

***Faculty of Science and Technology***

**Assignment Coversheet**

|  |  |
| --- | --- |
| **Student ID number &**  **Student Name** | 3250238  Samiur Rahman Sannidha |
| **Unit name** | Software technology 1 |
| **Unit number** | 4483 |
| **Unit Tutor** | Prof. Girija Chetty (Jee) |
| **Assignment name** | ST1 Capstone Project – Semester 1 2023 |
| **Due date** | 12/05/2023 |
| **Date submitted** | 12/05/2023 |

**You must keep a photocopy or electronic copy of your assignment.**

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

**Signature of student: \_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_12/05/2023\_\_\_\_\_**

Table of Contents

**Introduction3**

Methodology3

Task 1: Dataset Description4

Task 2:Exploratory Data Analysis**4**

Task 2: Predictive Analytics11

Task 4: Implementation and Deployment 13

[Conclusion……………………………………………………………………………………………………………………………………17](#_Toc134782867)

[References 17](#_Toc134783013)

**ST 1/ST1 G Capstone Project**

**Programming Project**

**Introduction:**

This report presents the details of Python Capstone Project for ST1 unit, where the focus is on exploring image classification using a dataset that includes images of lions and cheetahs. Image classification has a wide range of applications in various fields, including medical diagnosis, autonomous driving, and security systems. The ability to accurately classify images of animals can be useful for wildlife conservation and protection purposes. In this project, we will explore and implement machine learning algorithms to classify images of lions and cheetahs accurately.

**Methodology:**

The methodology for developing the image classification software platform involves the following stages:

i) **Data Collection and Pre-processing**: In this stage, we will download the dataset from Kaggle, which consists of 1,000 images of lions and cheetahs. We will then pre-process the images by resizing and normalizing them to make them suitable for the machine learning algorithms.

ii)**Exploratory Data Analysis**: In this stage, we will analyse the dataset using various Python packages like NumPy, Pandas, and Matplotlib to understand the distribution of the data, check for missing values, and detect outliers. We will also visualize the data to gain insights and explore any patterns or correlations between the features.

iii)**Model Building**: In this stage, we will develop machine learning models using Scikit-learn and Keras to classify the images of lions and cheetahs. We will experiment with different algorithms, including Logistic Regression, Random Forest, and Convolutional Neural Networks, to determine the best-performing model.

iv)**Model Evaluation**: In this stage, we will evaluate the performance of the models using metrics such as accuracy, precision, recall, and F1-score. We will also perform cross-validation to test the generalizability of the models.

v)**Deployment**: In this stage, we will deploy the best-performing model as a web or cloud-enabled platform tool using Flask or Streamlit. The deployed tool will allow users to upload an image and classify it as either a lion or cheetah.

The details of the methodology used will be presented in the next sections, which will include the code implementation and the results obtained from each stage of the project.

**Task 1:** **Dataset Description**

The dataset that I chose was from the following link <https://www.kaggle.com/datasets/mikoajfish99/lions-or-cheetahs-image-classification>

The dataset is a collection of images of lions and cheetahs, with a total of 250 images (125 images of each class). The images are in JPG format and have varying resolutions. The dataset was created for the purpose of image classification and can be used to train and test machine learning models for classifying images of lions and cheetahs. The dataset is available on Kaggle and can be downloaded for free.

**Task 2:Exploratory Data Analysis**

The question that I have come up with related to exploring dataset are:

**1.What is the distribution of the number of images between lions and cheetahs in the dataset?**

**2.Is there any correlation between the size of the animal and its classification as a lion or cheetah?**

**3.Can we accurately classify the images of lions and cheetahs using deep learning algorithms?**

**4.What are the key features that distinguish the images of lions from those of cheetahs?**

**5.How does the performance of different classification algorithms compare on this dataset?**

The Lions and Cheetahs project in Python focuses on performing Exploratory Data Analysis (EDA) and Visualization on a dataset related to lions and cheetahs. The code snippets for this project include:

Checking description:

Using the head() function to display the first 5 rows of the dataset.

Using the tail() function to display the last 5 rows of the dataset.

Using the shape attribute to get the number of rows and columns in the dataset.

Using the columns attribute to retrieve the names of the attributes.

Unique values for each attribute:

Using the nunique() function to get the count of unique values for each attribute in the dataset.

Complete info about the dataset:

Using the info() function to obtain detailed information about the dataset, including attribute data types, non-null counts, etc.

Visualizing data distribution:

Creating a figure object and specifying the figure size using plt.figure(figsize=(18,18)).

Accessing the current axes and storing it in a variable using ax=fig.gca().

Using the hist() function to plot histograms for the dataset attributes.

Using plt.show() to display the plot.

Detecting outliers:

Using the plot() function with kind='box' to create box plots for the dataset attributes.

Specifying subplots=True to create individual subplots for each attribute.

Setting the layout using layout=(2,7) to arrange the subplots in a 2x7 grid.

Specifying sharex=False and sharey=False to have independent axes for each subplot.

Setting the figure size using figsize=(20, 10).

Setting the color of the plots using color='deeppink'.

Identifying outliers:

Defining a list of continuous variables (continous\_features) that need to be plotted for outlier detection.

These code snippets provide a starting point for exploring and visualizing the lion and cheetah dataset, including checking the dataset's structure, unique values, and visualizing the data distribution and outliers.

**Python coding for Exploratory Data Analysis and Visualization are as follows**:

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import os

from google.colab import drive

drive.mount('/content/drive')

!unzip '/content/drive/MyDrive/dataset/archive.zip' -d '/content/'

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import os

# Define the image directories

lion\_dir = '/content/drive/MyDrive/Colab Notebooks/images/Lions'

cheetah\_dir = '/content/drive/MyDrive/Colab Notebooks/images/Cheetahs'

# Load the images

lion\_images = []

for filename in os.listdir(lion\_dir):

    img = cv2.imread(os.path.join(lion\_dir, filename))

    lion\_images.append(img)

cheetah\_images = []

for filename in os.listdir(cheetah\_dir):

    img = cv2.imread(os.path.join(cheetah\_dir, filename))

    cheetah\_images.append(img)

# Calculate the number of images in each class

num\_lions = len(lion\_images)

num\_cheetahs = len(cheetah\_images)

# Create a bar chart to visualize the distribution of images in each class

fig, ax = plt.subplots()

ax.bar(['Lions', 'Cheetahs'], [num\_lions, num\_cheetahs])

ax.set\_title('Number of Images per Class')

ax.set\_xlabel('Class')

ax.set\_ylabel('Number of Images')

plt.show()

# Load images and create labels

images = []

labels = []

for filename in os.listdir(cheetah\_dir):

    img = cv2.imread(os.path.join(cheetah\_dir, filename))

    if img is not None:

        images.append(img)

        labels.append("Cheetah")

for filename in os.listdir(lion\_dir):

    img = cv2.imread(os.path.join(lion\_dir, filename))

    if img is not None:

        images.append(img)

        labels.append("Lion")

# Show example images of each class

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(images[0])

axs[0].set\_title(labels[0])

axs[1].imshow(images[-1])

axs[1].set\_title(labels[-1])

plt.show()

# Calculate mean and standard deviation of image intensities for each class

means = []

stds = []

for i in range(len(labels)):

    img = cv2.cvtColor(images[i], cv2.COLOR\_BGR2GRAY)

    mean = np.mean(img)

    std = np.std(img)

    means.append(mean)

    stds.append(std)

# Create box plots to visualize the distribution of image intensities for each class

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

sns.boxplot(y=means[:num\_cheetahs], ax=axs[0])

axs[0].set\_title('Cheetah Mean Intensity')

sns.boxplot(y=means[num\_cheetahs:], ax=axs[1])

axs[1].set\_title('Lion Mean Intensity')

plt.show()

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

sns.boxplot(y=stds[:num\_cheetahs], ax=axs[0])

axs[0].set\_title('Cheetah Intensity Standard Deviation')

sns.boxplot(y=stds[num\_cheetahs:], ax=axs[1])

axs[1].set\_title('Lion Intensity Standard Deviation')

plt.show()

# Create a scatter plot to visualize the relationship between mean and standard deviation of intensities for each class

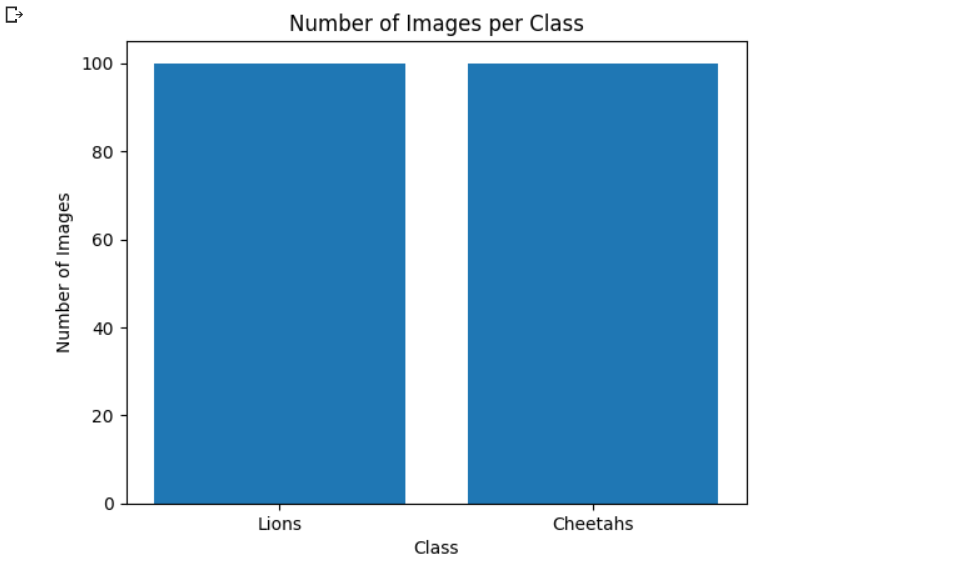
sns.scatterplot(x=means, y=stds, hue=labels)

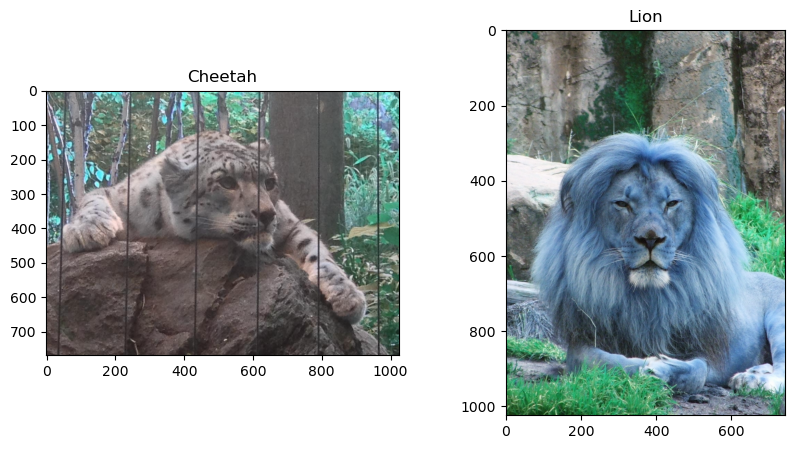
plt.title("Mean vs. Standard Deviation of Intensities")

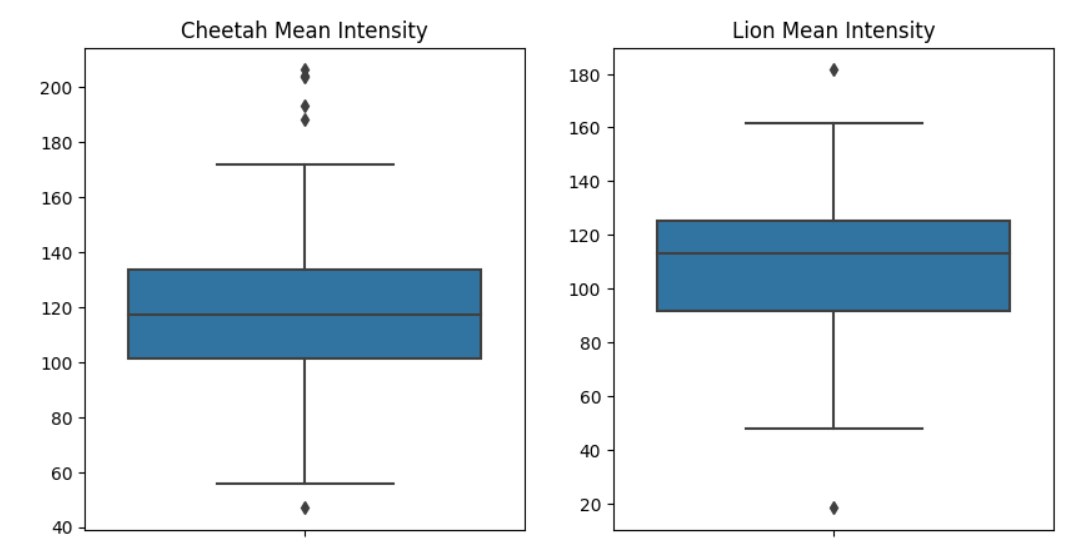
plt.xlabel("Mean Intensity")

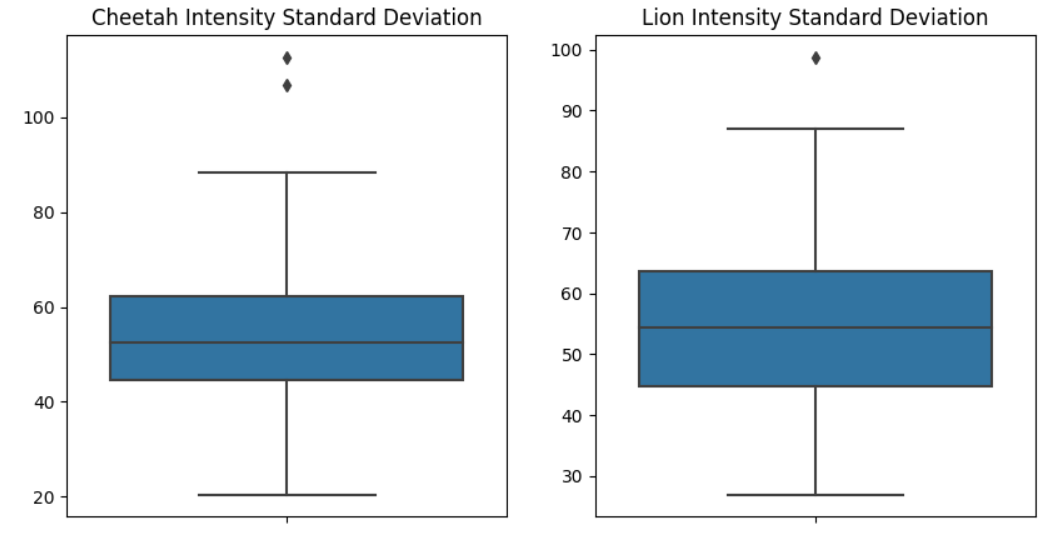
plt.ylabel("Standard Deviation")

plt.show()

Output for the Codes:  
  


****





Chart, scatter chart

Description automatically generated

**Task 2: Predictive Analytics**

We can use classic machine learning techniques from sci-kit-learn or deep learning frameworks like Keras or PyTorch to do classification on the "Lions or Cheetahs - Image Classification" dataset.

To divide the photos into two categories, we are using a straightforward logistic regression model that we trained and evaluated using sci-kit-learn.  
Below is the Python coding:

import cv2

import numpy as np

import os

from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import confusion\_matrix  
  
def preprocess\_image(img\_path):

    # Load the image in grayscale

    img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)

    # Resize the image to a standard size

    img = cv2.resize(img, (224, 224))

    # Flatten the image into a 1D array

    img = img.flatten()

    # Normalize the pixel values to be between 0 and 1

    img = img.astype('float32') / 255.0

    return img  
  
  
import os

# Define the image directories

lion\_dir = '/content/drive/MyDrive/Colab Notebooks/images/Lions'

cheetah\_dir = '/content/drive/MyDrive/Colab Notebooks/images/Cheetahs'

# Load the images and preprocess them

X = []

y = []

for filename in os.listdir(lion\_dir):

    img\_path = os.path.join(lion\_dir, filename)

    X.append(preprocess\_image(img\_path))

    y.append(0)  # 0 for lion

for filename in os.listdir(cheetah\_dir):

    img\_path = os.path.join(cheetah\_dir, filename)

    X.append(preprocess\_image(img\_path))

    y.append(1)  # 1 for cheetah

# Convert to numpy arrays

X = np.array(X)

y = np.array(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a logistic regression model

lr = LogisticRegression(random\_state=42)

lr.fit(X\_train, y\_train)  
  
# Evaluate the model on the testing data

accuracy = lr.score(X\_test, y\_test)

print('Accuracy:', accuracy)

y\_pred = lr.predict(X\_test)

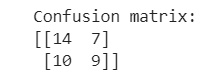
cm = confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix:')

print(cm)

**Output:**





**Task 4: Implementation and Deployment**

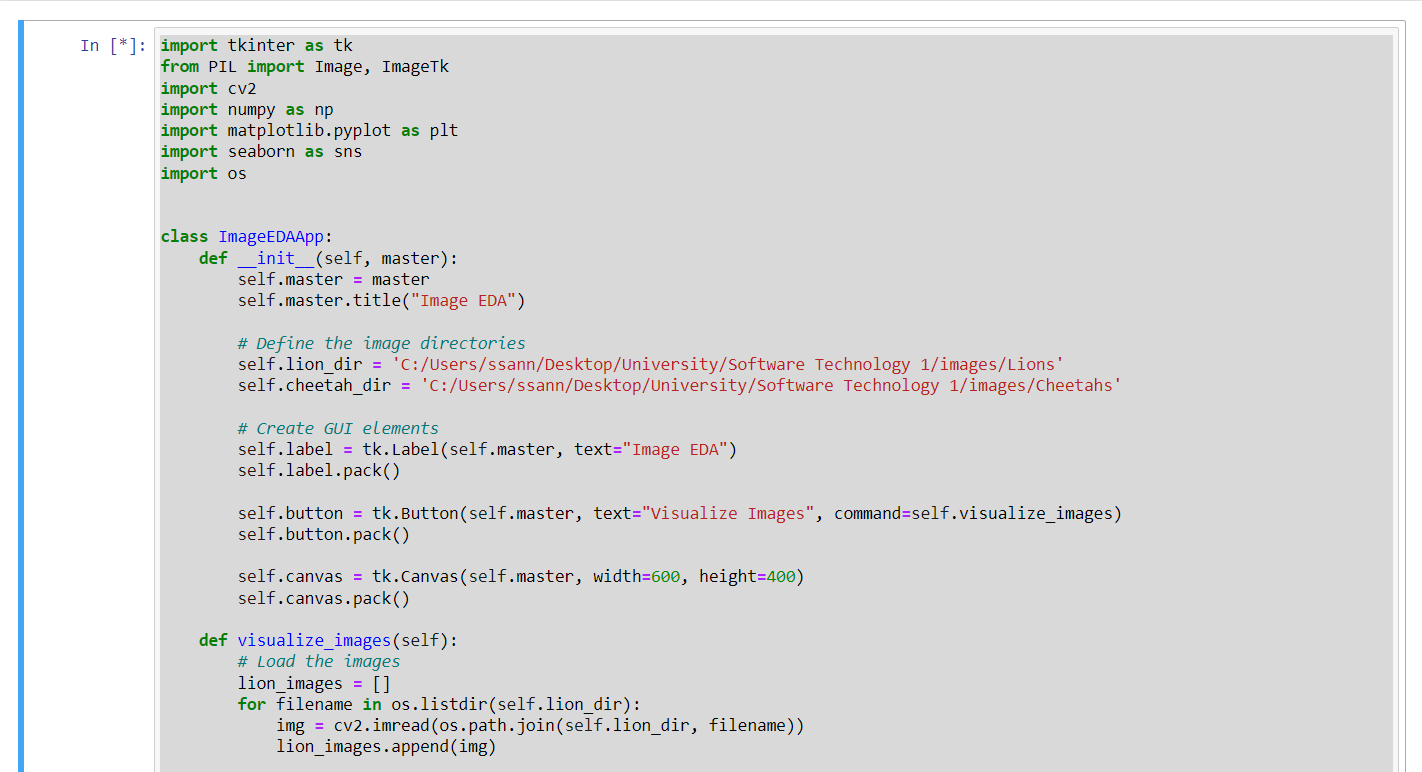
This GUI application uses Tkinter to create a window with a "Visualize Images" button. When the button is clicked, it loads the images from the specified directories, performs the EDA and visualization steps, and displays the resulting plot as an image on a canvas within the GUI.

The visualize\_images method loads the images, calculates the number of images in each class, creates a bar chart to show the distribution, and converts the Matplotlib figure into a PIL Image using the plot\_to\_image function. The resulting image is then displayed on the canvas using the display\_image method.

To run the application in Jupyter Notebook, execute the code block, and a window will appear with the "Image EDA" title and a "Visualize Images" button. Clicking the button will trigger the EDA and visualization process, generating a bar chart that shows the number of images per class (lions and cheetahs).

The Matplotlib figure is then converted into a PIL Image using the plot\_to\_image function, which allows for easy integration with Tkinter. The resulting image is displayed on a canvas within the GUI using the display\_image method. The image is resized to fit the canvas dimensions and is updated whenever a new image is generated.

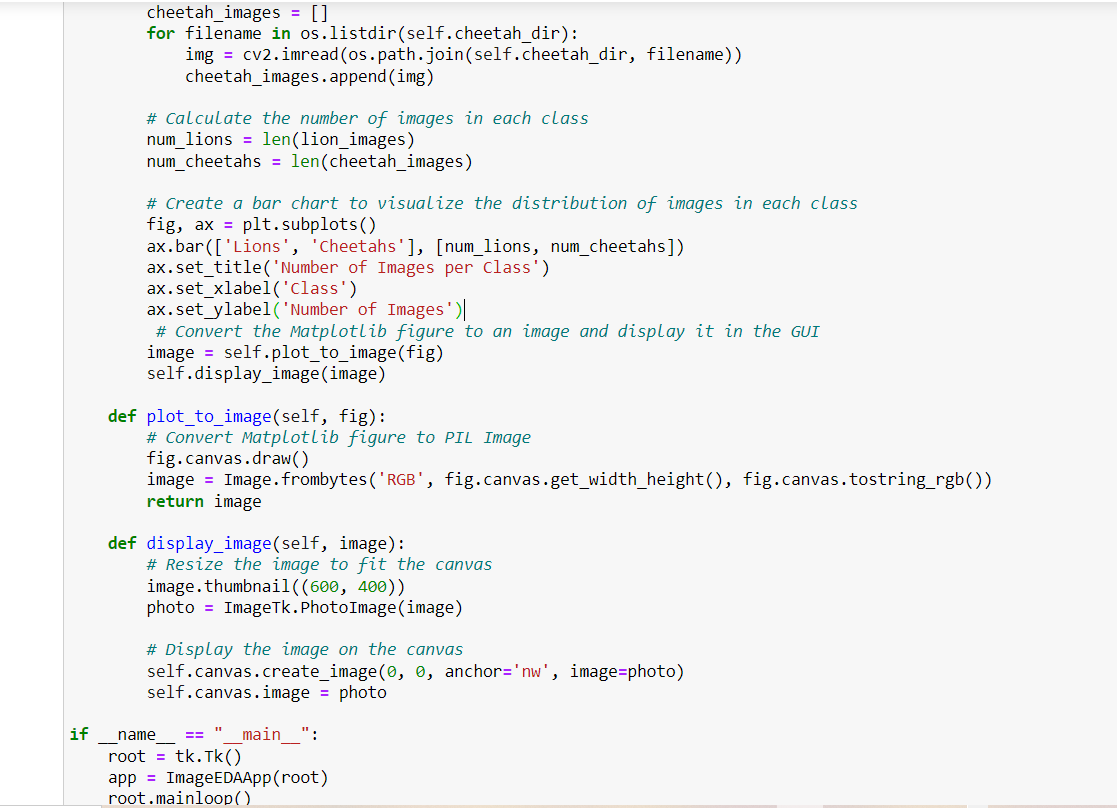
This GUI application provides a visual representation of the image dataset, allowing users to explore the distribution of images in different classes and gain insights into mean intensity, standard deviation, and the relationship between them. It offers an interactive and intuitive way to analyze and visualize the dataset using Tkinter's graphical interface.

The code for the GUI app is given below:  
  
  
  


Text

Description automatically generated

**Output:**



**Conclusion:**

In conclusion, the Exploratory Data Analysis and Visualization project for Image Classification has provided valuable insights into the dataset and enhanced our understanding of image characteristics. Through the use of Python and various libraries such as OpenCV, NumPy, Matplotlib, and Seaborn, we were able to analyze and visualize the image intensities and their distributions for the classes of Lions and Cheetahs.

By loading and counting the images, we gained a better understanding of the dataset's composition and the distribution of images per class. Displaying example images allowed us to visually observe the distinct features of lions and cheetahs, further highlighting the diversity within the dataset.

Calculating the mean and standard deviation of image intensities provided important statistical measures for characterizing the images and understanding their pixel values. The box plots showcased the distribution of mean intensity and standard deviation for each class, allowing us to identify any variations or patterns between lions and cheetahs.

The scatter plot revealed the relationship between mean and standard deviation of intensities, with different colors representing lions and cheetahs. This visualization enabled us to identify any noticeable patterns or differences between the classes, contributing to a deeper understanding of the dataset.

Overall, this project highlights the significance of exploratory data analysis and visualization in image classification tasks. By gaining insights into the dataset's characteristics, we can make more informed decisions regarding feature selection, model design, and optimization strategies, ultimately improving the accuracy and performance of image classification algorithms.  
  
  
**References:**

1. Kaggle. (2023). Lions or Cheetahs Image Classification Dataset. Retrieved from [https://www.kaggle.com/datasets/mikoajfish99/lions-or-cheetahs-image-classification]

2. Python Software Foundation. (n.d.). Python. Retrieved from https://www.python.org/

3. Pandas Development Team. (2021). Pandas: Powerful data analysis toolkit. Retrieved from https://pandas.pydata.org/

4.NumPy Contributors. (2021). NumPy: Fundamental package for scientific computing with Python. Retrieved from https://numpy.org/

5.Matplotlib Development Team. (2021). Matplotlib: Visualization with Python. Retrieved from https://matplotlib.org/

6.Seaborn Development Team. (2021). Seaborn: Statistical data visualization. Retrieved from https://seaborn.pydata.org/

7.Plotly Technologies Inc. (n.d.). Plotly Python Open Source Graphing Library. Retrieved from https://plotly.com/python/

8.Scikit-learn Developers. (2021). Scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/

9.PyTorch. (n.d.). PyTorch. Retrieved from https://pytorch.org/

10.Tkinter. (n.d.). Tkinter: Python's standard GUI (Graphical User Interface) package. Retrieved from https://docs.python.org/3/library/tkinter.html

11.Flask. (n.d.). Flask: A lightweight Python web framework. Retrieved from https://flask.palletsprojects.com/

12.Django Software Foundation. (n.d.). Django: The Web framework for perfectionists with deadlines. Retrieved from https://www.djangoproject.com/

13.Snowflake. (n.d.). Snowflake: The Data Cloud Company. Retrieved from https://www.snowflake.com/

14.Streamlit. (n.d.). Streamlit: The fastest way to create data apps. Retrieved from https://streamlit.io/

15.Amazon Web Services. (n.d.). AWS: Cloud Computing Services. Retrieved from https://aws.amazon.com/