```
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt
from matplotlib import image as img

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer

import glob
import seaborn as sns
import os
import random
from PIL import Image
import sys
from tqdm import tqdm
from tensorflow.keras.utils import to_categorical
import tensorflow as tf
from keras import layers
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Activation, Dropout,
Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
def plot_count(feature, title, df, size=1, show_all=False):
    f, ax = plt.subplots(1,1, figsize=(4*size,4))
    total = float(len(df))
    if show_all:
        g = sns.countplot(df[feature], palette='Set3')
        g.set_title("{} distribution".format(title))
    else:
        g = sns.countplot(df[feature], order =
df[feature].value_counts().index[:20], palette='Set3')
        if(size > 2):
            plt.xticks(rotation=90, size=8)
            for p in ax.patches:
                height = p.get_height()
                ax.text(p.get_x()+p.get_width()/2.,
                        height + 0.2,
                        '{:1.2f}%'.format(100*height/total),
                        ha="center")
        g.set_title("Number and percentage of {}".format(title))
    plt.show()
def check_disease(df,start,end):
    df = df.iloc[:, start:end]
    disease_name, zeroCount, oneCount = [], [], []
    rowLen = len(df)
    for (column_name, column) in df.iteritems():
        disease_name.append(column_name)
        zeroCount.append(df[column_name].value_counts()[0])
    oneCount = [rowLen - x for x in zeroCount]

    return disease_name, zeroCount, oneCount
def has_cataract(text):
    if "cataract" in text:
        return 1
    else:
        return 0
from tensorflow.keras.preprocessing.image import load_img,img_to_array
dataset_dir = "/content/archive.zip"
image_size=224
labels = []
dataset = []
def create_dataset(image_category,label):
    for img in tqdm(image_category):
        image_path = os.path.join(dataset_dir,img)
        try:
            image = cv2.imread(image_path,cv2.IMREAD_COLOR)
            image = cv2.resize(image,(image_size,image_size))
            dataset.append([np.array(image),np.array(label)])
        except:
            continue

    random.shuffle(dataset)
    return dataset
from tensorflow.keras.applications.vgg19 import VGG19
vgg = VGG19(weights="imagenet",include_top =
False,input_shape=(image_size,image_size,3))

for layer in vgg.layers:
    layer.trainable = False

from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten,Dense
model = Sequential()
model.add(vgg)
model.add(Flatten())
model.add(Dense(1,activation="sigmoid"))

model.summary()
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
checkpoint =
ModelCheckpoint("vgg19.h5",monitor="val_acc",verbose=1,save_best_only=T
rue,
                             save_weights_only=False,period=1)
earlystop = EarlyStopping(monitor="val_acc",patience=5,verbose=1)
```

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import cv2
import random
import pickle
from tqdm import tqdm
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
for dirname, _, filenames in os.walk('/content/archive.zip'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
def has_cataract(text):
    if "cataract" in text:
        return 1
    else:
        return 0
def create_dataset(image_category,label):
    for img in tqdm(image_category):
        image_path = os.path.join(dataset_dir,img)
        try:
            image = cv2.imread(image_path,cv2.IMREAD_COLOR)
            image = cv2.resize(image,(image_size,image_size))

        except:
            continue

        dataset.append([np.array(image),np.array(label)])
    random.shuffle(dataset)
    return dataset
import os
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt, image as mpimg
from tqdm import tqdm
from time import time
from collections import Counter
import random

import tensorflow as tf
from tensorflow.keras import models, layers, optimizers, losses,
metrics, utils, callbacks, applications
from sklearn.model_selection import train_test_split as tts
import cv2 as cv
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import cv2
import random
```

```
from tqdm import tqdm
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
from itertools import cycle
from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from sklearn.metrics import roc_auc_score
import os
```