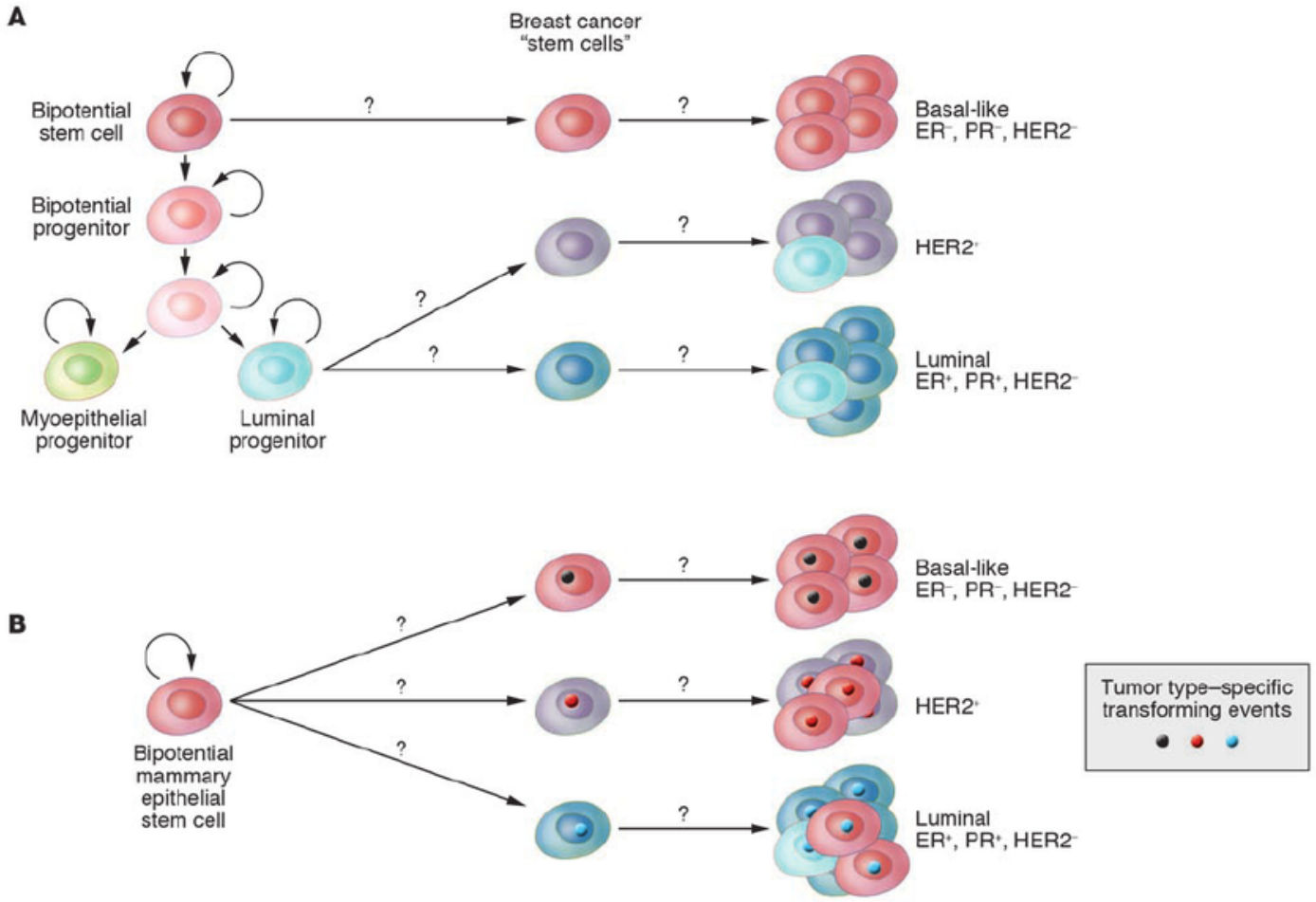



```
from google.colab.output import eval_js

import time
start_time = time.time()

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tqdm.notebook import tqdm

import keras
from keras import backend as K
from tensorflow.keras.layers import *
from keras.models import Sequential
from keras.layers import Dense, Conv2D
from keras.layers import Activation, MaxPooling2D, Dropout, Flatten, Reshape
from keras.wrappers.scikit_learn import KerasClassifier

import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import os
import random
from PIL import Image
```

```
import gdown

import argparse
import numpy as np
from keras.layers import Conv2D, Input, BatchNormalization, LeakyReLU, ZeroPadding2D, UpSampl
from keras.layers.merge import add, concatenate
from keras.models import Model
import struct
from google.colab.patches import cv2_imshow
from copy import deepcopy
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import roc_auc_score
from sklearn.base import BaseEstimator

from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

from sklearn.metrics import make_scorer
from sklearn.metrics import accuracy_score
from keras.applications.mobilenet import MobileNet

!pip install hypopt
from hypopt import GridSearch

from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering

!pip install -U opencv-contrib-python
import cv2

!pip install tensorflowjs
import tensorflowjs as tfjs

from google.colab import files
```

```
import requests, io, zipfile

# Prepare data

images_1 = os.makedirs('images_1', exist_ok=True)
images_2= os.makedirs('images_2', exist_ok=True)
images_all= os.makedirs('images_all', exist_ok=True)

metadata_path = 'metadata.csv'
image_path_1 = 'images_1.zip'
image_path_2 = 'images_2.zip'
images_rgb_path = 'hmnist_8_8_RGB.csv'

!wget -O metadata.csv 'metadata.csv'
!wget -O images_1.zip 'images_1.zip'
!wget -O images_2.zip 'images_2.zip'
!wget -O hmnist_8_8_RGB.csv 'hmnist_8_8_RGB.csv'

!unzip -q -o images_1.zip -d images_1
!unzip -q -o images_2.zip -d images_2

!pip install patool
import patoolib

import os.path
from os import path

from distutils.dir_util import copy_tree

fromDirectory = 'images_1'
toDirectory = 'images_all'

copy_tree(fromDirectory, toDirectory)

fromDirectory = 'images_2'
toDirectory = 'images_all'

copy_tree(fromDirectory, toDirectory)

print("Downloaded Data")
```

```
Collecting hypopt
  Downloading hypopt-1.0.9-py2.py3-none-any.whl (13 kB)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist-packages (
Installing collected packages: hypopt
Successfully installed hypopt-1.0.9
Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.7/dist-pa
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-4.5.3.56-cp37-cp37m-manylinux2014_x86_64.whl (56.1 M
     |████████████████████████████████| 56.1 MB 32 kB/s
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (
Installing collected packages: opencv-contrib-python
  Attempting uninstall: opencv-contrib-python
    Found existing installation: opencv-contrib-python 4.1.2.30
    Uninstalling opencv-contrib-python-4.1.2.30:
      Successfully uninstalled opencv-contrib-python-4.1.2.30
Successfully installed opencv-contrib-python-4.5.3.56
Collecting tensorflowjs
  Downloading tensorflowjs-3.8.0-py3-none-any.whl (64 kB)
     |████████████████████████████████| 64 kB 2.6 MB/s
Requirement already satisfied: tensorflow-hub<0.13,>=0.7.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: six<2,>=1.12.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tensorflow<3,>=2.1.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: numpy~=1.19.2 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: gast==0.4.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: tensorflow-estimator<2.6.0,>=2.5.0rc0 in /usr/local/lib/p
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: absl-py~=0.10 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: tensorboard~=2.5 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: grpcio~=1.34.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: keras-nightly~=2.5.0.dev in /usr/local/lib/python3.7/dist
Requirement already satisfied: typing-extensions~=3.7.4 in /usr/local/lib/python3.7/dist
Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.7/di
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: h5py~=3.1.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/p
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packa
```

```
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Installing collected packages: tensorflowjs
Successfully installed tensorflowjs-3.8.0
--2021-07-31 22:15:43--  https://storage.googleapis.com/inspirit-ai-data-bucket-1/Data/A
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.195.128, 74.125.199
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.195.128|:443... cor
HTTP request sent, awaiting response... 200 OK
Length: 553377 (540K) [text/csv]
Saving to: ‘metadata.csv’

metadata.csv        100%[===================>] 540.41K  --.-KB/s    in 0.004s

2021-07-31 22:15:43 (129 MB/s) - ‘metadata.csv’ saved [553377/553377]

--2021-07-31 22:15:43--  https://storage.googleapis.com/inspirit-ai-data-bucket-1/Data/A
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.142.128, 74.125.195
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.142.128|:443... cor
HTTP request sent, awaiting response... 200 OK
Length: 1366522108 (1.3G) [application/zip]
Saving to: ‘images_1.zip’

images_1.zip        100%[===================>]   1.27G   238MB/s    in 5.2s

2021-07-31 22:15:48 (252 MB/s) - ‘images_1.zip’ saved [1366522108/1366522108]

--2021-07-31 22:15:49--  https://storage.googleapis.com/inspirit-ai-data-bucket-1/Data/A
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.142.128, 74.125.195
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.142.128|:443... cor
HTTP request sent, awaiting response... 200 OK
Length: 1403566547 (1.3G) [application/zip]
Saving to: ‘images_2.zip’

images_2.zip        100%[===================>]   1.31G  44.1MB/s    in 34s

2021-07-31 22:16:22 (39.9 MB/s) - ‘images_2.zip’ saved [1403566547/1403566547]

--2021-07-31 22:16:22--  https://storage.googleapis.com/inspirit-ai-data-bucket-1/Data/A
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.195.128, 74.125.199
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.195.128|:443... cor
HTTP request sent, awaiting response... 200 OK
Length: 7524968 (7.2M) [text/csv]
Saving to: ‘hmnist_8_8_RGB.csv’

hmnist_8_8_RGB.csv  100%[===================>]   7.18M  41.6MB/s    in 0.2s

2021-07-31 22:16:23 (41.6 MB/s) - ‘hmnist_8_8_RGB.csv’ saved [7524968/7524968]
```

```
IMG_WIDTH = 100
IMG_HEIGHT = 75
```

```
     |████████████████████████████████| 77 kB 4.9 MB/s
```

```
X = []
```

```
X_gray = []

y = []


# initialize X, X_gray, and y variables
metadata = pd.read_csv(metadata_path)
metadata['category'] = metadata['dx'].replace({'basal': 0, 'HER2': 1, 'LuminalA': 2, 'ER': 3,


for i in tqdm(range(len(metadata))):
  image_meta = metadata.iloc[i]
  path = os.path.join(toDirectory, image_meta['image_id'] + '.jpg')
  img = cv2.imread(path,cv2.IMREAD_COLOR)
  img = cv2.resize(img,(IMG_WIDTH,IMG_HEIGHT))

  img_g = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  X_gray.append(img_g)

  X.append(img)
  y.append(image_meta['category'])

X_gray = np.array(X_gray)
X = np.array(X)
y = np.array(y)
```

```
100%  10015/10015 [01:31<00:00, 109.46it/s]
```

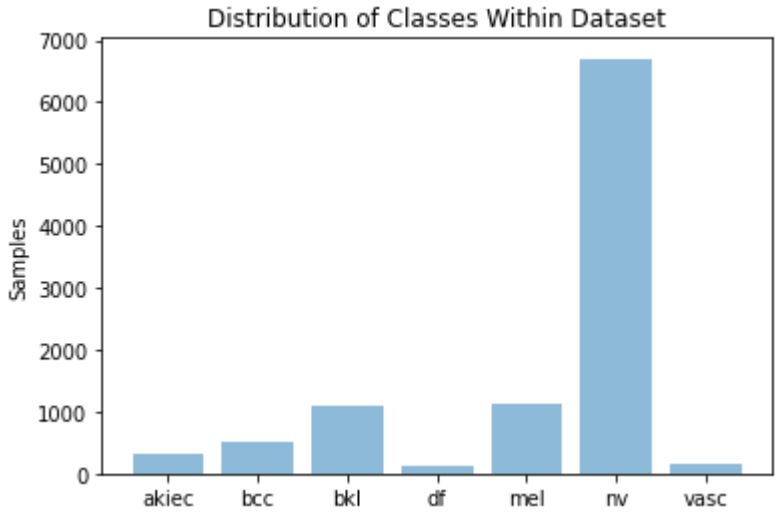
```
#looking at the shape of updated X, X_gray, and y variables
print(X_gray.shape)
print(X.shape)
print(y.shape)
```

```
(10015, 75, 100)
(10015, 75, 100, 3)
(10015,)
```

```
#plot the distribution of our dataset
objects = ('akiec', 'bcc', 'bkl', 'df', 'mel', 'nv', 'vasc')
y_pos = np.arange(len(objects))
occurances = []

for obj in objects:
  occurances.append(np.count_nonzero(obj == metadata['dx']))

print(occurances)

plt.bar(y_pos, occurances, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Samples')
```

```
plt.title('Distribution of Classes Within Dataset')

plt.show()
```

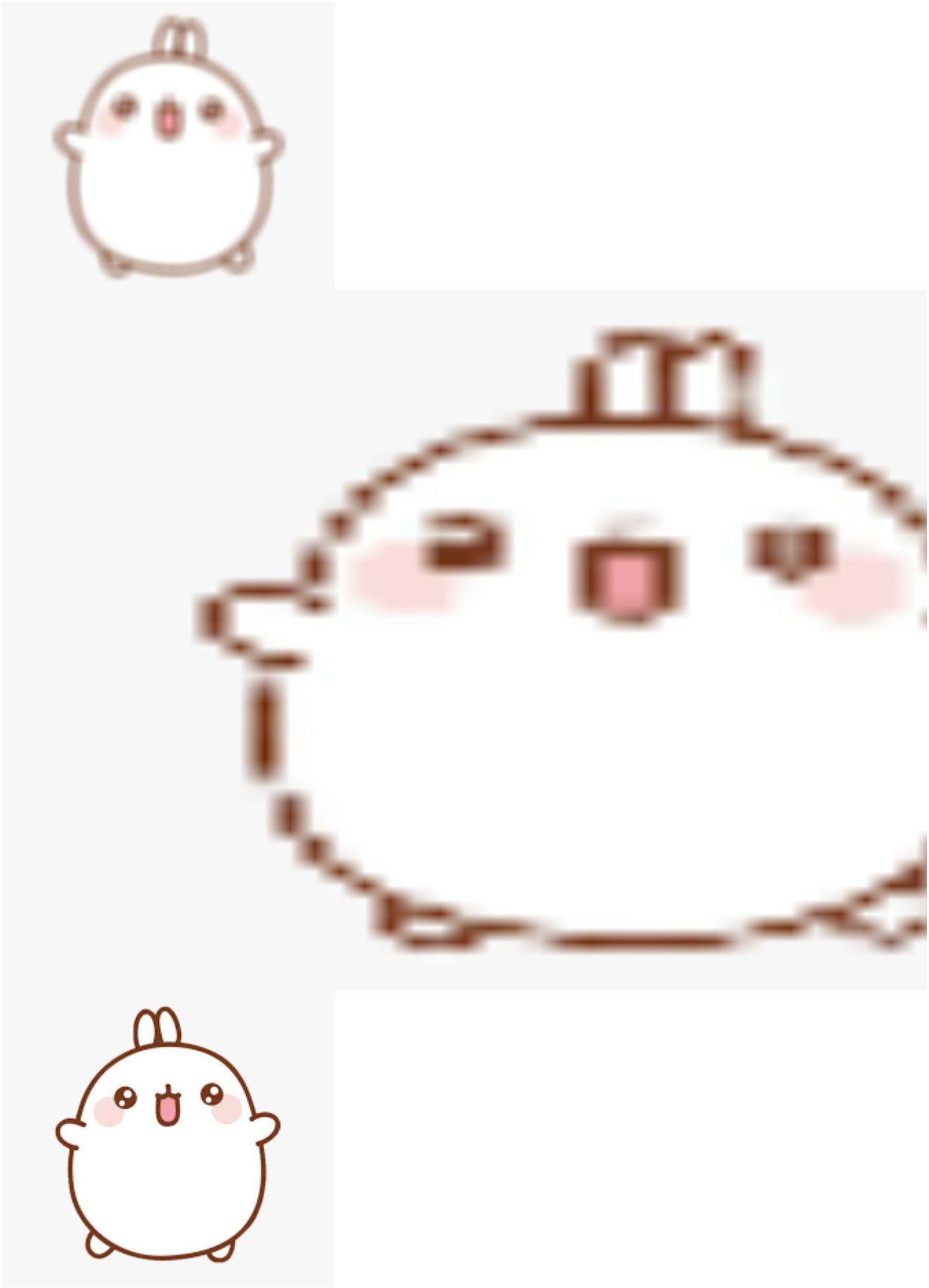
```
[327, 514, 1099, 115, 1113, 6705, 142]
```




```
sample_cap = 142
option = 1


#Run this to reduce dataset size. This method caps each class at *sample_cap* samples.
if (option == 1):
  objects = ['akiec', 'bcc', 'bkl', 'df', 'mel', 'nv', 'vasc']
  class_totals = [0,0,0,0,0,0,0]
  iter_samples = [0,0,0,0,0,0,0]
  indicies = []

  for i in range(len(X)):
    class_totals[y[i]] += 1

  print("Initial Class Samples")
  print(class_totals)

  for i in range(len(X)):
    if iter_samples[y[i]] != sample_cap:
      indicies.append(i)
      iter_samples[y[i]] += 1

  X = X[indicies]
  X_gray = X_gray[indicies]

  y = y[indicies]

  class_totals = [0,0,0,0,0,0,0]

  for i in range(len(X)):
    class_totals[y[i]] += 1
```

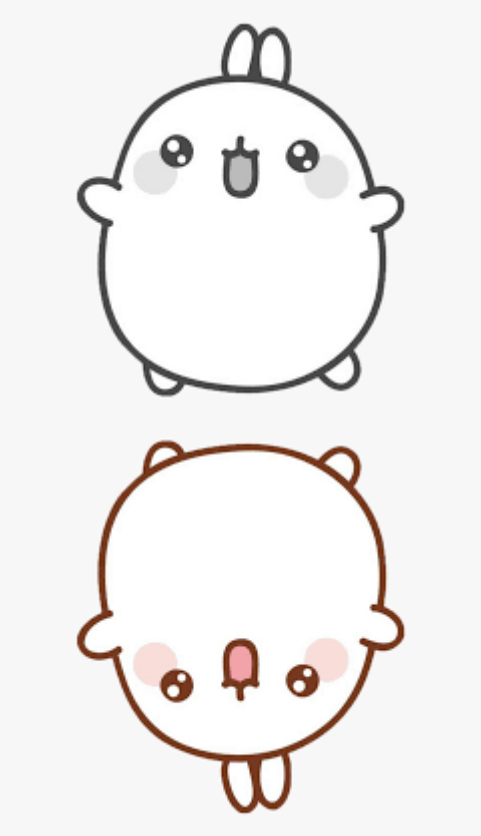
```
  print("Modified Class Samples")
  print(class_totals)
else:
  print("This option was not selected")


#OPEN CV IMAGE MANIPULATION!

jaguar = cv2.imread("kawaii_molang.png")
cv2_imshow(jaguar)
```

```
image = cv2.blur(jaguar,(10,10)) #blurred img
cv2_imshow(image)

image = cv2.resize(jaguar,(50, 50)) #resized/pixelated img
image = cv2.resize(image,(910, 510))
cv2_imshow(image)


new_image = cv2.flip(jaguar, 1) #reflected/flipped img
cv2_imshow(new_image)

gray_img = cv2.cvtColor(jaguar ,cv2.COLOR_BGR2GRAY) #gray-scaled
cv2_imshow(gray_img)
```



```
# Grayscale
jaguan_bw = cv2.cvtColor(jaguan,cv2.COLOR_BGR2GRAY)
```

```
jaguar_bw = cv2.cvtColor(jaguar,cv2.COLOR_BGR2GRAY)
cv2_imshow(jaguar_bw)

# Flip
jaguar_flip = cv2.flip(jaguar,0)
cv2_imshow(jaguar_flip)
```



```
#Zoom into our image
zoom = 0.33

centerX,centerY=int(jaguar.shape[0]/2),int(jaguar.shape[1]/2)
radiusX,radiusY= int((1-zoom)*jaguar.shape[0]*2),int((1-zoom)*jaguar.shape[1]*2)

minX,maxX=centerX-radiusX,centerX+radiusX
minY,maxY=centerY-radiusY,centerY+radiusY

cropped = jaguar[minX:maxX, minY:maxY]
zoom_img = cv2.resize(cropped, (jaguar.shape[1], jaguar.shape[0]))
cv2_imshow(zoom_img)
```

```
#DATA AUGMENTATION!

X_gray_train, X_gray_test, y_train, y_test = train_test_split(X_gray, y, test_size=0.4, rando
```

```
#also do a test/train split for x + y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)


#randomly decide to flip the image across the y-axis or apply a 33% zoom

X_augmented = []
X_gray_augmented = []

y_augmented = []

for i in tqdm(range(len(X_train))):
  transform = random.randint(0,1)
  if (transform == 0):
    # Flip the image across the y-axis
    X_augmented.append(cv2.flip(X_train[i],1))
    X_gray_augmented.append(cv2.flip(X_gray_train[i],1))
    y_augmented.append(y_train[i])
  else:
    # Zoom 33% into the image
    zoom = 0.33

    centerX,centerY=int(IMG_HEIGHT/2),int(IMG_WIDTH/2)
    radiusX,radiusY= int((1-zoom)*IMG_HEIGHT*2),int((1-zoom)*IMG_WIDTH*2)

    minX,maxX=centerX-radiusX,centerX+radiusX
    minY,maxY=centerY-radiusY,centerY+radiusY

    cropped = (X_train[i])[minX:maxX, minY:maxY]
    new_img = cv2.resize(cropped, (IMG_WIDTH, IMG_HEIGHT))
    X_augmented.append(new_img)

    cropped = (X_gray_train[i])[minX:maxX, minY:maxY]
    new_img = cv2.resize(cropped, (IMG_WIDTH, IMG_HEIGHT))
    X_gray_augmented.append(new_img)

    y_augmented.append(y_train[i])

X_augmented = np.array(X_augmented)
X_gray_augmented = np.array(X_gray_augmented)

y_augmented = np.array(y_augmented)
```

```
X_train = np.vstack((X_train,X_augmented))
X_gray_train = np.vstack((X_gray_train,X_gray_augmented))

y_train = np.append(y_train,y_augmented)
```

```
100%  6009/6009 [00:00<00:00, 16181.03it/s]
```

```
#Combine Augmented Data with Existing Samples
X_augmented = np.array(X_augmented)
X_gray_augmented = np.array(X_gray_augmented)

y_augmented = np.array(y_augmented)

X_train = np.vstack((X_train,X_augmented))
X_gray_train = np.vstack((X_gray_train,X_gray_augmented))

y_train = np.append(y_train,y_augmented)


print(X_gray_train.shape)
print(X_train.shape)
print(y_train.shape)
```

```
(18027, 75, 100)
(18027, 75, 100, 3)
(18027,)
```

```
#two additional data augmentation examples
X_augmented = []
X_gray_augmented = []

y_augmented = []

for i in tqdm(range(len(X_train))):
  transform = random.randint(0,1)
  if (transform == 0):

    # Resize the image by half on each dimension, and resize back to original
    # dimensions

    small_image = cv2.resize(X_train[i],(IMG_WIDTH//2,IMG_HEIGHT//2))
    normal_image = cv2.resize(small_image,(IMG_WIDTH,IMG_HEIGHT))

    small_grayscale_image = cv2.resize(X_gray_train[i],(IMG_WIDTH//2,IMG_HEIGHT//2))
    normal_grayscale_image = cv2.resize(small_grayscale_image,(IMG_WIDTH,IMG_HEIGHT))

    X_augmented.append(normal_image)
    X_gray_augmented.append(normal_grayscale_image)
    y_augmented.append(y_train[i])
```

```
    else:

      # Blur the image with a 4 x 4 kernel

      X_augmented.append(cv2.blur(X_train[i],(4,4)))
      X_gray_augmented.append(cv2.blur(X_gray_train[i],(4,4)))
      y_augmented.append(y_train[i])
```

```
100%|| 18027/18027 [00:01<00:00, 11743.45it/s]
```

```
# Combine Augmented Data with Existing Samples
X_augmented = np.array(X_augmented)
X_gray_augmented = np.array(X_gray_augmented)

y_augmented = np.array(y_augmented)

X_train = np.vstack((X_train,X_augmented))
X_gray_train = np.vstack((X_gray_train,X_gray_augmented))

y_train = np.append(y_train,y_augmented)


#CREATE KNN MODEL
knn = KNeighborsClassifier(n_neighbors=5)


#PERFORMING IMAGE FLATTENING

X_g_train_flat = X_gray_train.reshape(X_gray_train.shape[0],-1)
X_g_test_flat = X_gray_test.reshape(X_gray_test.shape[0],-1)
print (X_g_train_flat.shape)
print (X_g_test_flat.shape)
```

```
(36054, 7500)
(4006, 7500)
```

```
knn.fit(X_g_train_flat, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
def model_stats(name, y_test, y_pred, y_pred_proba):
  cm = confusion_matrix(y_test, y_pred)

  print(name)

  accuracy = accuracy_score(y_test,y_pred)
  print ("The accuracy of the model is " + str(round(accuracy,5)))
```

```
  roc_score = roc_auc_score(y_test, y_pred_proba, multi_class='ovo')

  print ("The ROC AUC Score of the model is " + str(round(roc_score,5)))

  return cm


y_pred = knn.predict(X_g_test_flat)
y_pred_proba = knn.predict_proba(X_g_test_flat)

knn_cm = model_stats("K Nearest Neighbors",y_test,y_pred,y_pred_proba)


def plot_cm(name, cm):
  classes = ['akiec', 'bcc', 'bkl', 'df', 'mel', 'nv', 'vasc']
  df_cm = pd.DataFrame(cm, index = [i for i in classes], columns = [i for i in classes])
  df_cm = df_cm.round(5)

  plt.figure(figsize = (12,8))
  sns.heatmap(df_cm, annot=True, fmt='g')
  plt.title(name + " Model Confusion Matrix")
  plt.xlabel("Predicted Label")
  plt.ylabel("True Label")
  plt.show()


#new function for KNN classifier
plot_cm("K Nearest Neighbors",knn_cm)


X_gray_test, X_gray_val, y_g_test, y_g_val = train_test_split(X_gray_test, y_test, test_size=

X_gray_test_flat = np.reshape(X_gray_test,(X_gray_test.shape[0],X_gray_test.shape[1]*X_gray_t
X_gray_val_flat = np.reshape(X_gray_val,(X_gray_val.shape[0],X_gray_val.shape[1]*X_gray_val.s


X_gray_test.shape


param_grid = {
              'n_neighbors' :     [2, 3, 4, 5],
              'weights' :          ['uniform', 'distance'],
              'algorithm' :        ['ball_tree', 'kd_tree', 'brute']
             }


gs_knn = GridSearch(model=KNeighborsClassifier(),param_grid=param_grid)

gs_knn.fit(X_g_train_flat.astype(np.float32), y_train.astype(np.float32),
       X_gray_val_flat.astype(np.float32), y_g_val.astype(np.float32),verbose=1)


y_pred = gs_knn.predict(X_gray_test_flat)
y_pred_proba = gs_knn.predict_proba(X_gray_test_flat)
```