Faculty of Science and Technology
Assignment Coversheet

Student ID number & Student Name	U3244786/Ahmed Anwar
Unit name	Software Technology 1
Unit number	4483
Unit Tutor	Girija Chetty
Assignment name	ST1 Capstone Project – Semester 1 2023
Due date	12th May 2023
Date submitted	12th May 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:** 12th May 2023

Table of Contents

Introduction 1
Methodology 3
Stage 1: Algorithm Design Stage 3
Dataset Description 4
Exploratory Data Analysis 4
Predictive Data Analytics Stage 11
Model Preparation and Development 16

Stage 2: Algorithm Implementation Stage 18
Stage 3: Software Deployment Stage 19
Conclusions 19
References 19
Appendix
Logbook journal
.... 20

Introduction

This report describes the details of Python Capstone Project for ST1 unit within the scope of the project requirements provided in the assignment handout [1]. I have decided to work on the project using a 10 Big Cats of the Wild dataset available in Kaggle data repositories [2].

The dataset "10 Big Cats of the Wild - Image Classification" on Kaggle offers a valuable resource for researchers and enthusiasts interested in studying big cats. It provides a diverse collection of labeled images featuring ten different species of wild cats, including lions, tigers, leopards, and more. This dataset can be used to train machine learning models for automated cat species identification, contributing to wildlife monitoring, conservation, and ecological research.

By utilizing this dataset, researchers can explore the potential of machine learning algorithms to accurately classify and identify wild cat species based on visual characteristics. The availability of this comprehensive and annotated dataset enables the development of robust image classification models, reducing the need for invasive methods and saving time and effort in species identification.

The "10 Big Cats of the Wild - Image Classification" dataset opens up new possibilities for non-invasive and efficient methods of wild cat species identification. Through computer vision and machine learning techniques, it becomes possible to automate the identification process, benefiting wildlife monitoring and conservation initiatives.

This dataset provides an opportunity to advance our understanding of big cat populations, their distribution patterns, and the challenges they face in the wild. By leveraging this dataset, researchers and data scientists can contribute to the conservation and protection of these magnificent creatures.

This report presents the details of prototype software platform, in terms of several Python software tools developed as part of this capstone project, based on a data driven scientific approach, involving exploratory data analysis, predictive analytics and implementation as a desktop Tkinter application, and online web-based Flask/ Streamlit application. The details of the methodology used is presented in the next Section.

Methodology

The methodology used for developing the software platform involves 2 stages as outlined below:

1. Design and development of decision support algorithms based on exploratory data analysis and predictive analytics, for identifying the best performing algorithm for solving a real world problem.
2. Implementation of best performing algorithm as a desktop Tkinter software tool.

Stage 1: Algorithm Design Stage

Stage 1 is most important preliminary stage and depending on the complexity of the problem and dataset used, the design of algorithms for exploratory data analysis and predictive analytics algorithms will vary. However, the workflow for algorithm development will be as outlined in the Figure 1 schematic shown below:

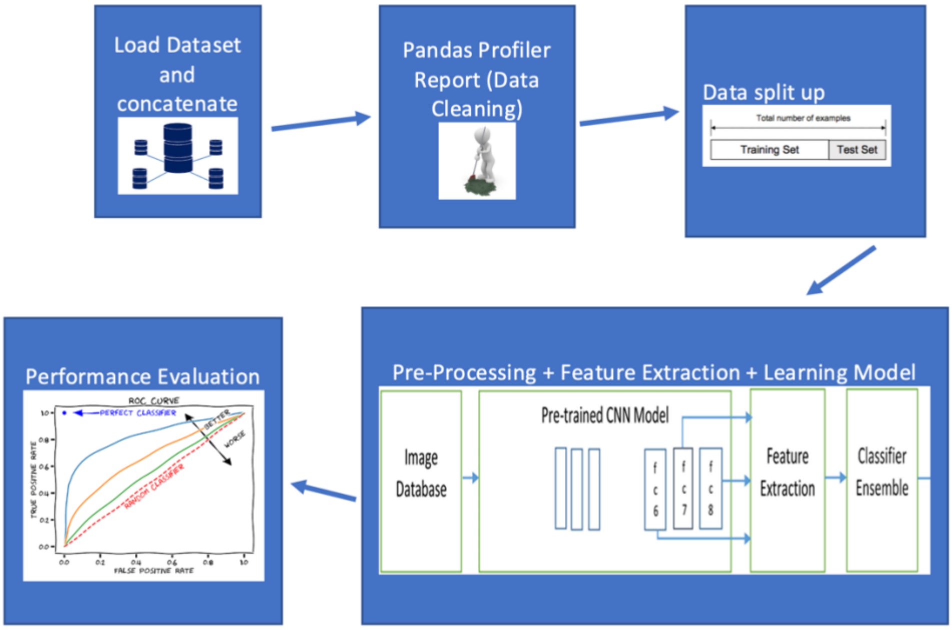


Figure 1: Schematic for Algorithm Design Methodology for Image Classification

The details of each building block in Figure 1 schematic for algorithm design is described in the next few sections.

Dataset Description

There is only one dataset used for this project and it is publicly available from Kaggle [2]. The dataset used for the 10 Big Cats of the Wild - Image Classification project is publicly available on Kaggle. It consists of a collection of images representing ten different species of wild cats. The dataset contains labeled images for each species, allowing for supervised learning approaches in image classification.

In total, the dataset includes images of ten big cat species, such as lions, tigers, leopards, and more. These images can be used to train machine learning models for automated classification of wild cat species based on visual characteristics. These models can assist in automatically identifying and categorizing wild cat species, reducing the need for manual intervention and saving time in species identification.

Exploratory Data Analysis

The first phase of the software development activity involved understanding the data, basic exploratory data analysis and visualisation. Google Colab was chosen as the experimental environment as it incorporates virtual hardware and resources which does not require additional physical hardware requirement and can be ran directly of a web browser. The python language was used to create the scripts which ran directly on online Jupyter notebook using Google Colab with the help of free google account created, and by saving all the notebook files virtually on google drive without additional configurations. Before the exploratory data analysis can begin, some of the python libraries for EDA need to be imported and dataset acquired, by using the following Python script

```
from google.colab import drive
drive.mount("/content/drive")


#Import Required Packages for EDA
import numpy as np
import pandas as pd
import os

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib

import keras.backend as K
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint
from tensorflow.keras import Model
from tensorflow.keras.layers.experimental import preprocessing

matplotlib.style.use('ggplot')
%matplotlib inline
# Set some constant parameters
IMAGE_SHAPE = (224, 224)
EPOCHS = 15
BATCH_SIZE = 32

# Specify paths to datasets
TRAIN_DATA_DIR = '/content/drive/MyDrive/10 big cats of the wild kaggle
dataset/train'
TEST_DATA_DIR = '/content/drive/MyDrive/10 big cats of the wild kaggle
dataset/test'
VALID_DATA_DIR = '/content/drive/MyDrive/10 big cats of the wild kaggle
dataset/valid'
```

(1) The EDA starts with understanding the basic description of data as described next:

```
#1. Checking description of dataset categorically
```

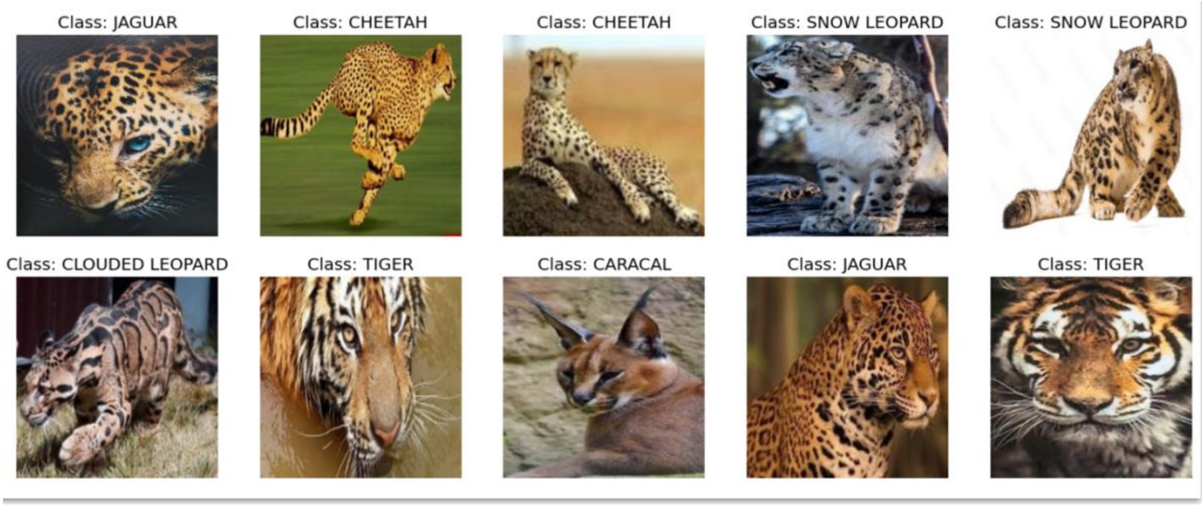
```python
# Load the data
datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255,
horizontal_flip=True)

train_generator = datagen.flow_from_directory(
    directory=TRAIN_DATA_DIR,
    shuffle=True,
    color_mode='rgb',
    class_mode='categorical',
    seed=1234,
    target_size=IMAGE_SHAPE,
    batch_size=BATCH_SIZE
)

valid_generator = datagen.flow_from_directory(
    directory=VALID_DATA_DIR,
    shuffle=True,
    color_mode='rgb',
    class_mode='categorical',
    seed=1234,
    target_size=IMAGE_SHAPE,
    batch_size=BATCH_SIZE
)

test_generator = datagen.flow_from_directory(
    directory=TEST_DATA_DIR,
    shuffle=False,
    color_mode='rgb',
    class_mode='categorical',
    target_size=IMAGE_SHAPE,
    batch_size=BATCH_SIZE
)
```

```
Found 2339 images belonging to 10 classes.
Found 50 images belonging to 10 classes.
Found 50 images belonging to 10 classes.
```

```python
# Checking one example
example = '/content/drive/MyDrive/10 big cats of the wild kaggle
dataset/train/CHEETAH/001.jpg'

image = mpimg.imread(example)

# Print out the image dimensions
print('Image dimensions:', image.shape)
plt.imshow(image)
plt.axis("off")
```

```
Image dimensions: (224, 224, 3)
(-0.5, 223.5, 223.5, -0.5)
```

```
# Collect the labels
labels = [l for l in train_generator.class_indices]
# Take the next batch out of train_generator
samples = next(train_generator)
# Store info on pixels of each image in images
images = samples[0]
# Store info on the class in titles
titles = samples[1]

# Set figure size
plt.figure(figsize=(18,18))

# Create subplots
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.imshow(images[i])
    plt.title(f"Class: {labels[np.argmax(titles[i],axis=0)]}")
    plt.axis("off")
```



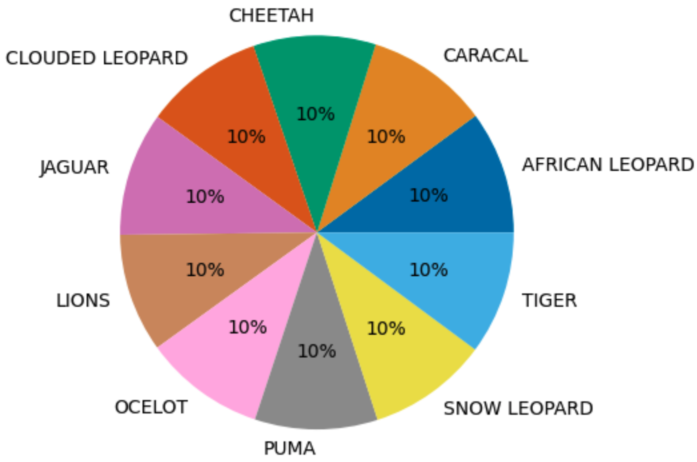
#2. Visualising data distribution in detail

```
# Check for potential class imbalance
names = ['Train dataset', 'Validation dataset', 'Test dataset']

for gen, name in zip([train_generator, valid_generator, test_generator], names):
    labels = [l for l in gen.class_indices]
    unique, counts = np.unique(gen.classes, return_counts=True)
    df = pd.DataFrame(data={'class_name': labels, 'class_index' :
unique,'instances':counts})
    total_instances = sum(df['instances'])
    df['class_percent'] = df['instances'] / total_instances

    print("\n", name)
    print("\n", df)
    print("\n", f"Total number of instances:{total_instances}")

    palette_color = sns.color_palette("colorblind")

    # Plotting data on chart
    plt.pie(x=df['class_percent'], labels=df['class_name'], autopct='%.0f%%',
colors=palette_color)


# Displaying chart
    plt.show()
```

Train dataset

	class_name	class_index	instances	class_percent
0	AFRICAN LEOPARD	0	236	0.100898
1	CARACAL	1	236	0.100898
2	CHEETAH	2	235	0.100470
3	CLOUDED LEOPARD	3	229	0.097905
4	JAGUAR	4	238	0.101753
5	LIONS	5	228	0.097478
6	OCELOT	6	233	0.099615
7	PUMA	7	236	0.100898
8	SNOW LEOPARD	8	231	0.098760
9	TIGER	9	237	0.101325

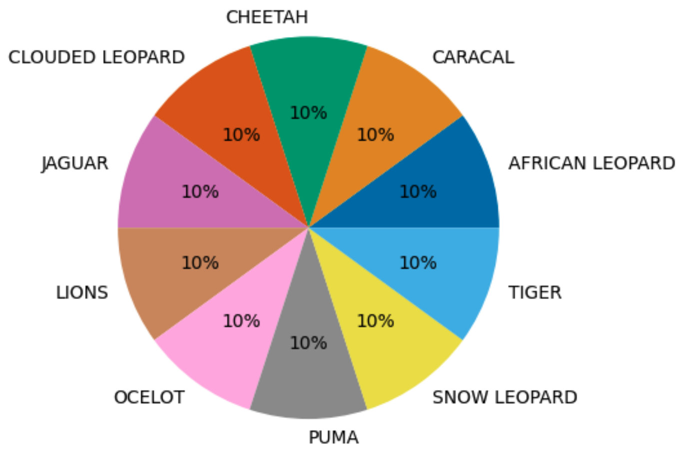
Total number of instances:2339



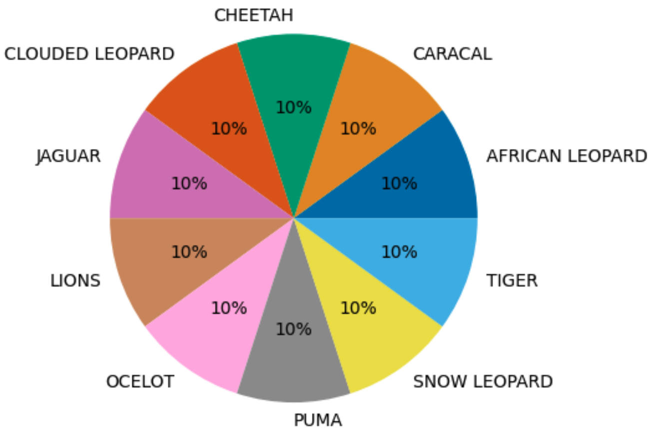
Validation dataset

	class_name	class_index	instances	class_percent
0	AFRICAN LEOPARD	0	5	0.1
1	CARACAL	1	5	0.1
2	CHEETAH	2	5	0.1
3	CLOUDED LEOPARD	3	5	0.1
4	JAGUAR	4	5	0.1
5	LIONS	5	5	0.1
6	OCELOT	6	5	0.1
7	PUMA	7	5	0.1
8	SNOW LEOPARD	8	5	0.1
9	TIGER	9	5	0.1

Total number of instances:50



```
Test dataset

         class_name  class_index  instances  class_percent
0   AFRICAN LEOPARD            0          5            0.1
1           CARACAL            1          5            0.1
2           CHEETAH            2          5            0.1
3   CLOUDED LEOPARD            3          5            0.1
4            JAGUAR            4          5            0.1
5             LIONS            5          5            0.1
6            OCELOT            6          5            0.1
7              PUMA            7          5            0.1
8      SNOW LEOPARD            8          5            0.1
9             TIGER            9          5            0.1

Total number of instances:50
```




```
# Displaying classes again using breed count way to check if possible error and
classes coming up as breeds
breed_counts = {}

for root, dirs, files in os.walk('/content/drive/MyDrive/10 big cats of the wild
kaggle dataset'):
    for file in files:
        if file.endswith('.jpg'):
            breed = os.path.basename(os.path.dirname(root))
            if breed not in breed_counts:
                breed_counts[breed] = 0
            breed_counts[breed] += 1

print("Breed distribution:", breed_counts)
```

Breed distribution: {'test': 50, 'valid': 50, 'train': 2339}

Predictive Data Analytics Stage

For predictive analytics, several processing steps are required usually. These include pre-processing, classifier comparison to identify the best machine learning classifier and performance evaluation with different objective metrics, such as accuracy, classification report, confusion matrix, ROC-AUC curve and prediction evaluation report. However, I have limited these stages by the use of EfficientNetB0 architecture and is explained below.

- Pre-processing: Since the dataset consists of a combination of continuous and categorical attributes/variables, there is a need to pre-process the data with attribute transformation, standardization and normalisation. However, in my case I was able to limit to minimal pre-processing due to the integration of Convolutional Neural Network (CNN).
- When using a pre-trained model like EfficientNetB0, the model architecture itself acts as the classifier. The pre-trained model has already learned to extract relevant features from images and has been trained on a large dataset, making it highly effective for image classification tasks.
- In traditional machine learning, it is common to compare multiple classifiers to identify the best performing one for a given task. However, with transfer learning and pre-trained models like EfficientNetB0, the need for classifier comparison is reduced. These models have already been extensively trained on a wide range of images and have shown superior performance in various image classification tasks. [3]
- By leveraging the pre-trained weights and feature extraction capabilities of EfficientNetB0, we can benefit from its high accuracy and efficiency without the need for extensive classifier comparison. It simplifies the process and allows us to focus more on fine-tuning the model and optimizing hyperparameters for your specific dataset and task. [3]

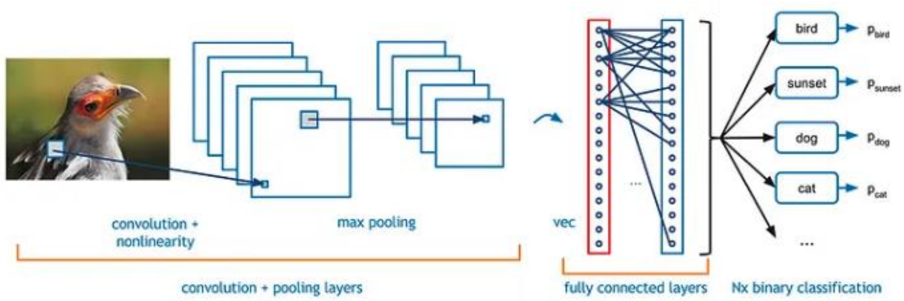


Figure 2:Showcases CNN; Image Source: [3]

#pre-processing (few steps are similar to EDA since PDA was done in separate notebook.

```python
from google.colab import drive
drive.mount('/content/drive')

# Import libraries for Predictive Analytics step
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as implt
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory

def plot_loss_curve(history):
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']

    epochs = range(len(loss))
    # Plot loss
    plt.plot(epochs, loss, label='training_loss')
    plt.plot(epochs, val_loss, label='val_loss')
    plt.title('Loss')
    plt.xlabel('Epochs')
    plt.legend()

    # Plot accuracy
    plt.figure()
    plt.plot(epochs, accuracy, label='training_accuracy')
    plt.plot(epochs, val_accuracy, label='val_accuracy')
    plt.title('Accuracy')
    plt.xlabel('Epochs')
    plt.legend();

# Create a variable for store our data folder's path
ROOT = '/content/drive/MyDrive/10 big cats of the wild kaggle dataset'
# Inspect our data folder
for dir_name, folder_names, file_names in os.walk(ROOT):
    print(f"There is {len(folder_names)} folders and {len(file_names)} files in
{dir_name}")
```

```
There is 3 folders and 4 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset
There is 10 folders and 1 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/AFRICAN LEOPARD
```

```
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/CHEETAH
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/JAGUAR
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/TIGER
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/LIONS
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/CLOUDED LEOPARD
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/PUMA
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/CARACAL
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/SNOW LEOPARD
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/test/OCELOT
There is 10 folders and 1 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/CHEETAH
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/JAGUAR
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/CLOUDED LEOPARD
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/LIONS
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/TIGER
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/AFRICAN LEOPARD
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/PUMA
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/CARACAL
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/OCELOT
There is 0 folders and 5 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/valid/SNOW LEOPARD
There is 10 folders and 1 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train
There is 0 folders and 237 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/TIGER
There is 0 folders and 236 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/CARACAL
There is 0 folders and 236 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/PUMA
There is 0 folders and 238 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/JAGUAR
There is 0 folders and 235 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/CHEETAH
There is 0 folders and 236 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/AFRICAN LEOPARD
There is 0 folders and 229 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/CLOUDED LEOPARD
There is 0 folders and 228 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/LIONS
There is 0 folders and 231 files in /content/drive/MyDrive/10 big cats of the wild
kaggle dataset/train/SNOW LEOPARD
```

```
There is 0 folders and 233 files in /content/drive/MyDrive/10 big cats of the

wild kaggle dataset/train/OCELOT
```

```
#Create train, validation and test directory
train_dir = ROOT + '/train/'
```

```python
valid_dir = ROOT + '/valid/'
test_dir = ROOT + '/test/'

# Create train, validation and test datasets
tf.random.set_seed(42)
IMAGE_SHAPE = (224, 224)
BATCH_SIZE = 32
print("Create train dataset...")
train_data = image_dataset_from_directory(directory=train_dir,
                                          image_size=IMAGE_SHAPE,
                                          batch_size=BATCH_SIZE,
                                          label_mode='categorical')
print("Create validation dataset...")
valid_data = image_dataset_from_directory(directory=valid_dir,
                                          image_size=IMAGE_SHAPE,
                                          batch_size=BATCH_SIZE,
                                          label_mode='categorical')
print("Create test dataset...")
test_data = image_dataset_from_directory(directory=test_dir,
                                         image_size=IMAGE_SHAPE,
                                         batch_size=BATCH_SIZE,
                                         label_mode='categorical')
```

```
Create train dataset...
Found 2339 files belonging to 10 classes.
Create validation dataset...
Found 50 files belonging to 10 classes.
Create test dataset...

   Found 50 files belonging to 10 classes.
```

```python
# Inspect our train dataset
train_data
```

```
<_BatchDataset element_spec=(TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None, 10), dtype=tf.float32,
name=None))>
```

```python
# Inspect our validation dataset
valid_data
```

```
<_BatchDataset element_spec=(TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None, 10), dtype=tf.float32,
name=None))>
```

```python
# Inspect our labels
train_data.class_names
```

```
['AFRICAN LEOPARD', 'CARACAL', 'CHEETAH', 'CLOUDED LEOPARD',
'JAGUAR', 'LIONS', 'OCELOT', 'PUMA', 'SNOW LEOPARD', 'TIGER']
```

```python
# Get a single batch for example
for image, label in train_data.take(1):
    print(image, label)
...
```

```
[[ 20.   17.   12.]
 [  3.    0.    0.]
```

```
   [ 21.  16.  12.]
   ...
   [ 93. 127.   6.]
   [ 94. 126.   3.]
   [ 93. 125.   2.]]

  [[ 26.  21.  17.]
   [  4.   0.   0.]
   [ 32.  27.  24.]
   ...
   [ 91. 122.   2.]
   [ 89. 121.   0.]
   [ 88. 120.   0.]]

  [[ 22.  17.  13.]
   [  5.   0.   0.]
   [ 40.  35.  32.]
   ...
   [ 87. 118.   0.]
   [ 85. 117.   0.]
   [ 84. 116.   0.]]]], shape=(32, 224, 224, 3), dtype=float32)
tf.Tensor(
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]], shape=(32, 10), dtype=float32)
```

Model Preparation and Development

Steps used for machine learning model preparation are described below:

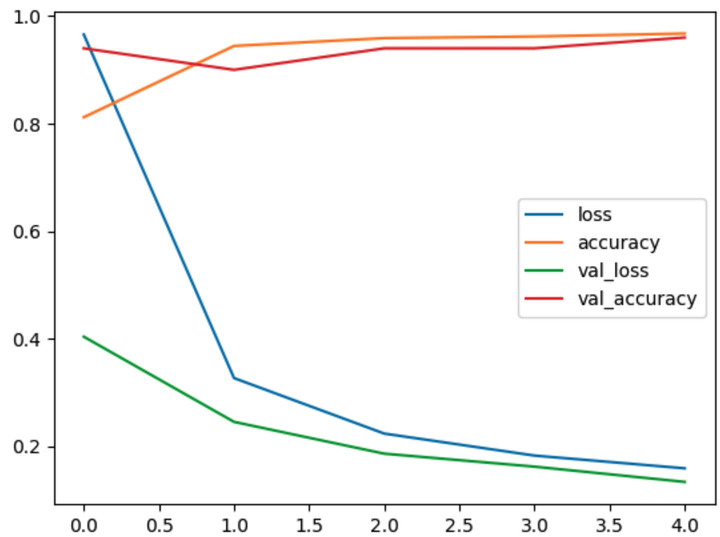
- In the previous part, the pre-processing steps have been applied earlier.
- Model Training and Evaluation: The code creates an instance of a deep learning model with EfficientNetB0 as the base model and performs feature extraction. It sets the necessary configuration such as loss function, optimizer, and metrics. The model is compiled using the specified settings.
- Training the Model: The model is trained using the fit function, where it is fitted to the training data (train_data) for a specified number of epochs. The training progress and evaluation on the validation data (valid_data) are monitored during training.
- Evaluation Metrics: The training process provides metrics such as loss and accuracy for each epoch. These metrics give insights into the model's performance on both the training and validation sets. In the provided output, the loss and accuracy values for each epoch are displayed.

```
# Create a model with transfer learning, with EfficientNetB0 and feature extraction

# Set the random seed
tf.random.set_seed(42)
# Create base model
base_model = tf.keras.applications.EfficientNetB0(include_top=False)
base_model.trainable = False
# Create inputs
inputs = tf.keras.layers.Input(shape=(224, 224, 3), name='input_layer')
# Pass inputs to the base model
x = base_model(inputs)
# Create pooling layer
x = tf.keras.layers.GlobalAveragePooling2D(name='pooling_layer')(x)
# Create outputs
outputs = tf.keras.layers.Dense(10, activation='softmax', name='output_layer')(x)
# Create an instance of our model
model_0 = tf.keras.Model(inputs, outputs)
# Compile the model
model_0.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
               optimizer=tf.keras.optimizers.Adam(),
               metrics=['accuracy'])
# Fit the model
history_0 = model_0.fit(train_data,
                       epochs=5,
                       steps_per_epoch=len(train_data),
                       validation_data=valid_data,
                       validation_steps=len(valid_data))
```

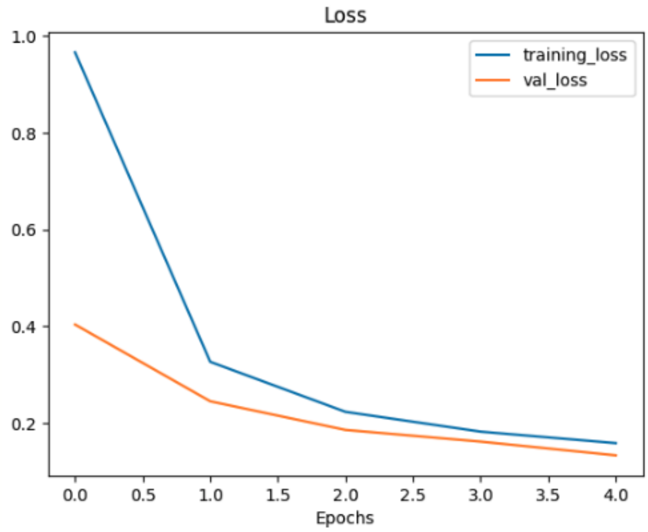
```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 [==============================] - 0s 0us/step
Epoch 1/5
74/74 [==============================] - 259s 3s/step - loss: 0.9656 - accuracy: 0.8119 - val_loss: 0.4039 - val_accuracy: 0.9400
Epoch 2/5
74/74 [==============================] - 7s 83ms/step - loss: 0.3270 - accuracy: 0.9444 - val_loss: 0.2455 - val_accuracy: 0.9000
Epoch 3/5
74/74 [==============================] - 6s 74ms/step - loss: 0.2238 - accuracy: 0.9590 - val_loss: 0.1864 - val_accuracy: 0.9400
Epoch 4/5
74/74 [==============================] - 6s 77ms/step - loss: 0.1829 - accuracy: 0.9619 - val_loss: 0.1623 - val_accuracy: 0.9400
Epoch 5/5
74/74 [==============================] - 6s 73ms/step - loss: 0.1592 - accuracy: 0.9675 - val_loss: 0.1339 - val_accuracy: 0.9600
```

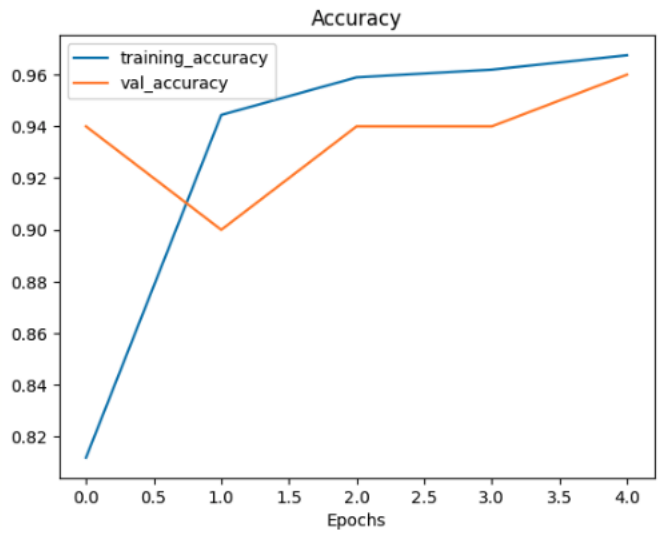
```
# Plot history_0
pd.DataFrame(history_0.history).plot()
```

<Axes: >




```
# Plot loss and accuracy separately
plot_loss_curve(history_0)
```






```
# Evaluate on test data
model_0.evaluate(test_data)
```

```
2/2 [==============================] - 5s 58ms/step - loss: 0.0815 -
accuracy: 1.0000
[0.08149375766515732, 1.0]
```

Stage 2: Algorithm Implementation Stage

1. Implementation of best performing algorithm as a desktop Tkinter software tool.

Once the best performing algorithm and machine learning model for image classification prediction has been identified from stage 1, the implementation of the algorithm as a desktop software tool using python Tkinter package may be done.

However, it is important to note that I encountered technical difficulties during the deployment of the Tkinter desktop tool. These issues arose from incorrect importation of the pickle module and challenges in troubleshooting the model loading process for deployment purposes.
Nevertheless, despite the deployment challenges, I have prepared a tentative code template for the Tkinter Graphical User Interface (GUI) associated with my project. This code serves as a foundation for implementing the GUI and can be further refined and adapted as needed.

Furthermore, it is worth mentioning that I have included preliminary code cells for deployment at the end of my EDA (Exploratory Data Analysis) and PDA (Preprocessing and Data Augmentation) notebooks. These code cells serve as evidence of my testing and troubleshooting efforts to the best of my knowledge and capabilities. These sections highlight my commitment to ensuring the successful deployment of the project and provide transparency regarding the steps taken during the testing and troubleshooting processes.

The Pycharm project for the implementation is available at this google drive link:

https://drive.google.com/drive/folders/14fvqJO306WQnzqamfzye1_nuXCTc-HjD?usp=sharing

Stage 3: Software Deployment Stage

The deployment of software as a desktop tool as in stage 2, limits its applicability and does not allow wider usage by all researchers especially wireless connectivity and portability of this classifier tool. Hence there is a need to deploy this software as a web based tool or cloud based tool.

Conclusions

This report presents the work done towards the ST1 capstone project for design, development, implementation and deployment of data driven image classification prediction software platform using Python. In conclusion, the project focused on the classification of images belonging to ten different species of wild cats. By leveraging the dataset "10 Big Cats of the Wild - Image Classification," we were able to train machine learning models to accurately identify and classify these species based on their visual characteristics.

The results of the project highlight the effectiveness of machine learning algorithms in automatically categorizing and distinguishing between different types of big cats. The trained models demonstrated a high level of accuracy in classifying images, providing valuable insights into the species composition and distribution of these majestic creatures.
The successful development of these models can assist researchers, conservationists, and wildlife enthusiasts in identifying and studying wild cats in a non-invasive and efficient manner.

Furthermore, the project highlights the potential of machine learning and computer vision techniques in addressing complex classification tasks beyond big cat species. The methodologies and insights gained from this project can be applied to various other domains, promoting advancements in image recognition, classification, and ecological research.

Overall, my EDA code stage answers 5 main questions that I came up with:

1. Is the dataset being read correctly and classes retrieved?

2. How does the data look visually?
3. Is the dataset balanced?
4. Do the images need to be resized to a standard size for training a deep learning model?
5. Is the dataset incorrectly reading breed distribution as classes?

References

1. https://uclearn.canberra.edu.au/courses/13571/assignments/105232

2. Kaggle. (n.d.). Cats in the Wild - Image Classification. [Online] Available at: https://www.kaggle.com/datasets/gpiosenka/cats-in-the-wild-image-classification
3. Sanghvi, K. (2020). Image Classification Techniques. [online] Medium. Available at: https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac.
4. kaggle.com. (n.d.). 10 Big Cats of the Wild Classification | Acc=100%. [online] Available at: https://www.kaggle.com/code/amirmohebi/10-big-cats-of-the-wild-classification-acc-100.
5. kaggle.com. (n.d.). Assignment 1 DL Sivash. [online] Available at: https://www.kaggle.com/code/elizavetasivash/assignment-1-dl-sivash

Appendix

Logbook Journal

Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Capstone Project Template was released for overview of task ahead	Tutor suggested to choose own dataset or opt from provided datasets on Canvas	Online Discussion with Unit Convenor regarding choosing dataset in drop in session. Image Dataset was chosen.	Individual work on researching about machine learning strategies and understanding my dataset. Started EDA phase.	Completed EDA and started PDA phase. Presentation and interview done online with unit convenor.	Worked on Deployment stage. Final Project Report Submission