```
#importing libraries
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow.keras.layers as L
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
from sklearn.model_selection import train_test_split
```

```
!pip install opendatasets --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opendatasets in /usr/local/lib/python3.9/dist-packages (0.1.22)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from opendatasets) (8.1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from opendatasets) (4.65.0)
Requirement already satisfied: kaggle in /usr/local/lib/python3.9/dist-packages (from opendatasets) (1.5.13)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (1.26.15)
Requirement already satisfied: certifi in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (2022.12.7)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (2.27.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.9/dist-packages (from python-slugify->kaggle->opendata
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle->opendata
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle->opendatasets) (3.4)
```

```
import opendatasets as od
dataset_url="https://www.kaggle.com/datasets/nipunarora8/age-gender-and-ethnicity-face-data-csv"
od.download(dataset_url)
```

```
Skipping, found downloaded files in "./age-gender-and-ethnicity-face-data-csv" (use force=True to force download)
```

```
data = pd.read_csv('/content/age-gender-and-ethnicity-face-data-csv/age_gender.csv')
```

```
# Converting pixels into numpy array
data['pixels']=data['pixels'].apply(lambda x:  np.array(x.split(), dtype="float32"))
```

```
data.head()
```

	age	ethnicity	gender	img_name	pixels
0	1	2	0	20161219203650636.jpg.chip.jpg	[129.0, 128.0, 128.0, 126.0, 127.0, 130.0, 133...
1	1	2	0	20161219222752047.jpg.chip.jpg	[164.0, 74.0, 111.0, 168.0, 169.0, 171.0, 175....
2	1	2	0	20161219222832191.jpg.chip.jpg	[67.0, 70.0, 71.0, 70.0, 69.0, 67.0, 70.0, 79....
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[193.0, 197.0, 198.0, 200.0,

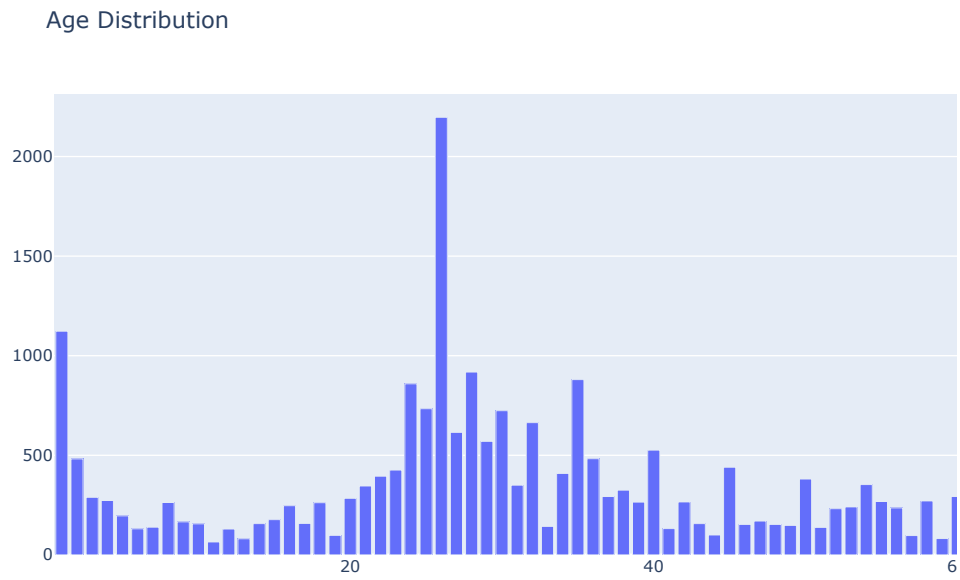
```
print('Total rows: {}'.format(len(data)))
print('Total columns: {}'.format(len(data.columns)))
```

```
Total rows: 23705
Total columns: 5
```

```
## normalizing pixels data
data['pixels'] = data['pixels'].apply(lambda x: x/255)
```

```
# calculating distributions
age_dist = data['age'].value_counts()
#ethnicity_dist = data['ethnicity'].value_counts()
gender_dist = data['gender'].value_counts().rename(index={0:'Male',1:'Female'})

def ditribution_plot(x,y,name):
    fig = go.Figure([
        go.Bar(x=x, y=y)
    ])

    fig.update_layout(title_text=name)
    fig.show()
```

```
ditribution_plot(x=age_dist.index, y=age_dist.values, name='Age Distribution')
```




```
#ditribution_plot(x=ethnicity_dist.index, y=ethnicity_dist.values, name='Ethnicity Distribution')

ditribution_plot(x=gender_dist.index, y=gender_dist.values, name='Gender Distribution')
```


```
X = np.array(data['pixels'].tolist())

## Converting pixels from 1D to 3D
X = X.reshape(X.shape[0],48,48,1)


plt.figure(figsize=(16,16))
for i in range(1500,1520):
    plt.subplot(5,5,(i%25)+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(data['pixels'].iloc[i].reshape(48,48))
    plt.xlabel(
        "Age:"+str(data['age'].iloc[i])+
       # "  Ethnicity:"+str(data['ethnicity'].iloc[i])+
        "  Gender:"+ str(data['gender'].iloc[i])
```

```
    )
plt.show()
```

Model for Gender Prediction

```
#spliting data
y = data['gender']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.22, random_state=37)


model = tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    L.BatchNormalization(),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(64, (3, 3), activation='relu'),
    L.MaxPooling2D((2, 2)),
    L.Flatten(),
    L.Dense(64, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1, activation='sigmoid')
])

model.compile(optimizer='sgd',
```

```
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])


## Stop training when validation loss reach 0.2700
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_loss')<0.2700):
            print("\nReached 0.2700 val_loss so cancelling training!")
            self.model.stop_training = True

callback = myCallback()

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 46, 46, 32)        320

 batch_normalization (BatchN  (None, 46, 46, 32)       128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 23, 23, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 64)                409664

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 428,673
Trainable params: 428,609
Non-trainable params: 64
_________________________________________________________________
```

```
#for sgd
history = model.fit(
    X_train, y_train, epochs=20, validation_split=0.1, batch_size=64, callbacks=[callback])
```

```
Epoch 1/20
260/260 [==============================] - 67s 253ms/step - loss: 0.5340 - accuracy: 0.7284 - val_loss: 0.5979 - val_accuracy: 0.77
Epoch 2/20
260/260 [==============================] - 64s 245ms/step - loss: 0.4068 - accuracy: 0.8177 - val_loss: 0.4044 - val_accuracy: 0.83
Epoch 3/20
260/260 [==============================] - 66s 252ms/step - loss: 0.3614 - accuracy: 0.8350 - val_loss: 0.3633 - val_accuracy: 0.84
Epoch 4/20
260/260 [==============================] - 65s 249ms/step - loss: 0.3345 - accuracy: 0.8513 - val_loss: 0.3244 - val_accuracy: 0.85
Epoch 5/20
260/260 [==============================] - 73s 279ms/step - loss: 0.3176 - accuracy: 0.8599 - val_loss: 0.3141 - val_accuracy: 0.85
Epoch 6/20
260/260 [==============================] - 66s 255ms/step - loss: 0.3063 - accuracy: 0.8657 - val_loss: 0.3155 - val_accuracy: 0.85
Epoch 7/20
260/260 [==============================] - 66s 254ms/step - loss: 0.2903 - accuracy: 0.8736 - val_loss: 0.3139 - val_accuracy: 0.86
Epoch 8/20
260/260 [==============================] - 66s 256ms/step - loss: 0.2828 - accuracy: 0.8765 - val_loss: 0.2912 - val_accuracy: 0.87
Epoch 9/20
260/260 [==============================] - 66s 254ms/step - loss: 0.2734 - accuracy: 0.8822 - val_loss: 0.2905 - val_accuracy: 0.87
Epoch 10/20
260/260 [==============================] - 66s 254ms/step - loss: 0.2630 - accuracy: 0.8868 - val_loss: 0.2825 - val_accuracy: 0.87
Epoch 11/20
260/260 [==============================] - 65s 251ms/step - loss: 0.2565 - accuracy: 0.8906 - val_loss: 0.2777 - val_accuracy: 0.87
Epoch 12/20
260/260 [==============================] - 66s 254ms/step - loss: 0.2488 - accuracy: 0.8941 - val_loss: 0.2712 - val_accuracy: 0.88
Epoch 13/20
260/260 [==============================] - 65s 249ms/step - loss: 0.2438 - accuracy: 0.8976 - val_loss: 0.2902 - val_accuracy: 0.86
Epoch 14/20
260/260 [==============================] - ETA: 0s - loss: 0.2377 - accuracy: 0.9001
Reached 0.2700 val_loss so cancelling training!
260/260 [==============================] - 66s 255ms/step - loss: 0.2377 - accuracy: 0.9001 - val_loss: 0.2699 - val_accuracy: 0.88
```

```
from tensorflow.keras.optimizers import Adagrad
model = tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
```

```
    L.BatchNormalization(),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(64, (3, 3), activation='relu'),
    L.MaxPooling2D((2, 2)),
    L.Flatten(),
    L.Dense(64, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1, activation='sigmoid')
])
optimizer = Adagrad(learning_rate=0.01)

model.compile(optimizer=optimizer,
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])
history_ada = model.fit(
    X_train, y_train, epochs=20, validation_split=0.1, batch_size=64, callbacks=[callback])
```

```
Epoch 1/20
260/260 [==============================] - ETA: 0s - loss: -125047384.0000 - accuracy: 0.0481
Reached 110 val_loss so cancelling training!
260/260 [==============================] - 65s 245ms/step - loss: -125047384.0000 - accuracy: 0.0481 - val_loss: -131382672.0000 -
```

```
learning_rate = 0.001
beta1 = 0.9
beta2 = 0.999
epsilon = 1e-8

#optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, beta_1=beta1, beta_2=beta2, epsilon=epsilon)

model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])
history = model.fit(
    X_train, y_train, epochs=20, validation_split=0.1, batch_size=64, callbacks=[callback])
```

```
Epoch 1/20
260/260 [==============================] - 68s 256ms/step - loss: 0.2979 - accuracy: 0.8710 - val_loss: 0.2764 - val_accuracy: 0.88
Epoch 2/20
260/260 [==============================] - 66s 253ms/step - loss: 0.2641 - accuracy: 0.8861 - val_loss: 0.2838 - val_accuracy: 0.87
Epoch 3/20
260/260 [==============================] - 66s 253ms/step - loss: 0.2451 - accuracy: 0.8953 - val_loss: 0.2887 - val_accuracy: 0.88
Epoch 4/20
260/260 [==============================] - ETA: 0s - loss: 0.2208 - accuracy: 0.9070
Reached 0.2700 val_loss so cancelling training!
260/260 [==============================] - 65s 252ms/step - loss: 0.2208 - accuracy: 0.9070 - val_loss: 0.2692 - val_accuracy: 0.89
```

```
#Evaluate training history
fig = px.line(
    history.history, y=['loss', 'val_loss'],
    labels={'index': 'epoch', 'value': 'loss'},
    title='Training History')
fig.show()
```

Training History

```
mse, mae = model.evaluate(X_test,y_test,verbose=0)
print('Test Mean squared error: {}'.format(mse))
print('Test Mean absolute error: {}'.format(mae))
```

```
Test Mean squared error: 0.2536311149597168
Test Mean absolute error: 0.8861196041107178
```

Model for Age prediction

```
y= data['age']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.22, random_state=37
)
```

```
model = tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    L.BatchNormalization(),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(64, (3, 3), activation='relu'),
    L.MaxPooling2D((2, 2)),
    L.Conv2D(128, (3, 3), activation='relu'),
    L.MaxPooling2D((2, 2)),
    L.Flatten(),
    L.Dense(64, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1, activation='relu')
])

sgd = tf.keras.optimizers.SGD(momentum=0.9)
model.compile(optimizer='adam',
              loss='mean_squared_error',
              metrics=['mae'])


## Stop training when validation loss reach 110
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_loss')<110):
            print("\nReached 110 val_loss so cancelling training!")
            self.model.stop_training = True

callback = myCallback()


model.summary()
```

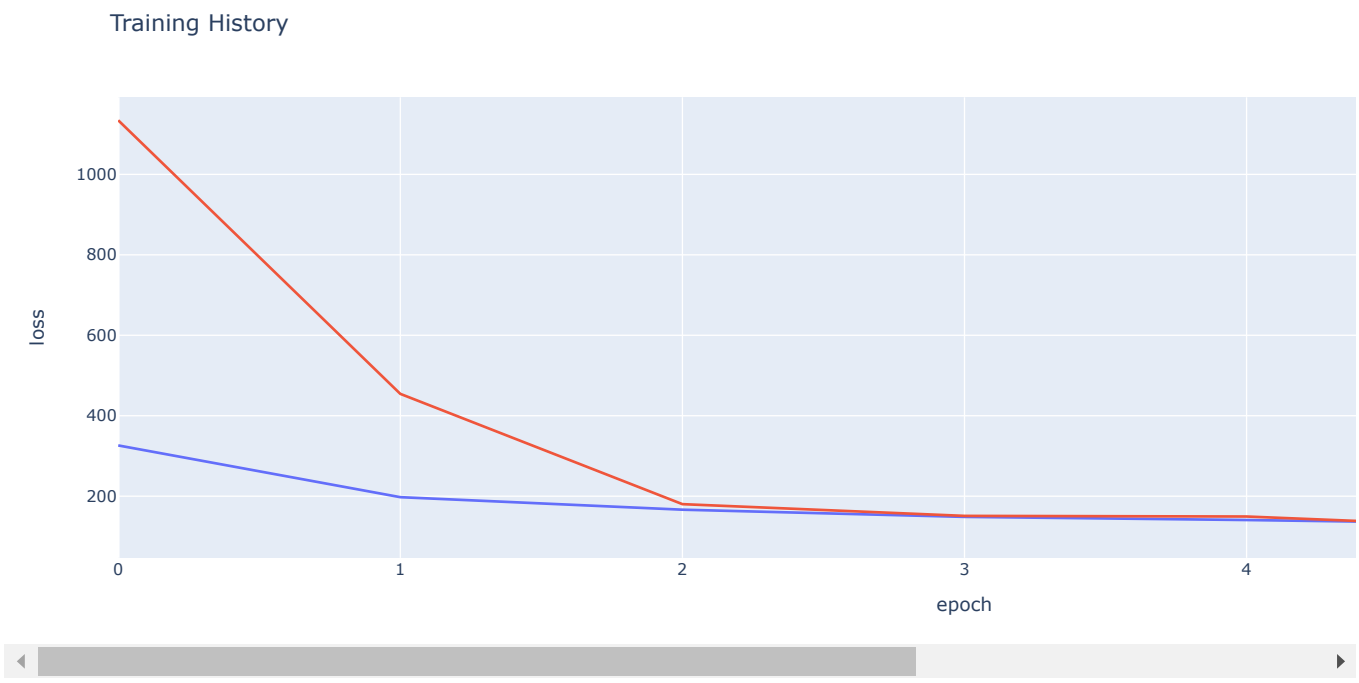
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 46, 46, 32)        320

 batch_normalization_2 (Batc  (None, 46, 46, 32)       128
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 23, 23, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 8, 8, 128)         73856

 max_pooling2d_6 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dense_4 (Dense)             (None, 64)                131136

 dropout_2 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 224,001
```

```
Trainable params: 223,937
Non-trainable params: 64
_________________________________________________________________
```

```
history = model.fit(
    X_train, y_train, epochs=20, validation_split=0.1, batch_size=64, callbacks=[callback]
)
```

```
Epoch 1/20
260/260 [==============================] - 83s 315ms/step - loss: 326.1502 - mae: 13.7014 - val_loss: 1134.4440 - val_mae: 28.0298
Epoch 2/20
260/260 [==============================] - 79s 304ms/step - loss: 197.5288 - mae: 10.6292 - val_loss: 454.3263 - val_mae: 16.2759
Epoch 3/20
260/260 [==============================] - 79s 304ms/step - loss: 166.4010 - mae: 9.6543 - val_loss: 180.1934 - val_mae: 9.7002
Epoch 4/20
260/260 [==============================] - 79s 303ms/step - loss: 148.5289 - mae: 9.0606 - val_loss: 151.0116 - val_mae: 9.6886
Epoch 5/20
260/260 [==============================] - 79s 302ms/step - loss: 140.7376 - mae: 8.7289 - val_loss: 149.4542 - val_mae: 9.7187
Epoch 6/20
260/260 [==============================] - 79s 303ms/step - loss: 130.9966 - mae: 8.4501 - val_loss: 121.3183 - val_mae: 7.7990
Epoch 7/20
260/260 [==============================] - ETA: 0s - loss: 125.3670 - mae: 8.2420
Reached 110 val_loss so cancelling training!
260/260 [==============================] - 79s 303ms/step - loss: 125.3670 - mae: 8.2420 - val_loss: 103.8825 - val_mae: 7.5552
```

```
#evaluating taring history
fig = px.line(
    history.history, y=['loss', 'val_loss'],
    labels={'index': 'epoch', 'value': 'loss'},
    title='Training History')
fig.show()
```




```
mse, mae = model.evaluate(X_test,y_test,verbose=0)
print('Test Mean squared error: {}'.format(mse))
print('Test Mean absolute error: {}'.format(mae))
```

```
Test Mean squared error: 95.03944396972656
Test Mean absolute error: 7.308180332183838
```