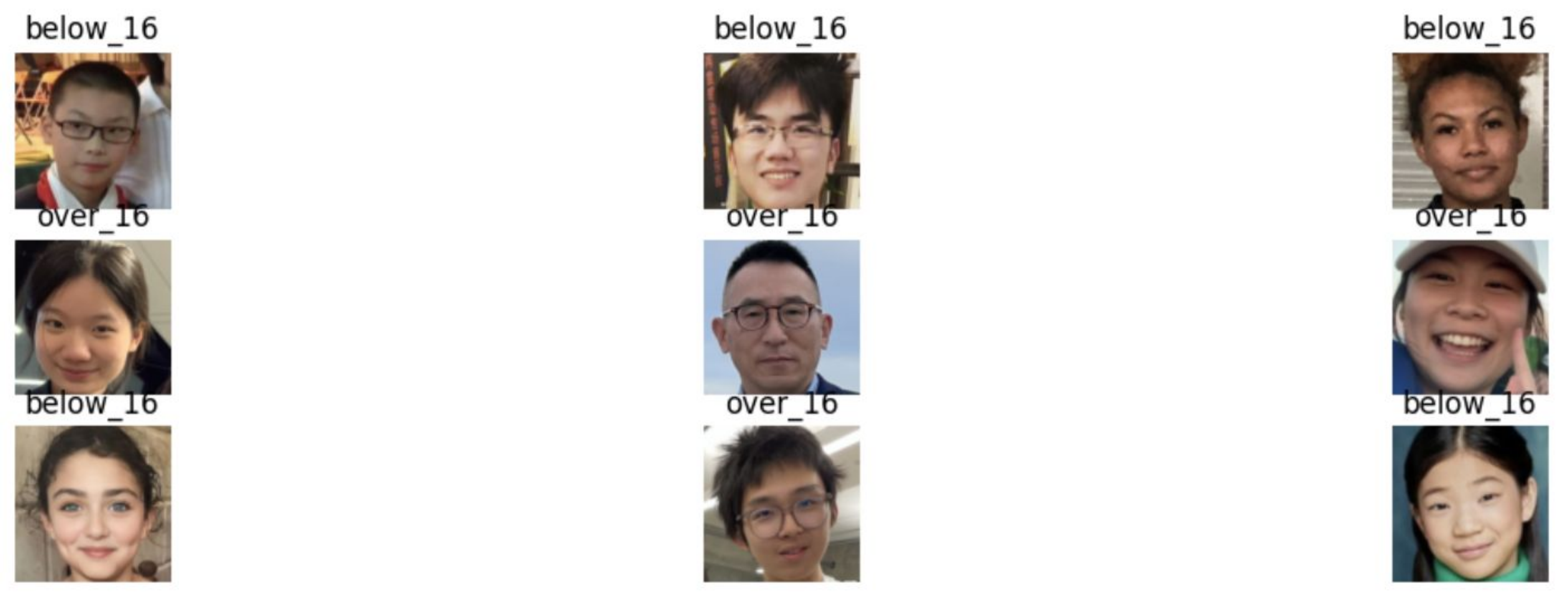
- **VGG Features extracted from datasets**
- **Saved to drive for training classifier**



```
# save features to folder as tensors
# the print are for checking how much images are completed in the transformation
for img, label in loader:
    features = vgg19.features(img)
    features_tensor = torch.from_numpy(features.detach().numpy())
    torch.save(features_tensor.squeeze(0),
               folder + '/' + str(classes[label]) + '/' + str(classes[label]) + '_' + str(n) + '.tensor')
    n += 1
```

Our Own Data

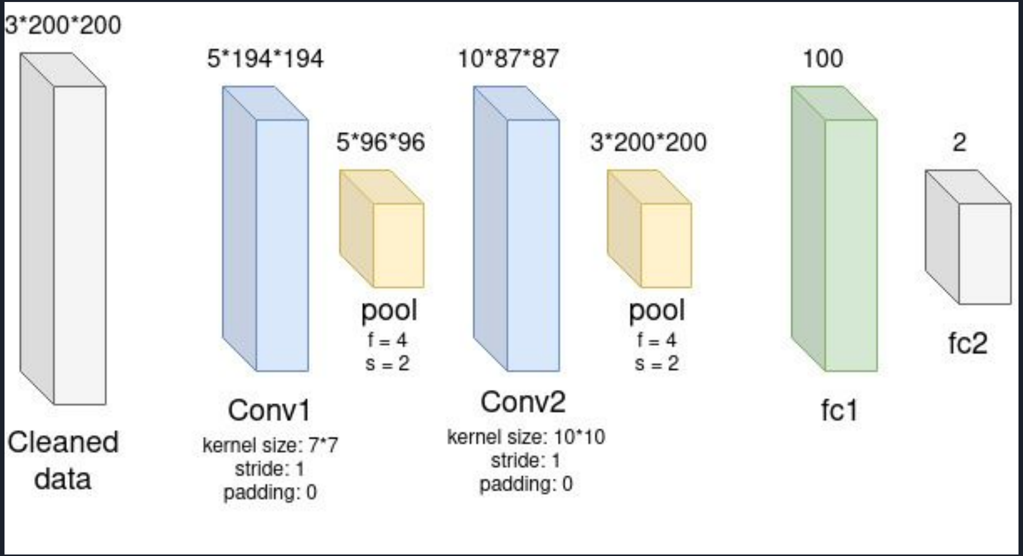
Sample data from the dataset we collected



Baseline Model

CNN_MNISTClassifier_2:

- Epoch number: 30
- Learning rate: 0.005
- Momentum: 0.4
- Batch size: 32



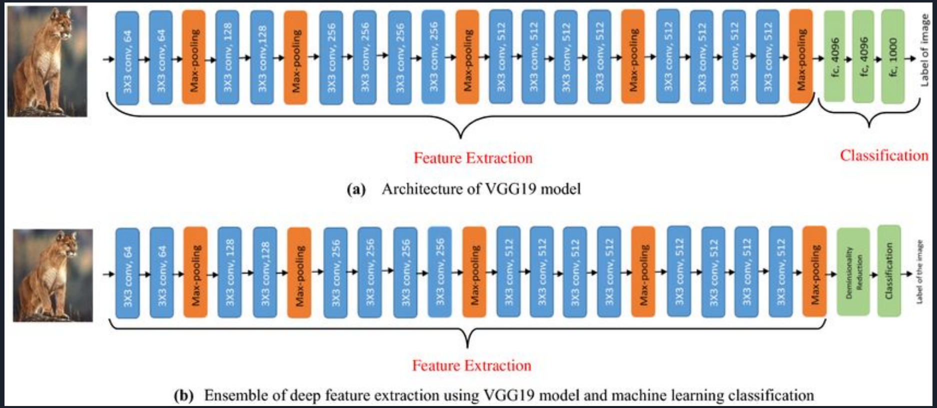
Primary Model(VGG19)

Layers: 21

Convolutional : 16

Pooling: 5

Depth: 4



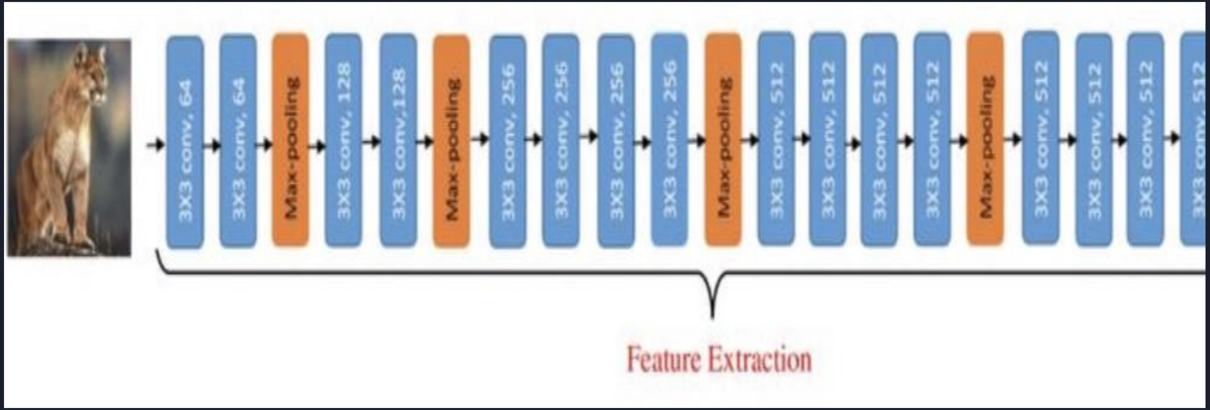
(a) Architecture of VGG19 model

(b) Ensemble of deep feature extraction using VGG19 model and machine learning classification

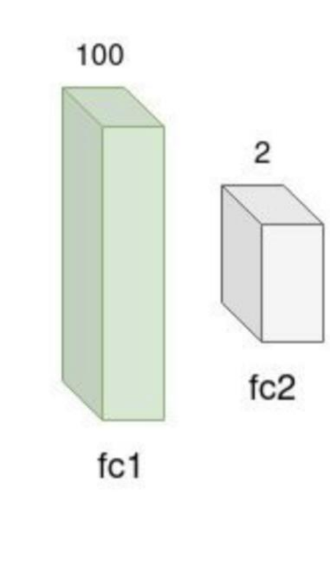
[4]

VGG19 has high accuracy on face recognition and image classification. Because of the pre-trained model has 4 different depth, hence, it could have high level generalization ability

Primary Model (Classifier)

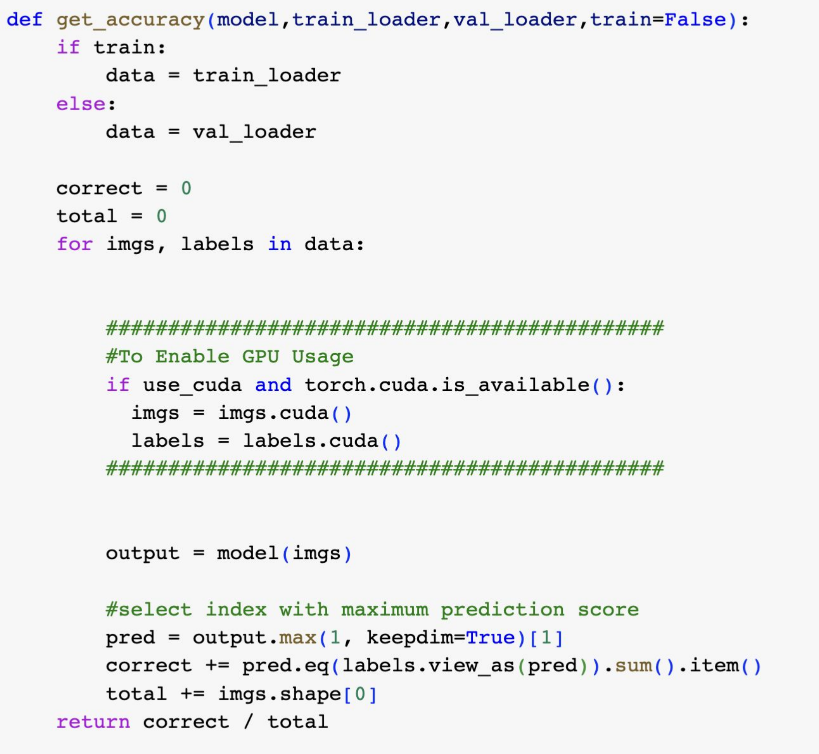


VGG(without classifier)



Classifier

Final Results - How do we measure?



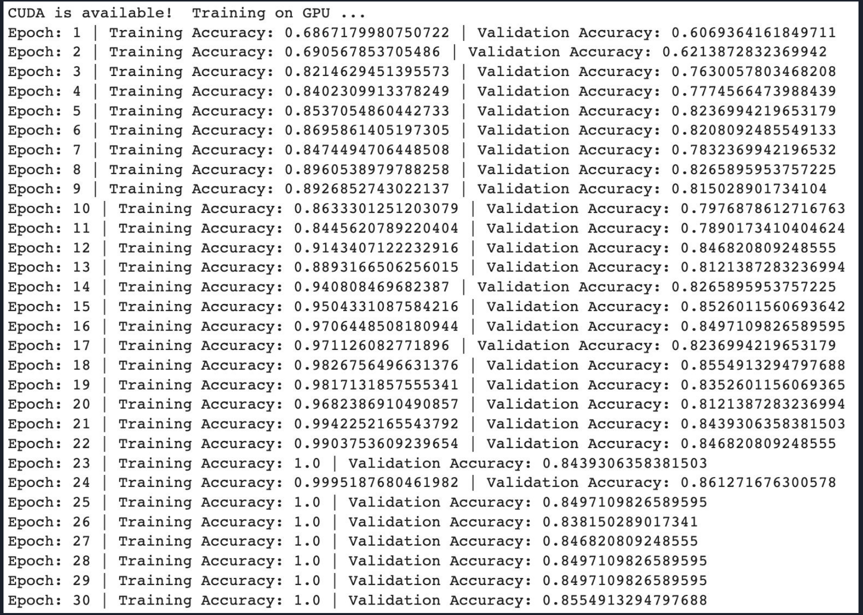
```python
def get_accuracy(model,train_loader,val_loader,train=False):
    if train:
        data = train_loader
    else:
        data = val_loader

    correct = 0
    total = 0
    for imgs, labels in data:


        #############################################
        #To Enable GPU Usage
        if use_cuda and torch.cuda.is_available():
          imgs = imgs.cuda()
          labels = labels.cuda()
        #############################################


        output = model(imgs)

        #select index with maximum prediction score
        pred = output.max(1, keepdim=True)[1]
        correct += pred.eq(labels.view_as(pred)).sum().item()
        total += imgs.shape[0]
    return correct / total
```



```
CUDA is available!  Training on GPU ...
Epoch: 1 | Training Accuracy: 0.6867179980750722 | Validation Accuracy: 0.6069364161849711
Epoch: 2 | Training Accuracy: 0.690567853705486 | Validation Accuracy: 0.6213872832369942
Epoch: 3 | Training Accuracy: 0.8214629451395573 | Validation Accuracy: 0.7630057803468208
Epoch: 4 | Training Accuracy: 0.8402309913378249 | Validation Accuracy: 0.7774566473988439
Epoch: 5 | Training Accuracy: 0.8537054860442733 | Validation Accuracy: 0.8236994219653179
Epoch: 6 | Training Accuracy: 0.8695861405197305 | Validation Accuracy: 0.8208092485549133
Epoch: 7 | Training Accuracy: 0.8474494706448508 | Validation Accuracy: 0.7832369942196532
Epoch: 8 | Training Accuracy: 0.8960538979788258 | Validation Accuracy: 0.8265895953757225
Epoch: 9 | Training Accuracy: 0.8926852743022137 | Validation Accuracy: 0.815028901734104
Epoch: 10 | Training Accuracy: 0.8633301251203079 | Validation Accuracy: 0.7976878612716763
Epoch: 11 | Training Accuracy: 0.8445620789220404 | Validation Accuracy: 0.7890173410404624
Epoch: 12 | Training Accuracy: 0.9143407122232916 | Validation Accuracy: 0.846820809248555
Epoch: 13 | Training Accuracy: 0.8893166506256015 | Validation Accuracy: 0.8121387283236994
Epoch: 14 | Training Accuracy: 0.940808469682387 | Validation Accuracy: 0.8265895953757225
Epoch: 15 | Training Accuracy: 0.9504331087584216 | Validation Accuracy: 0.8526011560693642
Epoch: 16 | Training Accuracy: 0.9706448508180944 | Validation Accuracy: 0.8497109826589595
Epoch: 17 | Training Accuracy: 0.971126082771896 | Validation Accuracy: 0.8236994219653179
Epoch: 18 | Training Accuracy: 0.9826756496631376 | Validation Accuracy: 0.8554913294797688
Epoch: 19 | Training Accuracy: 0.9817131857555341 | Validation Accuracy: 0.8352601156069365
Epoch: 20 | Training Accuracy: 0.9682386910490857 | Validation Accuracy: 0.8121387283236994
Epoch: 21 | Training Accuracy: 0.9942252165543792 | Validation Accuracy: 0.8439306358381503
Epoch: 22 | Training Accuracy: 0.9903753609239654 | Validation Accuracy: 0.846820809248555
Epoch: 23 | Training Accuracy: 1.0 | Validation Accuracy: 0.8439306358381503
Epoch: 24 | Training Accuracy: 0.9995187680461982 | Validation Accuracy: 0.861271676300578
Epoch: 25 | Training Accuracy: 1.0 | Validation Accuracy: 0.8497109826589595
Epoch: 26 | Training Accuracy: 1.0 | Validation Accuracy: 0.838150289017341
Epoch: 27 | Training Accuracy: 1.0 | Validation Accuracy: 0.846820809248555
Epoch: 28 | Training Accuracy: 1.0 | Validation Accuracy: 0.8497109826589595
Epoch: 29 | Training Accuracy: 1.0 | Validation Accuracy: 0.8497109826589595
Epoch: 30 | Training Accuracy: 1.0 | Validation Accuracy: 0.8554913294797688
```

Final Results - Quantitative & Qualitative

Training Accuracy:

- Baseline Model - 100%
- Primary Model - 96.49%

Validation Accuracy:

- **Baseline Model - 85.55%**
- **Primary Model - 87.28%**

Testing Accuracy - Kaggle's Data

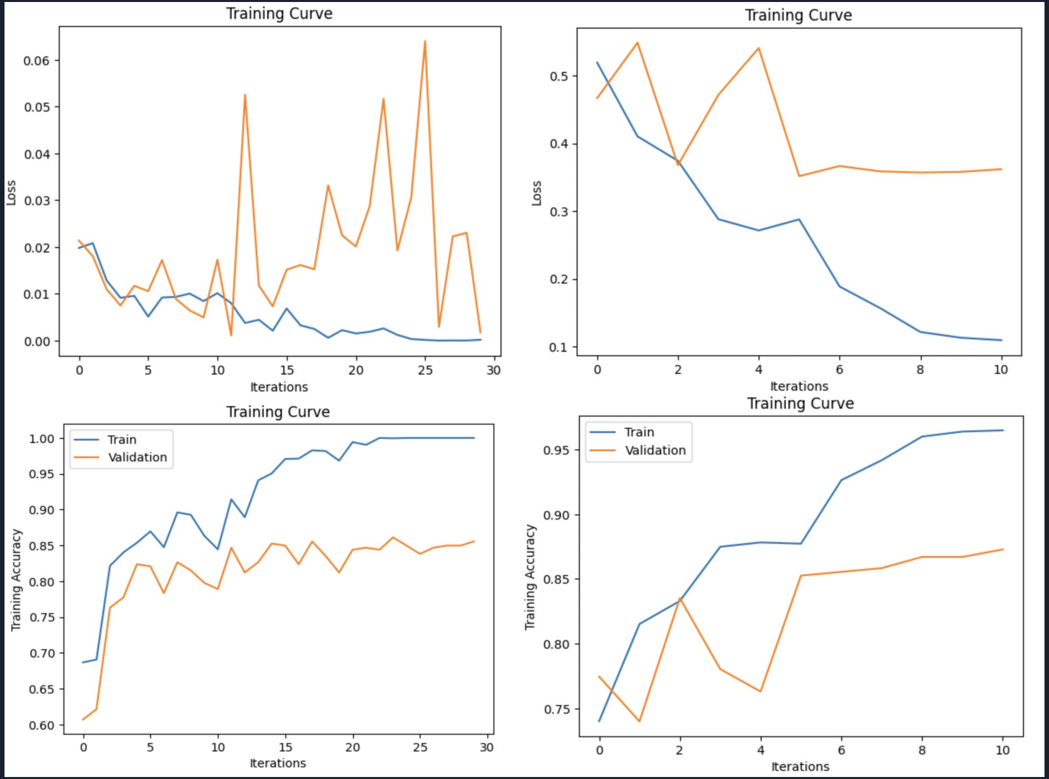
- Baseline Model - 85.3%
- Primary Model - 86.45%

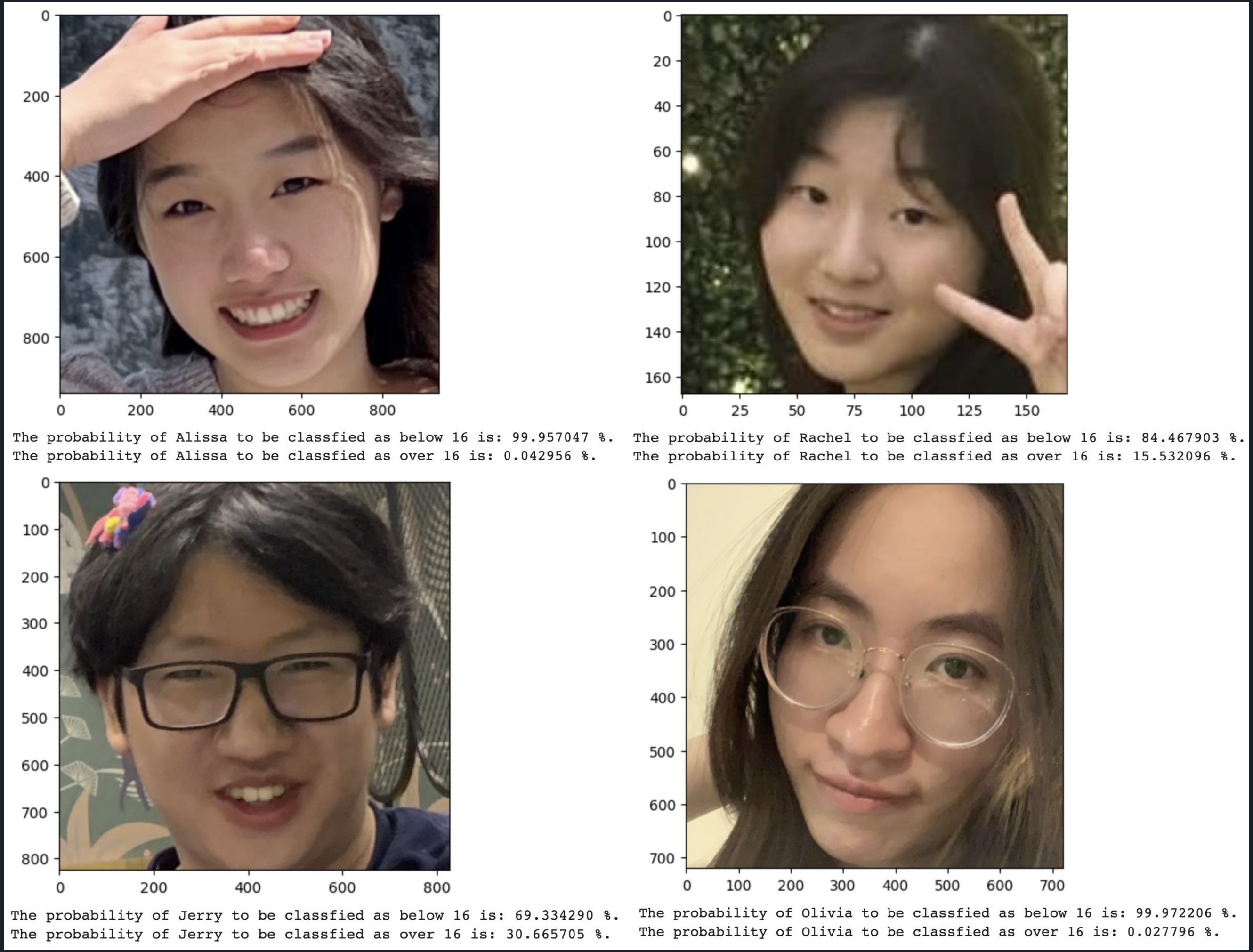
Testing Accuracy - Our Own Data

- **Baseline Model - 49.04%**
- **Primary Model - 55.77%**

Baseline

Primary





```
The probability of Alissa to be classfied as below 16 is: 99.957047 %.
The probability of Alissa to be classfied as over 16 is: 0.042956 %.
```

```
The probability of Rachel to be classfied as below 16 is: 84.467903 %.
The probability of Rachel to be classfied as over 16 is: 15.532096 %.
```

```
The probability of Jerry to be classfied as below 16 is: 69.334290 %.
The probability of Jerry to be classfied as over 16 is: 30.665705 %.
```

```
The probability of Olivia to be classfied as below 16 is: 99.972206 %.
The probability of Olivia to be classfied as over 16 is: 0.027796 %.
```

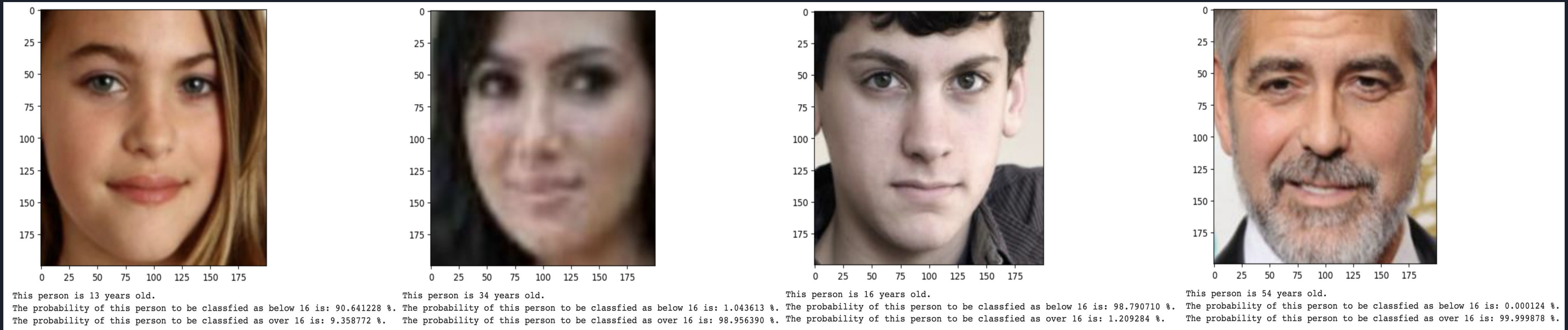
Reference

[1] “Whatsapp Review,” PCMAG, https://www.pcmag.com/reviews/whatsapp (accessed Aug. 4, 2023).

[2] A. Press, “Motion Picture Association changing its rating system to include more information on violence,” Fox News, https://www.foxnews.com/entertainment/motion-picture-association-changing-its-rating-system-to-include-more-information-on-violence (accessed Aug. 4, 2023).

[3] Yoti, https://www.yoti.com/wp-content/uploads/Yoti-Age-Estimation-White-Paper-May-2022.pdf (accessed Aug. 4, 2023).

[4] M. Bansal, M. Kumar, M. Sachdeva, and A. Mittal, “Transfer learning for image classification using VGG19: Caltech-101 Image Data Set - Journal of Ambient Intelligence and humanized computing,” SpringerLink, https://link.springer.com/article/10.1007/s12652-021-03488-z (accessed Aug. 4, 2023).



```
This person is 13 years old.
The probability of this person to be classfied as below 16 is: 90.641228 %.
The probability of this person to be classfied as over 16 is: 9.358772 %.
```

```
This person is 34 years old.
The probability of this person to be classfied as below 16 is: 1.043613 %.
The probability of this person to be classfied as over 16 is: 98.956390 %.
```

```
This person is 16 years old.
The probability of this person to be classfied as below 16 is: 98.790710 %.
The probability of this person to be classfied as over 16 is: 1.209284 %.
```

```
This person is 54 years old.
The probability of this person to be classfied as below 16 is: 0.000124 %.
The probability of this person to be classfied as over 16 is: 99.999878 %.
```