DL LAB PROJECT *REPORT*

-Recognising gender and age using audio files

M.DHANUSH VARMA
220968082
DSE-A 16

IMPORTING LIBRARIES.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import librosa
import librosa.display

from sklearn.preprocessing import MinMaxScaler
from tqdm import tqdm, notebook,trange
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score as acc_score
from sklearn.metrics import recall_score as rec_score
from sklearn.metrics import precision_score as prec_score
from sklearn.metrics import f1_score as f_scores
from sklearn.metrics import confusion_matrix
```

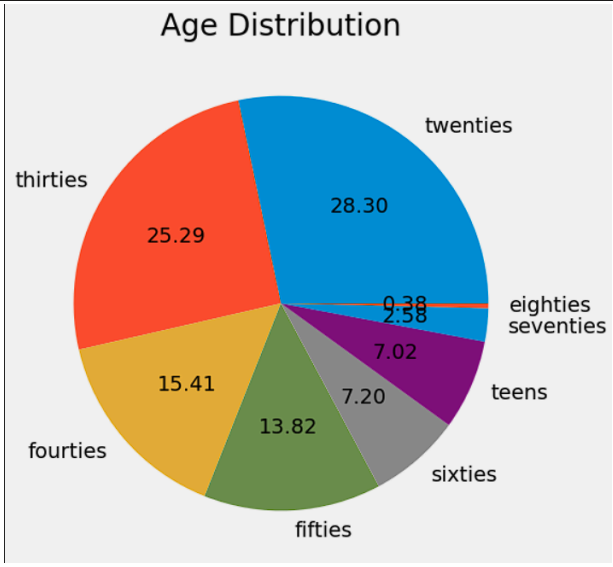
READING DATASET

```
train_data = pd.read_csv("cv-valid-train.csv")
```

```
train_data.head()
```

	filename	text	up_votes	down_votes	age	gender	accent	duration
0	cv-valid-train/sample-000000.mp3	learn to recognize omens and follow them the o...	1	0	NaN	NaN	NaN	NaN
1	cv-valid-train/sample-000001.mp3	everything in the universe evolved he said	1	0	NaN	NaN	NaN	NaN
2	cv-valid-train/sample-000002.mp3	you came so that you could learn about your dr...	1	0	NaN	NaN	NaN	NaN
3	cv-valid-train/sample-000003.mp3	so now i fear nothing because it was those ome...	1	0	NaN	NaN	NaN	NaN
4	cv-valid-train/sample-000004.mp3	if you start your emails with greetings let me...	3	2	NaN	NaN	NaN	NaN

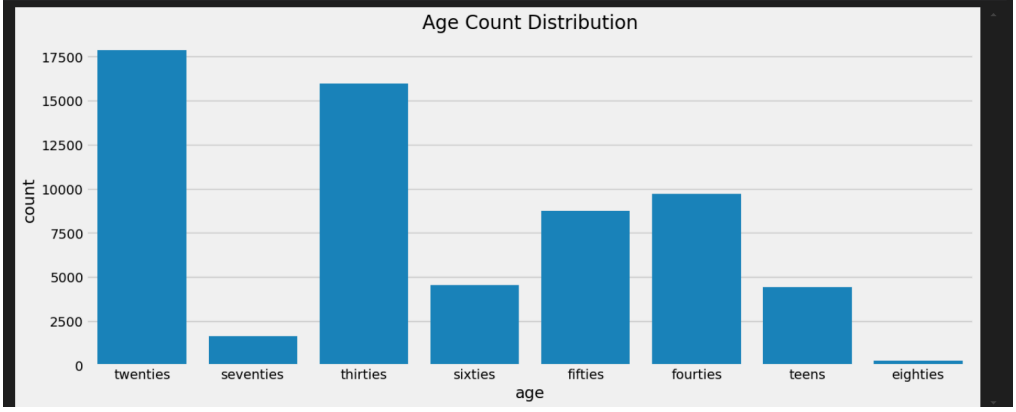
```
plt.title("Age Distribution")
plt.pie(train_data.age.value_counts().values,labels=train_data.age.value_c
ounts().index,autopct="%0.2f")
plt.show()
```



Age Count Distribution

```python
plt.title("Age Count Distribution")
sns.countplot(x=train_data.age)
plt.show()
```

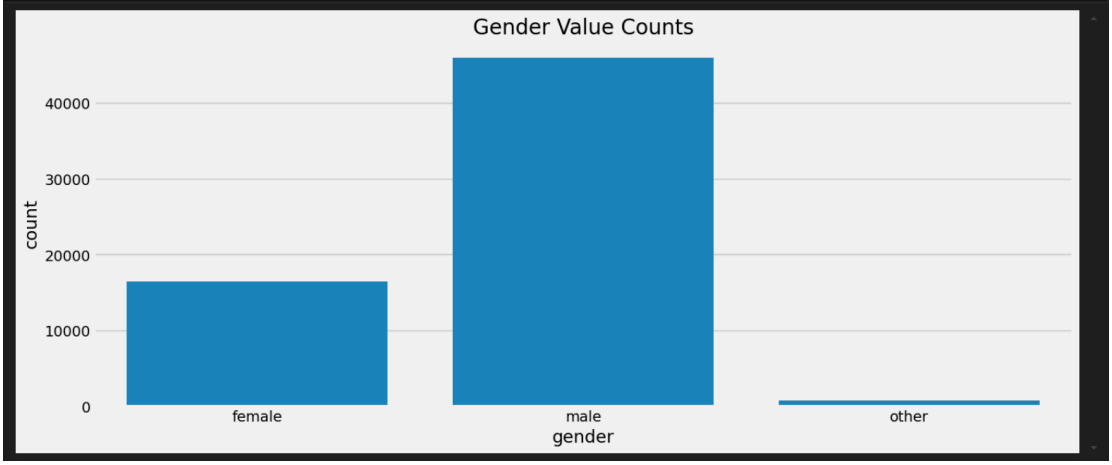
Python



Gender Count Values

```python
plt.title("Gender Value Counts")
sns.countplot(x=train_data.gender)
plt.show()
```

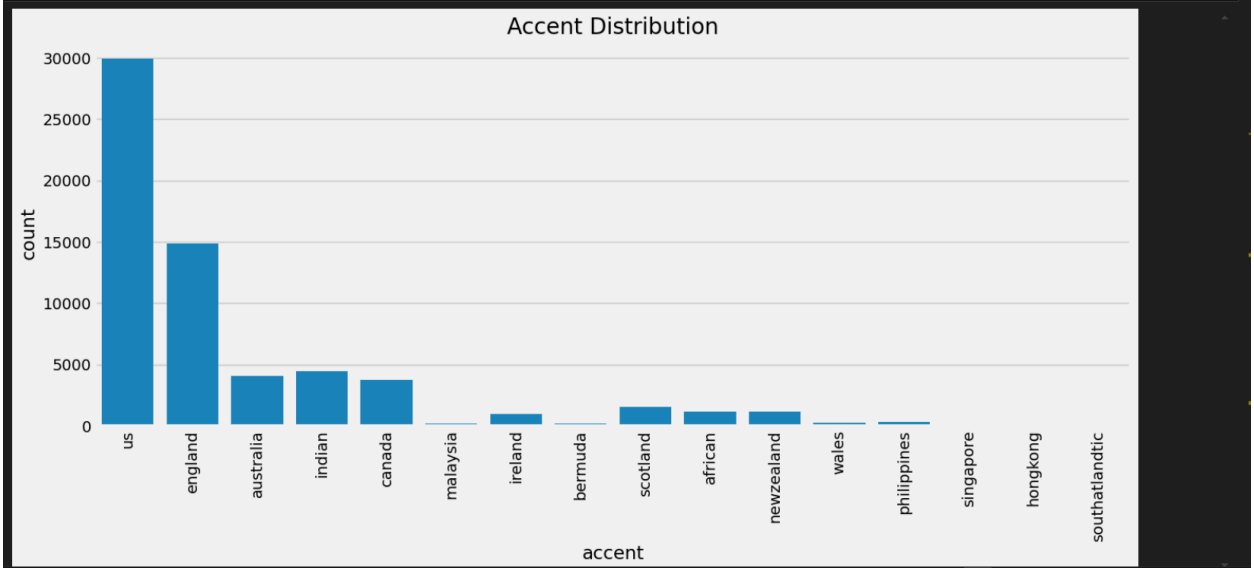
Python



Accent Distribution

```python
plt.title("Accent Distribution")
sns.countplot(x=train_data.accent)
plt.xticks(rotation=90)
plt.show()
```

Python




```python
accent_to_keep = train_data.accent.value_counts().index[0:9]

idx = []
for i in range(train_data.shape[0]):
    if train_data.accent[i] in accent_to_keep:
        idx.append(i)
```

Python

```python
train_data = train_data.loc[idx]
train_data.reset_index(inplace=True)
train_data.drop("index",axis=1,inplace=True)
```

Python

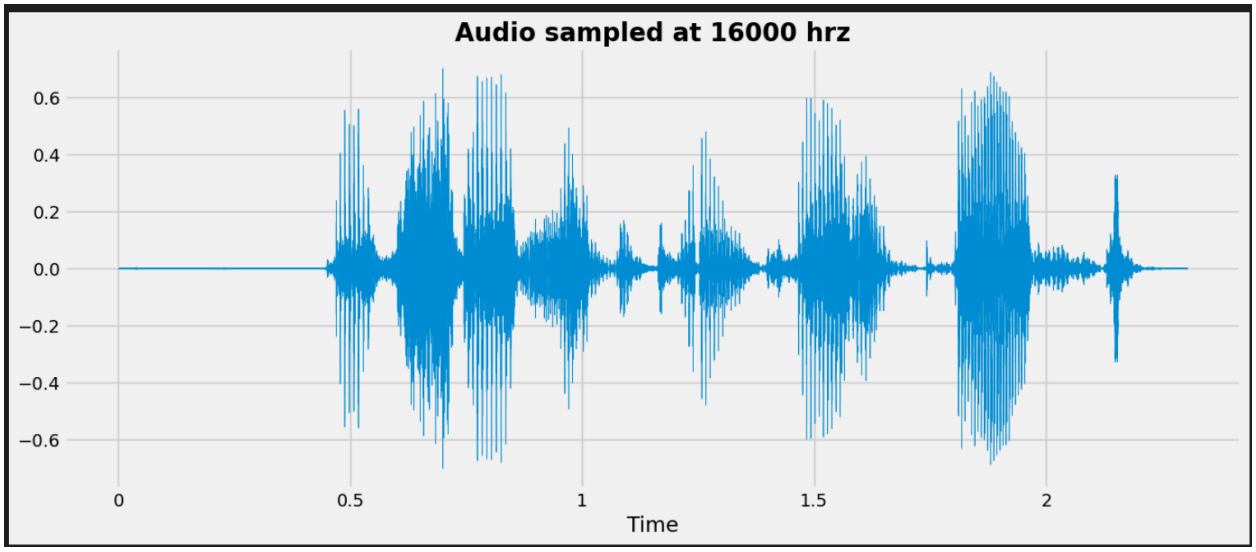
Visualizing Some features of Audio wave

Visualizing some features of Audio wave

```
voice_dir = "cv-valid-train-extracted"
sample_rate = 16000
```

```
# Taking Random voice sample
idx = np.random.randint(0,len(train_data))
# storing idx'th voice fiel name
filename = voice_dir +"/"+train_data.filename[idx]
```

```
# Loading audio sample
audio_array, sampling_rate = librosa.load(filename, sr=sample_rate)
# Creating figure
plt.figure(figsize=(12, 3))
plt.figure()
librosa.display.waveshow(audio_array, sr=sampling_rate)
plt.title('Audio sampled at {} hrz'.format(sample_rate),fontsize = 20, loc='center', fontdict=dict(weight='bold'))
plt.show()
```



Mel-Frequency Cepstral Coefficient

Mel-Frequency Cepstral Coefficient

```python
# Loading Audio sample
audio_array,sampling_rate = librosa.load(filename, sr=sample_rate)
# MFCC with 10 components
mfcc = librosa.feature.mfcc(y=audio_array, sr=sample_rate, n_mfcc = 10)

plt.figure(figsize=(20,5))
librosa.display.specshow(mfcc)
plt.ylabel('MFCC',fontsize = 20, loc='center', fontdict=dict(weight='bold'))
plt.colorbar()
plt.show()
```

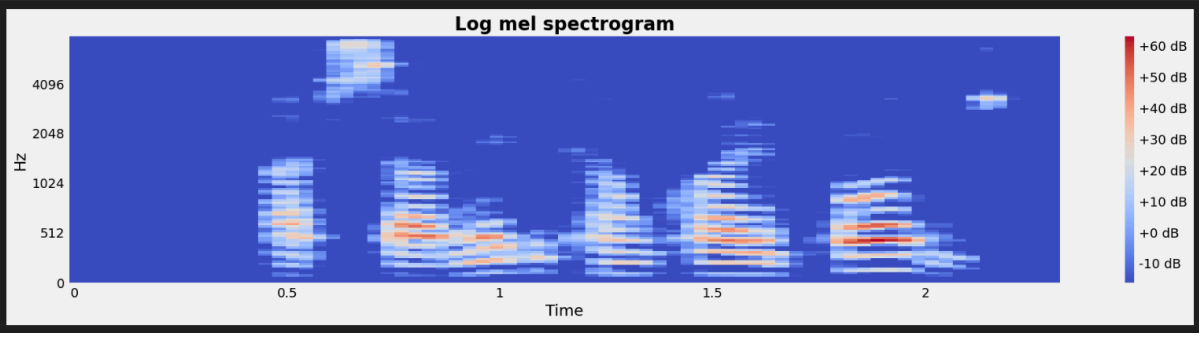
Python

Log Mel-spectogram

Log Mel-spectogram

```python
audio_array, sampling_rate = librosa.load(filename, sr=sample_rate)
mel_spectogram = librosa.feature.melspectrogram(y=audio_array, sr=sampling_rate, n_mels=128,fmax=8000)

# Convert to log scale
log_S = librosa.amplitude_to_db(mel_spectogram)

# Plotting Log Mel-Spectogram
plt.figure(figsize=(20,5))
librosa.display.specshow(log_S, sr=sampling_rate, x_axis='time', y_axis='mel')
plt.title('Log mel spectrogram',fontsize = 20, loc='center', fontdict=dict(weight='bold'))
plt.colorbar(format='%+02.0f dB')
plt.tight_layout()
```

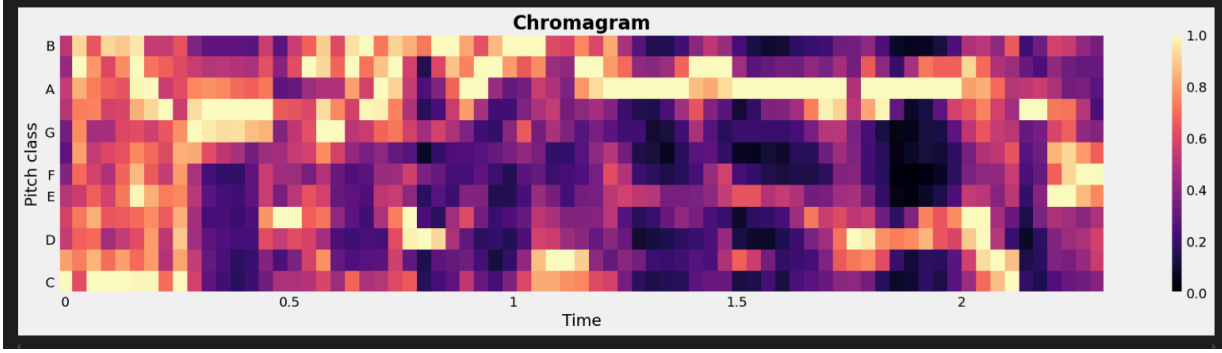
Python



Chroma

```python
audio_array, sampling_rate = librosa.load(filename, sr=sample_rate)
chroma = librosa.feature.chroma_cqt(y=audio_array, sr=sampling_rate)


plt.figure(figsize=(20,5))

librosa.display.specshow(chroma, sr=sampling_rate, x_axis='time', y_axis='chroma', vmin=0, vmax=1)
plt.title('Chromagram',fontsize = 20, loc='center', fontdict=dict(weight='bold'))
plt.colorbar()
plt.tight_layout()
```

Python



Audio Preprocessing

```python
scaler = MinMaxScaler()
```

```python
# making Directories to store processed data
os.mkdir("./age_data")
os.mkdir("./gender_data")
os.mkdir("./accent_data")
```

```python
def process_data(file,target_dir):
    filename = voice_dir +  "/" + file
    y, s = librosa.load(filename, sr=16000)
    y_filt = librosa.effects.preemphasis(y)
    S_preemph = librosa.amplitude_to_db(np.abs(librosa.stft(y_filt)), ref=np.max)
    S_preemph = scaler.fit_transform(S_preemph)
    #librosa.display.specshow(S_preemph, y_axis='log', x_axis='time')
    plt.imshow(S_preemph.T,cmap='plasma')
    plt.axis("off")
    file = file.split("/")[1]
    address = target_dir+"/"+"{}.png".format(file)
    plt.savefig(address)
    plt.close()
```

Storing Gender data

```
idx = []
cnt_male,cnt_female = 0,0
num_samples = 500
for i in range(train_data.shape[0]):
    if train_data.gender[i] == "male":
        if cnt_male <= num_samples:
            cnt_male +=1
            idx.append(i)

    if train_data.gender[i] == "female":
        if cnt_female <= num_samples:
            cnt_female +=1
            idx.append(i)
train_data_gender = train_data.loc[idx]
train_data_gender.reset_index(inplace=True)
train_data_gender.drop("index",axis=1,inplace=True)
```

```
for idx in trange(train_data_gender.shape[0]):
    file,gender_group = train_data_gender.filename[idx],train_data_gender.gender[idx]
    path = './gender_data'+"/"+gender_group
    if os.path.isdir(path):
        process_data(file,path)
    else :
        os.mkdir(path)
        process_data(file,path)
```

```
100%|██████████| 1002/1002 [01:34<00:00, 10.62it/s]
```

```python
import numpy as np
import pandas as pd
import os
import librosa

# Assume train_data is already loaded
voice_dir = 'cv-valid-train-extracted'  # Adjust the path if necessary

# Sample counter and limit
cnt_teens = 0
num_samples = 5  # Number of samples you want to process

# Process samples
for idx in range(len(train_data)):
    if cnt_teens < num_samples:
        filename = os.path.join(voice_dir, train_data.filename[idx])
        print(f"Selected filename: {filename}")

        # Load the audio file
        audio_array, sampling_rate = librosa.load(filename, sr=None)
        print(f"Loaded audio with shape: {audio_array.shape}, Sample Rate: {sampling_rate}")

        # Increment counter for samples processed
        cnt_teens += 1
    else:
        break  # Exit the loop after processing the required number of samples
```

```
Selected filename: cv-valid-train-extracted/cv-valid-train/sample-000005.mp3
Loaded audio with shape: (279936,), Sample Rate: 48000
Selected filename: cv-valid-train-extracted/cv-valid-train/sample-000008.mp3
Loaded audio with shape: (82944,), Sample Rate: 48000
Selected filename: cv-valid-train-extracted/cv-valid-train/sample-000013.mp3
Loaded audio with shape: (202752,), Sample Rate: 48000
Selected filename: cv-valid-train-extracted/cv-valid-train/sample-000014.mp3
Loaded audio with shape: (258048,), Sample Rate: 48000
Selected filename: cv-valid-train-extracted/cv-valid-train/sample-000019.mp3
Loaded audio with shape: (178560,), Sample Rate: 48000
```

```
num_samples = 500
idx = []
cnt_teens,cnt_twenties,cnt_thirties,cnt_fourties,cnt_fifties,cnt_sixties,cnt_seventies = 0,0,0,0,0,0,0
for i in range(train_data.shape[0]):
    if train_data.age[i] == "twenties":
        if cnt_twenties <= num_samples:
            cnt_twenties +=1
            idx.append(i)

    if train_data.age[i] == "thirties":
        if cnt_thirties <= num_samples:
            cnt_thirties +=1
            idx.append(i)
    if train_data.age[i] == "fourties":
        if cnt_fourties <= num_samples:
            cnt_fourties +=1
            idx.append(i)
    if train_data.age[i] == "fifties":
        if cnt_fifties <= num_samples:
            cnt_fifties +=1
            idx.append(i)
    if train_data.age[i] == "sixties":
        if cnt_sixties <= num_samples:
            cnt_sixties +=1
            idx.append(i)
    if train_data.age[i] == "seventies":
        if cnt_seventies <= num_samples:
            cnt_seventies +=1
            idx.append(i)
    if train_data.age[i] == "teens":
        if cnt_teens <= num_samples:
            cnt_teens +=1
            idx.append(i)
train_data_age = train_data.loc[idx]
train_data_age.reset_index(inplace=True)
train_data_age.drop("index",axis=1,inplace=True)
```

```
for idx in trange(train_data_age.shape[0]):
    file,age_group = train_data_age.filename[idx],train_data_age.age[idx]
    path = './age_data'+"/"+age_group
    if os.path.isdir(path):
        process_data(file,path)
    else :
        os.mkdir(path)
        process_data(file,path)
```

```
100%|| 3507/3507 [05:36<00:00, 10.41it/s]
```

Storing Accent data

```
num_samples = 200
idx = []
cnt_dict = {"us":0,"england":0,"indian":0,"australia":0,"canada":0,
 "scotland":0,"newzealand":0,"african":0,"ireland":0 }
for i in range(train_data.shape[0]):
    if cnt_dict[train_data.accent[i]] <= num_samples:
        cnt_dict[train_data.accent[i]] +=1
        idx.append(i)


train_data_accent = train_data.loc[idx]
train_data_accent.reset_index(inplace=True)
train_data_accent.drop("index",axis=1,inplace=True)
```

```
for idx in trange(train_data_accent.shape[0]):
    file,accent_group = train_data_accent.filename[idx],train_data_age.accent[idx]
    path = './accent_data'+"/"+accent_group
    if os.path.isdir(path):
        process_data(file,path)
    else :
        os.mkdir(path)
        process_data(file,path)
```

```
100%|██████████| 1809/1809 [02:50<00:00, 10.59it/s]
```

Defining Models

```
def preprocessFn(img):
    return (img-127.0)/127.0
```

```
depth = 3
INIT_LR = 0.001
EPOCHS = 10
inputShape=(128,128,3)

data_gen = ImageDataGenerator(validation_split=0.2,preprocessing_function=preprocessFn)
```

```python
def recall_score(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_score(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))

def spec_sens(confusion_matrix):
    FP = confusion_matrix.sum(axis=0) - np.diag(confusion_matrix)
    FN = confusion_matrix.sum(axis=1) - np.diag(confusion_matrix)
    TP = np.diag(confusion_matrix)
    TN = confusion_matrix.sum() - (FP + FN + TP)

    # Sensitivity, hit rate, recall, or true positive rate
    sensitivity = TP/(TP+FN)
    # Specificity or true negative rate
    specificity = TN/(TN+FP)

    return sensitivity,specificity
```

Dense Architecture

```python
def dense_model(base_model,num_classes):
    model = Sequential()
    model.add(base_model)
    model.add(Flatten())
    model.add(BatchNormalization())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(num_classes, activation='softmax'))

    return model
```

```python
def define_models(classes):
    num_class = classes
    #Xception
    model_xcep = Xception(include_top=False, weights="imagenet",input_shape =inputShape)
    model_xception = dense_model(model_xcep,num_class)
    model_xception.layers[0].trainable = False

    # InceptionV3
    model_incep = InceptionV3(include_top=False, weights="imagenet",input_shape =inputShape)
    model_inception = dense_model(model_incep,num_class)
    model_inception.layers[0].trainable = False

    #VGG16
    model_1 = VGG16(include_top=False, weights="imagenet",input_shape =inputShape)
    model_vgg1 = dense_model(model_1,num_class)
    model_vgg1.layers[0].trainable = False

    #VGG 19
    model_2 = VGG19(include_top=False, weights="imagenet",input_shape =inputShape)
    model_vgg2 = dense_model(model_2,num_class)
    model_vgg2.layers[0].trainable = False

    #ResNet50
    model_res = ResNet50(include_top=False, weights="imagenet",input_shape =inputShape)
    model_resnet = dense_model(model_res,num_class)
    model_resnet.layers[0].trainable = False

    return [model_xception,model_inception,model_vgg1,model_vgg2,model_resnet]
```

Ensemble

```
def stacking_predictions(models,data):
    # array to store values
    stackValues = None
    for model in models:
        # making predictions for each model
        y_pred = model.predict(data)
        # stack predictions into [rows, members, probabilities]
        if stackValues is None:
            stackValues = y_pred
        else:
            stackValues = np.dstack((stackValues,y_pred))
    # flatten predictions to [rows, members x probabilities]
    stackValues = stackValues.reshape((stackValues.shape[0], stackValues.shape[1]*stackValues.shape[2]))
    return stackValues
```

```
def fit_models(models,data):
    # stacked data with ensemble
    stackedValues = stacking_predictions(models,data)
    log_reg = LogisticRegression()
    log_reg.fit(stackedValues,data.labels)
    return log_reg
```

```
def stacked_prediction(members, model, inputX):
    # create dataset using ensemble
    stackedX = stacking_predictions(members, inputX)
    # make a prediction
    yhat = model.predict(stackedX)
    return yhat
```

GENDER PREDICTION

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report
from tensorflow.keras.applications import InceptionV3, VGG16, ResNet50
from tensorflow.keras import models, layers
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set your constants
INIT_LR = 0.001
num_classes = 2  # Adjust based on your gender classes (e.g., male/female)
inputShape = (224, 224, 3)  # VGG16 and ResNet require 224x224 input size
```

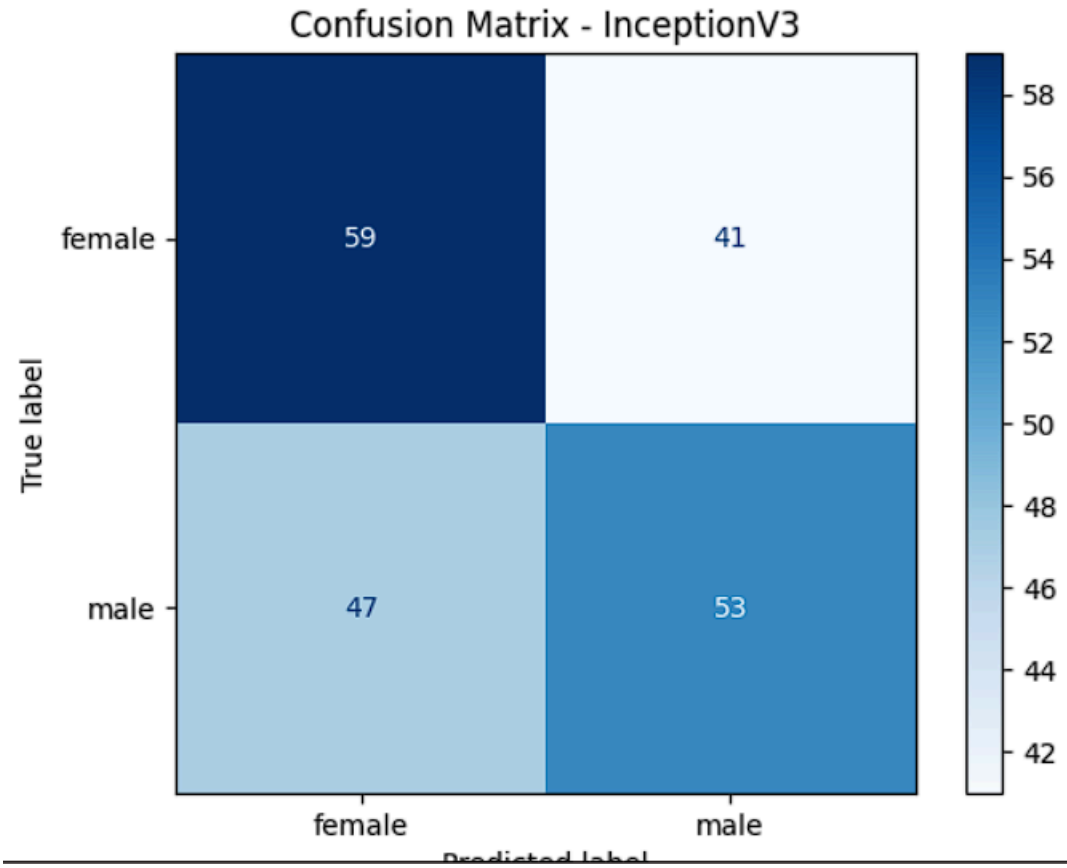
```
# Initialize ImageDataGenerator with validation split for gender data
data_gen_gender = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2  # Split data into training and validation sets
)

# Create training and validation data generators for gender data
train_data_gender = data_gen_gender.flow_from_directory(
    directory=r"./gender_data",
    target_size=inputShape[0:2],
    batch_size=4,
    class_mode='categorical',
    shuffle=True,
    subset='training'
)

val_data_gender = data_gen_gender.flow_from_directory(
    directory=r"./gender_data",
    target_size=inputShape[0:2],
    batch_size=1,
    class_mode='categorical',
    shuffle=True,
    subset='validation'
)

# Function to create models
def create_model(model_type):
    if model_type == "inception":
        base_model = InceptionV3(weights='imagenet', include_top=False,
input_shape=inputShape)
    elif model_type == "vgg16":
        base_model = VGG16(weights='imagenet', include_top=False,
input_shape=inputShape)
    elif model_type == "resnet":
        base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=inputShape)
    else:
        raise ValueError("Model type not recognized.")

    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
```

```
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])

    return model

# Function to compile and train the model
def compile_and_train_model(model, train_data, val_data):
    opt = Adam(learning_rate=INIT_LR)
    model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])

    print(f"[INFO] training {model.name}...")
    history = model.fit(train_data, validation_data=val_data, epochs=10)
# Adjust epochs as needed
    return history

# Train InceptionV3
model_inception = create_model("inception")
history_inception = compile_and_train_model(model_inception,
train_data_gender, val_data_gender)

# Train VGG16
model_vgg16 = create_model("vgg16")
history_vgg16 = compile_and_train_model(model_vgg16, train_data_gender,
val_data_gender)

    plt.tight_layout()
    plt.show()

# Example for visualizing predictions with InceptionV3
val_images, val_labels = next(iter(val_data_gender))
class_labels = list(train_data_gender.class_indices.keys())
predict_and_display(val_images, val_labels, model_inception, class_labels)

# Confusion Matrix and Classification Report for InceptionV3
val_predictions = model_inception.predict(val_data_gender)
predicted_classes = np.argmax(val_predictions, axis=1)
true_classes = val_data_gender.classes
cm = confusion_matrix(true_classes, predicted_classes)
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=class_labels)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix - InceptionV3")
plt.show()

report = classification_report(true_classes, predicted_classes,
target_names=class_labels)
print("Classification Report - InceptionV3:\n", report)

# Now repeat the evaluation for VGG16
val_predictions_vgg16 = model_vgg16.predict(val_data_gender)
predicted_classes_vgg16 = np.argmax(val_predictions_vgg16, axis=1)

# Confusion Matrix and Classification Report for VGG16
cm_vgg16 = confusion_matrix(true_classes, predicted_classes_vgg16)
disp_vgg16 = ConfusionMatrixDisplay(confusion_matrix=cm_vgg16,
display_labels=class_labels)
disp_vgg16.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix - VGG16")
plt.show()

report_vgg16 = classification_report(true_classes,
predicted_classes_vgg16, target_names=class_labels)
print("Classification Report - VGG16:\n", report_vgg16)

# Repeat the evaluation for ResNet50
val_predictions_resnet = model_resnet.predict(val_data_gender)
predicted_classes_resnet = np.argmax(val_predictions_resnet, axis=1)

# Confusion Matrix and Classification Report for ResNet50
cm_resnet = confusion_matrix(true_classes, predicted_classes_resnet)
disp_resnet = ConfusionMatrixDisplay(confusion_matrix=cm_resnet,
display_labels=class_labels)
disp_resnet.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix - ResNet50")
plt.show()

report_resnet = classification_report(true_classes,
predicted_classes_resnet, target_names=class_labels)
print("Classification Report - ResNet50:\n", report_resnet)
```

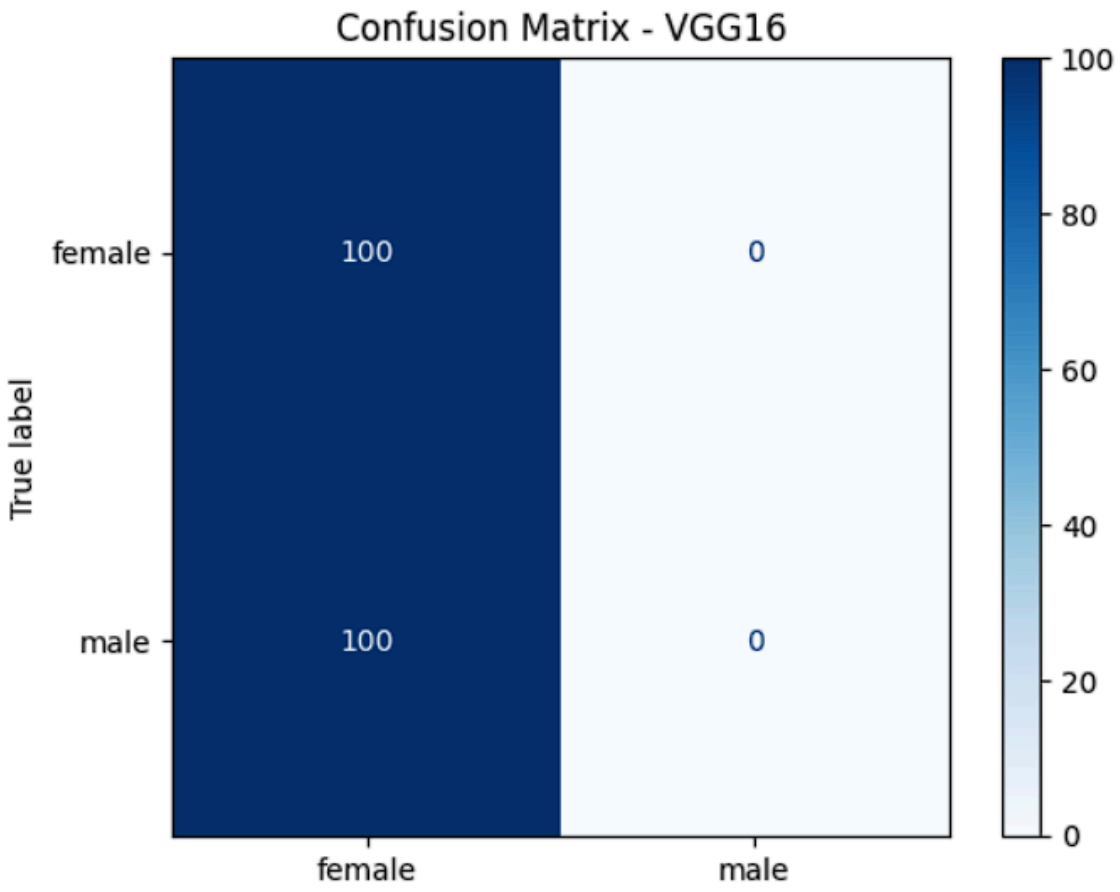
```python
# Saving models
model_inception.save("inception_gender_model.h5")
model_vgg16.save("vgg16_gender_model.h5")
model_resnet.save("resnet_gender_model.h5")

# Loading and evaluating the models (Example for ResNet)
loaded_model_resnet = tf.keras.models.load_model("resnet_gender_model.h5")
loss, accuracy = loaded_model_resnet.evaluate(val_data_gender)
print(f"Loaded ResNet model accuracy: {accuracy:.2f}")
```

```
Found 802 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
[INFO] training sequential_3...
Epoch 1/10
201/201 [==============================] - 37s 85ms/step - loss: 0.8766 - accuracy: 0.5162 - val_loss: 0.6860 - val_accuracy: 0.6100
Epoch 2/10
201/201 [==============================] - 12s 57ms/step - loss: 0.6341 - accuracy: 0.6347 - val_loss: 0.6694 - val_accuracy: 0.7150
Epoch 3/10
201/201 [==============================] - 11s 57ms/step - loss: 0.5886 - accuracy: 0.7070 - val_loss: 7.0608 - val_accuracy: 0.5550
Epoch 4/10
201/201 [==============================] - 11s 57ms/step - loss: 0.4896 - accuracy: 0.7805 - val_loss: 1.4632 - val_accuracy: 0.8400
Epoch 5/10
201/201 [==============================] - 11s 57ms/step - loss: 0.5511 - accuracy: 0.7731 - val_loss: 4.7557 - val_accuracy: 0.7150
Epoch 6/10
201/201 [==============================] - 12s 57ms/step - loss: 0.4451 - accuracy: 0.8367 - val_loss: 0.5678 - val_accuracy: 0.7550
Epoch 7/10
201/201 [==============================] - 11s 57ms/step - loss: 0.4652 - accuracy: 0.8292 - val_loss: 0.4648 - val_accuracy: 0.8400
Epoch 8/10
201/201 [==============================] - 12s 59ms/step - loss: 0.4367 - accuracy: 0.8130 - val_loss: 0.6991 - val_accuracy: 0.8250
Epoch 9/10
201/201 [==============================] - 12s 57ms/step - loss: 0.4781 - accuracy: 0.7893 - val_loss: 2.0355 - val_accuracy: 0.5050
Epoch 10/10
201/201 [==============================] - 11s 57ms/step - loss: 0.4295 - accuracy: 0.8416 - val_loss: 0.4080 - val_accuracy: 0.8900
```

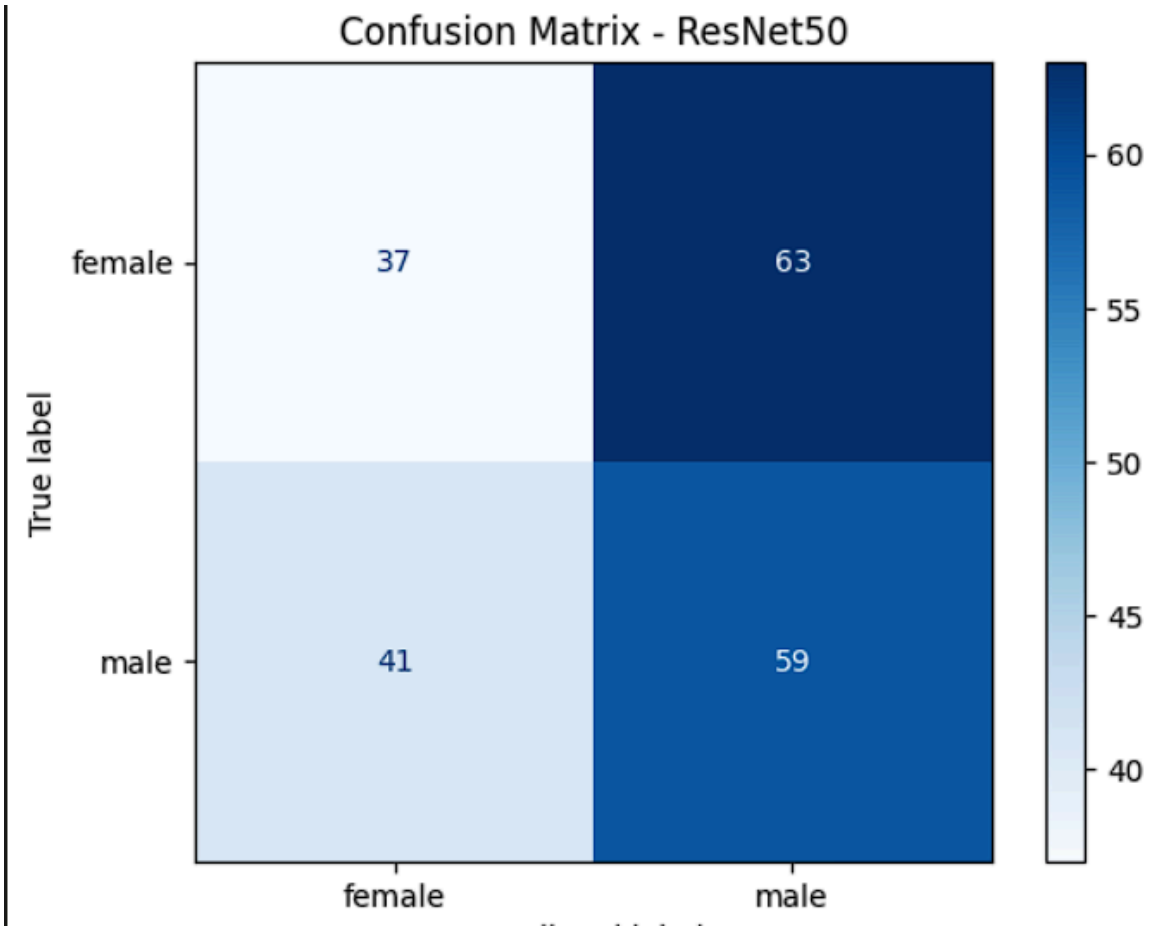


Classification Report - InceptionV3:

	precision	recall	f1-score	support
female	0.56	0.59	0.57	100
male	0.56	0.53	0.55	100
accuracy			0.56	200
macro avg	0.56	0.56	0.56	200
weighted avg	0.56	0.56	0.56	200



	precision	recall	f1-score	support
female	0.50	1.00	0.67	100
male	0.00	0.00	0.00	100
accuracy			0.50	200
macro avg	0.25	0.50	0.33	200
weighted avg	0.25	0.50	0.33	200



	precision	recall	f1-score	support
female	0.47	0.37	0.42	100
male	0.48	0.59	0.53	100
accuracy			0.48	200
macro avg	0.48	0.48	0.47	200
weighted avg	0.48	0.48	0.47	200

```
Loaded ResNet model accuracy: 0.85
```

AGE PREDICTION

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report
from tensorflow.keras.applications import InceptionV3, ResNet50, VGG16
from tensorflow.keras import models, layers
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set your constants
INIT_LR = 0.001
num_classes = 7
inputShape = (299, 299, 3)

# Initialize ImageDataGenerator
data_gen = ImageDataGenerator(rescale=1.0/255, validation_split=0.2)

# Create data generators
train_data = data_gen.flow_from_directory(
    directory=r"./age_data",
    target_size=inputShape[0:2],
    batch_size=4,
    class_mode='categorical',
    shuffle=True,
    subset='training'
)

val_data = data_gen.flow_from_directory(
    directory=r"./age_data",
    target_size=inputShape[0:2],
    batch_size=1,
    class_mode='categorical',
    shuffle=True,
    subset='validation'
)

# Function to define and compile a model
def create_model(base_model_class):
    base_model = base_model_class(weights='imagenet', include_top=False,
input_shape=inputShape)
    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(256, activation='relu'),
```

```
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    model.compile(loss="categorical_crossentropy",
optimizer=Adam(learning_rate=INIT_LR), metrics=["accuracy"])
    return model

# Train the InceptionV3 model
model_inception = create_model(InceptionV3)
print("[INFO] training InceptionV3 network...")
history_inception = model_inception.fit(train_data,
validation_data=val_data)

# Train the ResNet model
model_resnet = create_model(ResNet50)
print("[INFO] training ResNet network...")
history_resnet = model_resnet.fit(train_data, validation_data=val_data)

# Train the VGG16 model
model_vgg = create_model(VGG16)
print("[INFO] training VGG16 network...")
history_vgg = model_vgg.fit(train_data, validation_data=val_data)

# Prediction method: Visualize predictions
def predict_and_display(images, true_labels, model, class_labels):
    predictions = model.predict(images)
    predicted_classes = np.argmax(predictions, axis=1)

    plt.figure(figsize=(10, 10))
    for i in range(len(images)):
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])
        plt.title(f"True: {class_labels[np.argmax(true_labels[i])]} |
Pred: {class_labels[predicted_classes[i]]}")
        plt.axis('off')
    plt.tight_layout()
    plt.show()

val_images, val_labels = next(iter(val_data))
class_labels = list(train_data.class_indices.keys())

# Prediction methods for InceptionV3
```

```
print("[INFO] InceptionV3 Predictions...")
predict_and_display(val_images, val_labels, model_inception, class_labels)
val_predictions = model_inception.predict(val_data)
predicted_classes_inception = np.argmax(val_predictions, axis=1)
true_classes = val_data.classes
cm_inception = confusion_matrix(true_classes, predicted_classes_inception)
ConfusionMatrixDisplay(confusion_matrix=cm_inception,
display_labels=class_labels).plot(cmap=plt.cm.Blues)
plt.show()
print(classification_report(true_classes, predicted_classes_inception,
target_names=class_labels))

# Save the InceptionV3 model
model_inception.save("inception_model.h5")

# Load and evaluate the InceptionV3 model
loaded_model_inception = tf.keras.models.load_model("inception_model.h5")
loss_inception, accuracy_inception =
loaded_model_inception.evaluate(val_data)
print(f"Loaded InceptionV3 model accuracy: {accuracy_inception:.2f}")

# Prediction methods for ResNet
print("[INFO] ResNet Predictions...")
predict_and_display(val_images, val_labels, model_resnet, class_labels)
val_predictions_resnet = model_resnet.predict(val_data)
predicted_classes_resnet = np.argmax(val_predictions_resnet, axis=1)
cm_resnet = confusion_matrix(true_classes, predicted_classes_resnet)
ConfusionMatrixDisplay(confusion_matrix=cm_resnet,
display_labels=class_labels).plot(cmap=plt.cm.Blues)
plt.show()
print(classification_report(true_classes, predicted_classes_resnet,
target_names=class_labels))

# Save the ResNet model
model_resnet.save("resnet_model.h5")

# Load and evaluate the ResNet model
loaded_model_resnet = tf.keras.models.load_model("resnet_model.h5")
loss_resnet, accuracy_resnet = loaded_model_resnet.evaluate(val_data)
print(f"Loaded ResNet model accuracy: {accuracy_resnet:.2f}")

# Prediction methods for VGG16
```

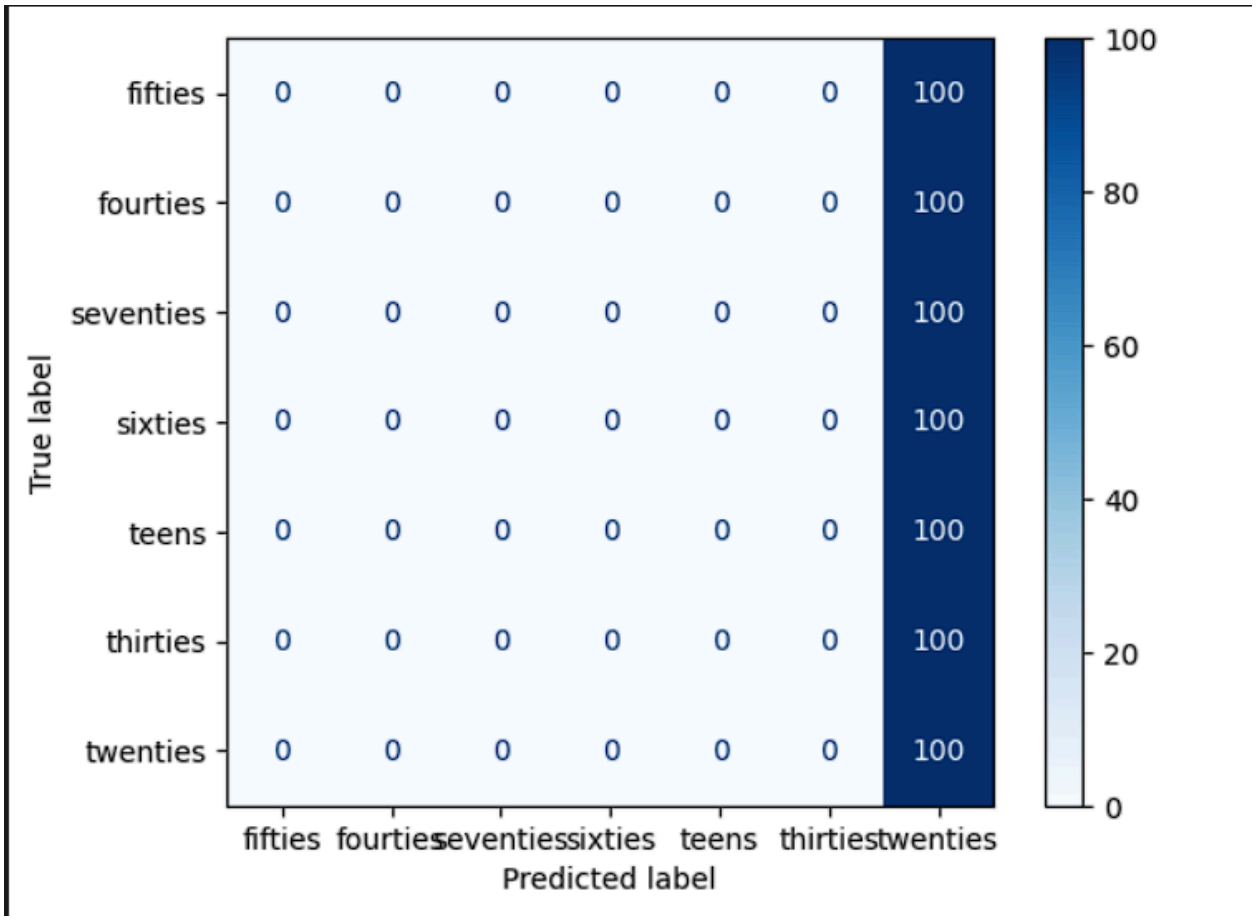
```
print("[INFO] VGG16 Predictions...")
predict_and_display(val_images, val_labels, model_vgg, class_labels)
val_predictions_vgg = model_vgg.predict(val_data)
predicted_classes_vgg = np.argmax(val_predictions_vgg, axis=1)
cm_vgg = confusion_matrix(true_classes, predicted_classes_vgg)
ConfusionMatrixDisplay(confusion_matrix=cm_vgg,
display_labels=class_labels).plot(cmap=plt.cm.Blues)
plt.show()
print(classification_report(true_classes, predicted_classes_vgg,
target_names=class_labels))

# Save the VGG16 model
model_vgg.save("vgg_model.h5")

# Load and evaluate the VGG16 model
loaded_model_vgg = tf.keras.models.load_model("vgg_model.h5")
loss_vgg, accuracy_vgg = loaded_model_vgg.evaluate(val_data)
print(f"Loaded VGG16 model accuracy: {accuracy_vgg:.2f}")
```

	precision	recall	f1-score	support
fifties	0.00	0.00	0.00	100
fourties	0.14	0.11	0.12	100
seventies	0.29	0.02	0.04	100
sixties	0.08	0.05	0.06	100
teens	0.14	0.80	0.25	100
thirties	0.00	0.00	0.00	100
twenties	0.00	0.00	0.00	100
accuracy			0.14	700
macro avg	0.09	0.14	0.07	700
weighted avg	0.09	0.14	0.07	700

```
700/700 [==============================] - 9s 12ms/step - loss: 2.5303 - accuracy: 0.1457
Loaded InceptionV3 model accuracy: 0.15
```

	precision	recall	f1-score	support
fifties	0.00	0.00	0.00	100
fourties	0.00	0.00	0.00	100
seventies	0.00	0.00	0.00	100
sixties	0.00	0.00	0.00	100
teens	0.00	0.00	0.00	100
thirties	0.00	0.00	0.00	100
twenties	0.14	1.00	0.25	100
accuracy			0.14	700
macro avg	0.02	0.14	0.04	700
weighted avg	0.02	0.14	0.04	700

```
Loaded ResNet model accuracy: 0.14
```


fifties
fourties
seventies
sixties
teens
thirties
twenties
True label
0 0 0 0 0 0 100
fifties
fourties
seventies
sixties
teens
thirties
twenties
Predicted label
100
80
60
40
20
0

	precision	recall	f1-score	support
fifties	0.00	0.00	0.00	100
fourties	0.00	0.00	0.00	100
seventies	0.00	0.00	0.00	100
sixties	0.00	0.00	0.00	100
teens	0.00	0.00	0.00	100
thirties	0.00	0.00	0.00	100
twenties	0.14	1.00	0.25	100
accuracy			0.14	700
macro avg	0.02	0.14	0.04	700
weighted avg	0.02	0.14	0.04	700

```
Loaded VGG16 model accuracy: 0.14
```

DEPLOYMENT

CODE:

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier  # Import Decision Tree
model
from sklearn.metrics import (accuracy_score, recall_score,
                             precision_score, f1_score, confusion_matrix)
import librosa  # for audio analysis
import librosa.display
import joblib  # for loading the model

# Streamlit app title
st.title("Data and Audio Analysis with Streamlit")

# Load pre-trained gender model
try:
    resnet_gender_model = joblib.load('gender_model.pkl')
except Exception as e:
    st.error(f"Error loading gender model: {e}")

# Load pre-trained age model
try:
```

```
    age_model = joblib.load('age_model.pkl')
except Exception as e:
    st.error(f"Error loading age model: {e}")

# Function to extract audio features for predictions
def extract_audio_features(y, sr):
    # Extract features like MFCCs, Chroma, Mel Spectrogram, etc.
    mfccs = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13).T, axis=0)
    chroma = np.mean(librosa.feature.chroma_stft(y=y, sr=sr).T, axis=0)
    mel = np.mean(librosa.feature.melspectrogram(y=y, sr=sr).T, axis=0)
    return np.hstack((mfccs, chroma, mel))

# File upload for CSV or MP3
uploaded_file = st.file_uploader("Upload your CSV or MP3 file",
type=["csv", "mp3"])

# Process CSV file if uploaded
if uploaded_file and uploaded_file.name.endswith('.csv'):
    try:
        data = pd.read_csv(uploaded_file)
        st.write("Data Preview:")
        st.write(data.head())

        # Summary Statistics
        if st.checkbox("Show Summary Statistics"):
            st.write(data.describe())

        # Data Visualization
        if st.checkbox("Show Data Distribution"):
            column = st.selectbox("Choose column for distribution plot:",
data.columns)
            fig, ax = plt.subplots()
            sns.histplot(data[column], kde=True, ax=ax)
            st.pyplot(fig)
```

```
        # Model Training
        if st.checkbox("Train a Decision Tree Model"):
            target = st.selectbox("Select Target Variable", data.columns)
            features = st.multiselect("Select Feature Variables", [col for
col in data.columns if col != target])

            if st.button("Train Model"):
                if len(features) == 0:
                    st.error("Please select at least one feature
variable.")
                else:
                    X = data[features]
                    y = data[target]
                    model = DecisionTreeClassifier()  # Instantiate
Decision Tree model
                    model.fit(X, y)
                    predictions = model.predict(X)

                    # Display Metrics
                    st.write("Accuracy:", accuracy_score(y, predictions))
                    st.write("Recall:", recall_score(y, predictions,
average='macro'))
                    st.write("Precision:", precision_score(y, predictions,
average='macro'))
                    st.write("F1 Score:", f1_score(y, predictions,
average='macro'))

                    # Confusion Matrix
                    cm = confusion_matrix(y, predictions)
                    fig, ax = plt.subplots()
                    sns.heatmap(cm, annot=True, fmt="d", ax=ax)
                    ax.set(title="Confusion Matrix")
                    st.pyplot(fig)
```

```
    except Exception as e:
        st.error(f"Error processing CSV: {e}")

# Process MP3 file if uploaded
elif uploaded_file and uploaded_file.name.endswith('.mp3'):
    # Play audio in Streamlit
    st.audio(uploaded_file, format='audio/mp3')

    # Load and analyze audio
    if st.checkbox("Show Audio Waveform"):
        try:
            # Load audio file
            y, sr = librosa.load(uploaded_file, sr=None)  # sr=None
preserves original sampling rate

            # Plot waveform
            fig, ax = plt.subplots()
            librosa.display.waveshow(y, sr=sr, ax=ax)
            ax.set(title="Audio Waveform")
            st.pyplot(fig)

            # Extract and display additional audio features if desired
            if st.checkbox("Show Audio Features"):
                st.write("Audio Sample Rate:", sr)
                st.write("Audio Duration:", librosa.get_duration(y=y,
sr=sr), "seconds")

                # Display Spectrogram
                st.write("Spectrogram:")
                D = librosa.amplitude_to_db(np.abs(librosa.stft(y)),
ref=np.max)
                fig, ax = plt.subplots()
                img = librosa.display.specshow(D, sr=sr, x_axis='time',
y_axis='log', ax=ax)
```

```
                fig.colorbar(img, ax=ax, format="%+2.0f dB")
                ax.set(title="Spectrogram")
                st.pyplot(fig)

            # Gender Prediction
            if st.checkbox("Predict Gender"):
                audio_features = extract_audio_features(y, sr)
                gender_prediction =
resnet_gender_model.predict(audio_features.reshape(1, -1))
                st.write(f"Predicted Gender: {gender_prediction[0]}")

            # Age Prediction
            if st.checkbox("Predict Age"):
                audio_features = extract_audio_features(y, sr)
                age_prediction =
age_model.predict(audio_features.reshape(1, -1))
                st.write(f"Predicted Age: {age_prediction[0]}")

        except Exception as e:
            st.error(f"Error processing audio: {e}")
```

Output:

1)The app displays the title "Data and Audio Analysis with Streamlit.

2)It attempts to load two pre-trained models:

gender_model.pkl for gender prediction.

age_model.pkl for age prediction.

3)If either model fails to load, an error message is displayed.

4)Users can upload either a CSV or MP3 file through a file uploader widget.

5)If an MP3 file is uploaded:

Audio Playback: The audio can be played within the app.

Audio Waveform: If checked, it displays the waveform of the audio file.

Audio Features: If checked, it shows:

Sample rate of the audio.

Duration of the audio.

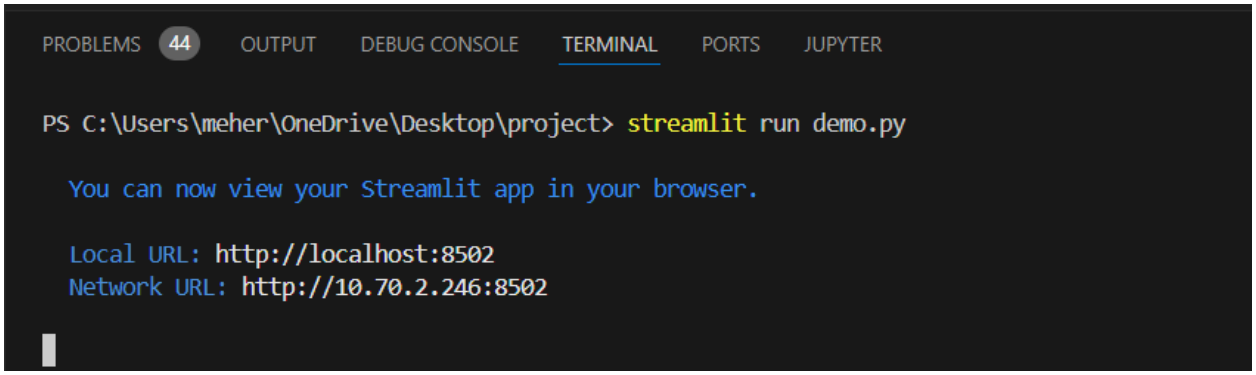
A spectrogram of the audio, visualizing its frequency content over time.

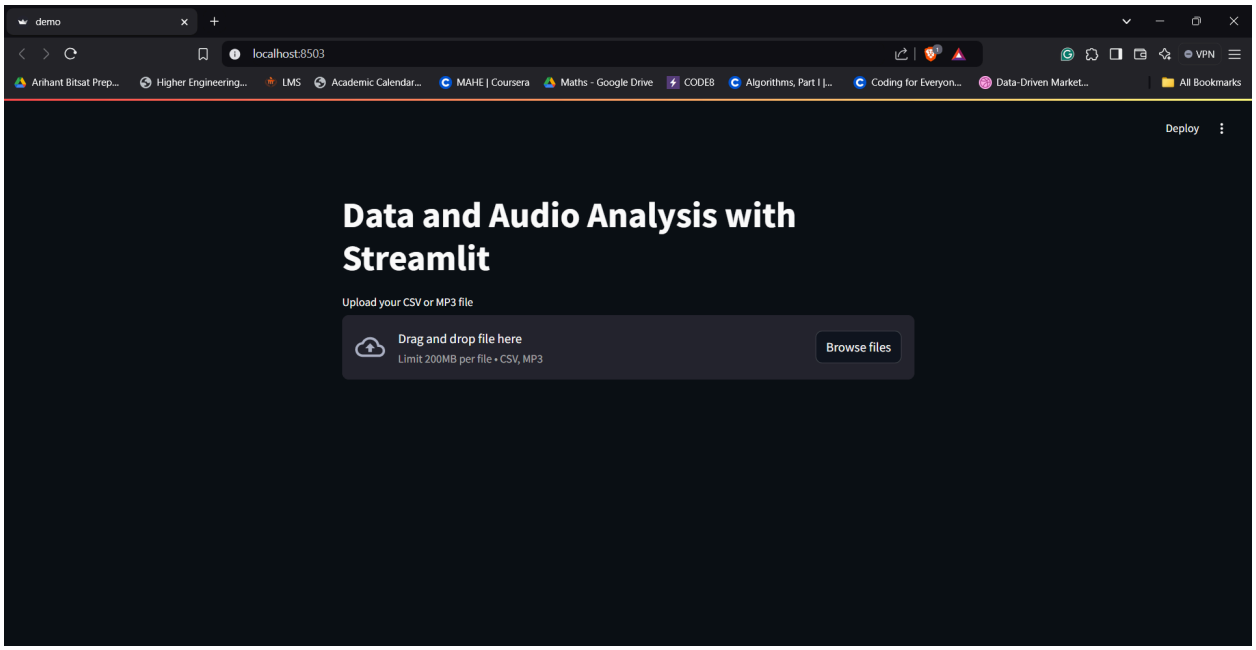
6). Predictions

Predict Gender: If checked, the app extracts audio features and uses the gender model to predict gender, displaying the predicted value.

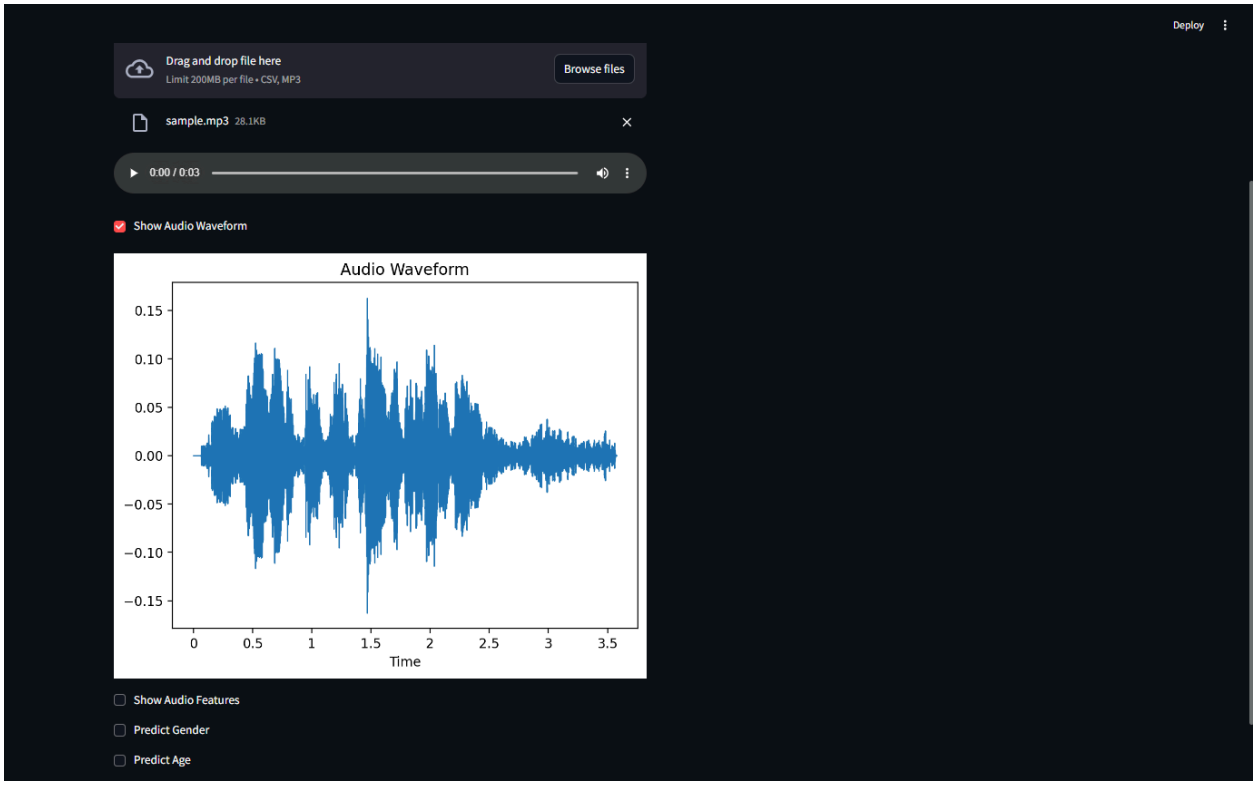
Predict Age: If checked, it uses the age model to predict age, displaying the predicted value.

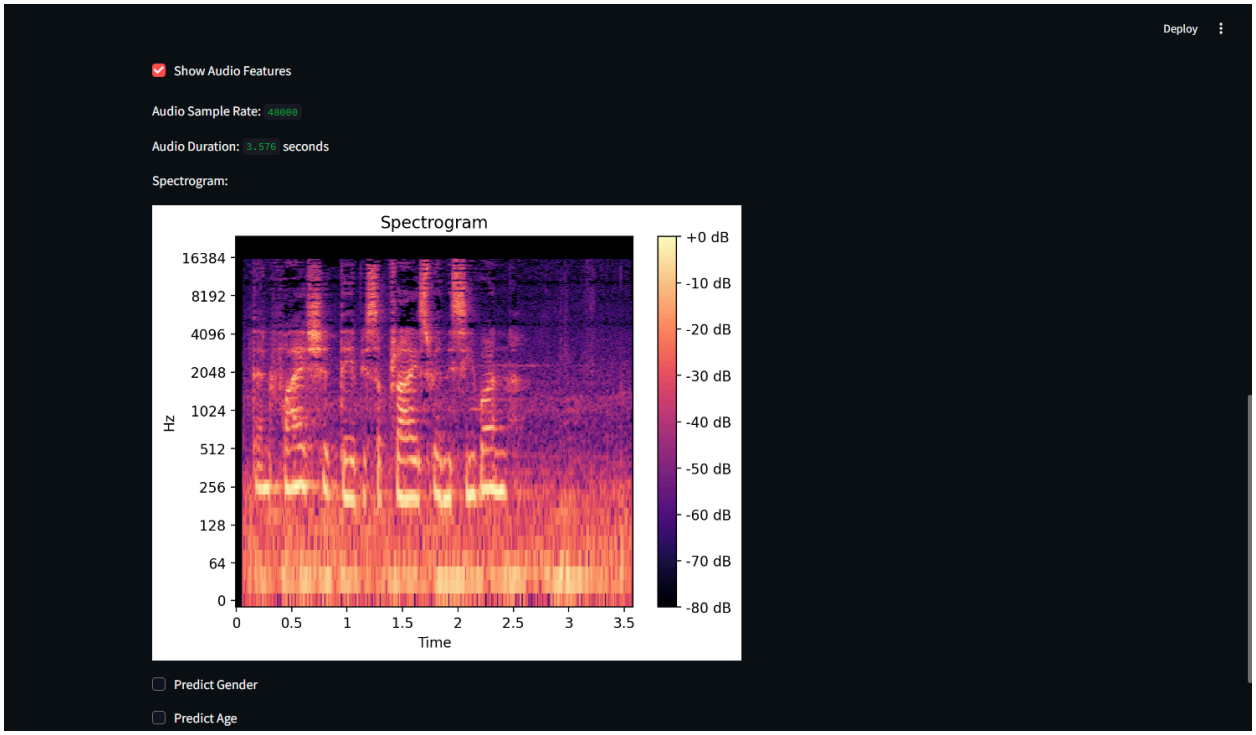
7)The application includes error handling to notify users if there are issues with loading models or processing files.





Output:


Deploy
Drag and drop file here
Limit 200MB per file • CSV, MP3
Browse files
sample.mp3 28.1KB
0:00 / 0:03
Show Audio Waveform
Audio Waveform
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
0.5
1
1.5
2
2.5
3
3.5
Time
Show Audio Features
Predict Gender
Predict Age


Deploy
Show Audio Features
Audio Sample Rate: 48000
Audio Duration: 3.576 seconds
Spectrogram:
Spectrogram
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
0.5
1
1.5
2
2.5
3
3.5
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB
Predict Gender
Predict Age