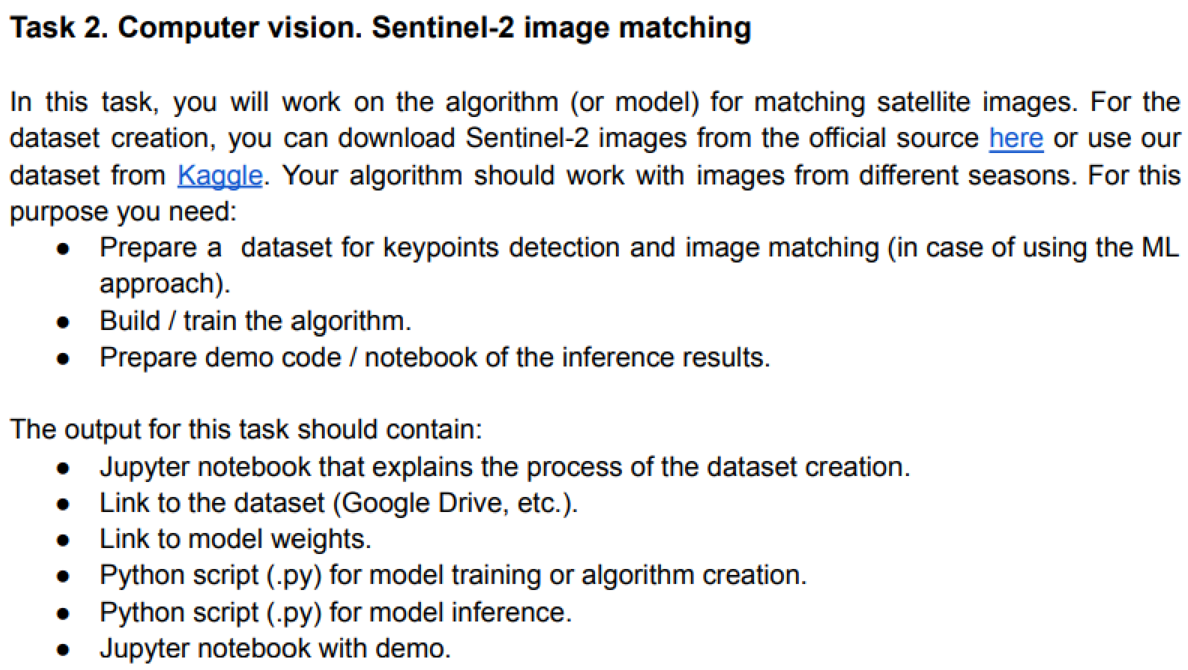


Task 2. Computer vision. Sentinel-2 image matching

In this task, you will work on the algorithm (or model) for matching satellite images. For the dataset creation, you can download Sentinel-2 images from the official source here or use our dataset from Kaggle. Your algorithm should work with images from different seasons. For this purpose you need:

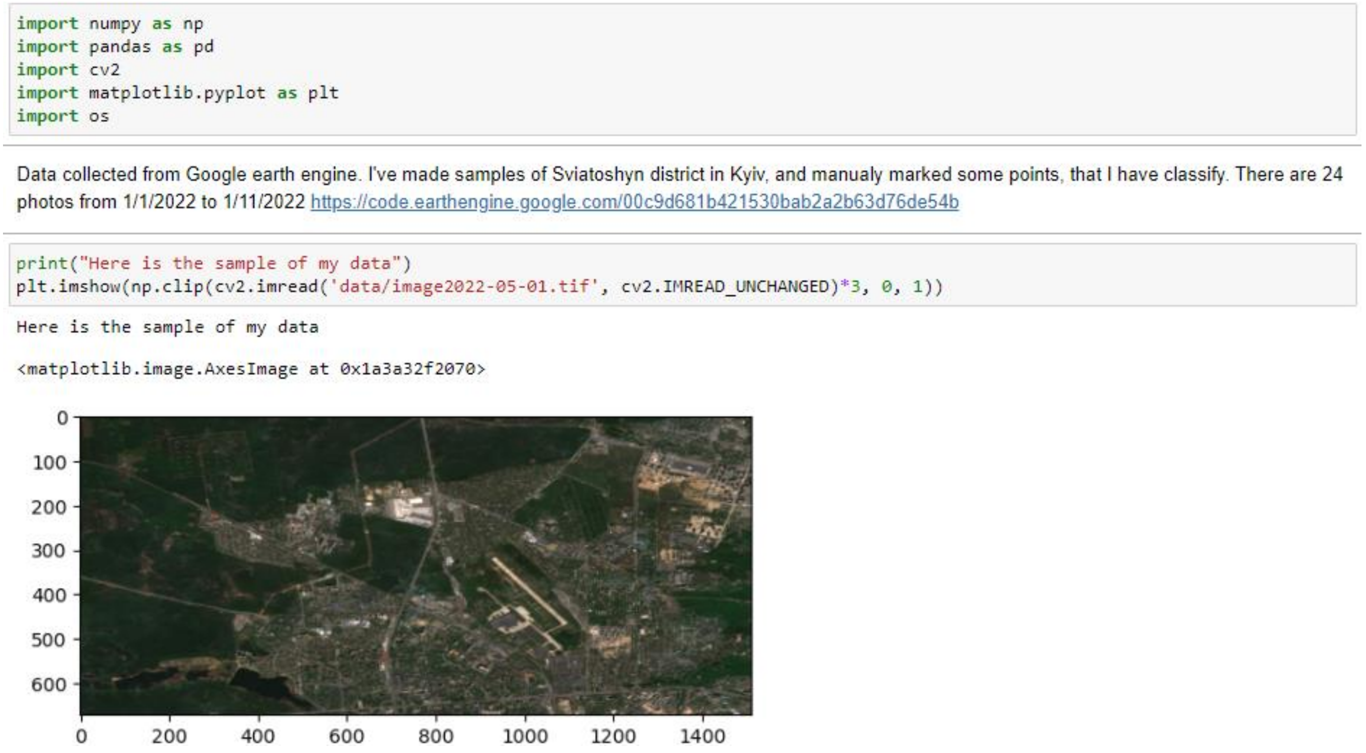
- Prepare a dataset for keypoints detection and image matching (in case of using the ML approach).
- Build / train the algorithm.
- Prepare demo code / notebook of the inference results.

The output for this task should contain:

- Jupyter notebook that explains the process of the dataset creation.
- Link to the dataset (Google Drive, etc.).
- Link to model weights.
- Python script (.py) for model training or algorithm creation.
- Python script (.py) for model inference.
- Jupyter notebook with demo.

In this task I collected by using Google earth engine. I've made samples of Sviatoshyn district in Kyiv, and manualy marked some points, that I have classify. There are 24 photos from 1/1/2022 to 1/11/2022

https://code.earthengine.google.com/00c9d681b421530bab2a2b63d76de54b



```
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import os
```

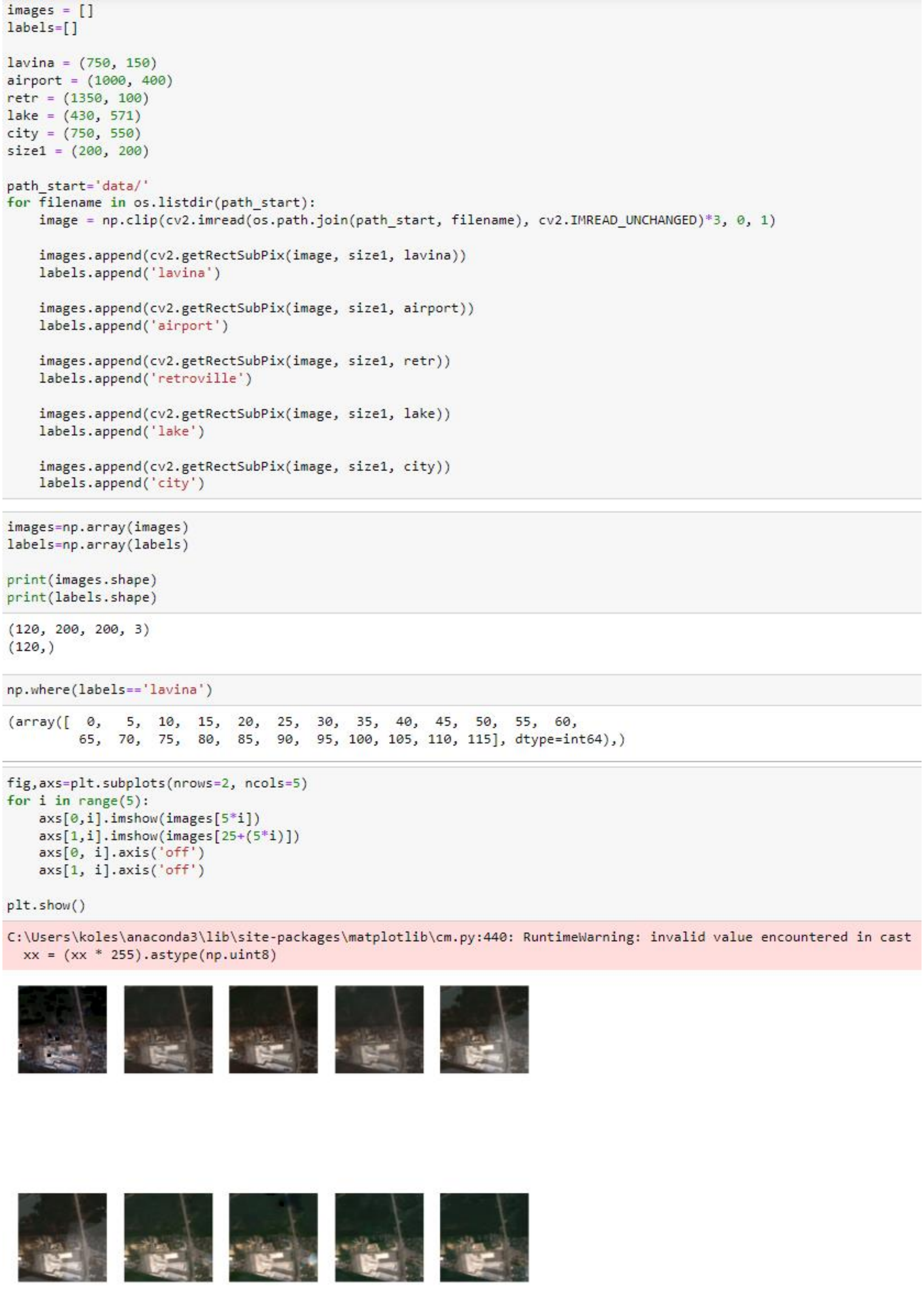
Data collected from Google earth engine. I've made samples of Sviatoshyn district in Kyiv, and manualy marked some points, that I have classify. There are 24 photos from 1/1/2022 to 1/11/2022 https://code.earthengine.google.com/00c9d681b421530bab2a2b63d76de54b

```
print("Here is the sample of my data")
plt.imshow(np.clip(cv2.imread('data/image2022-05-01.tif', cv2.IMREAD_UNCHANGED)*3, 0, 1))
```

```
Here is the sample of my data
```

```
<matplotlib.image.AxesImage at 0x1a3a32f2070>
```

So I created data samples as part of my big image. Parts is 200x200. I manually marked points for 5 classes, and from each photo I`m creating data sample for each class. I multiplied each pixels by 3 to increase brightness.



```
images = []
labels=[]

lavina = (750, 150)
airport = (1000, 400)
retr = (1350, 100)
lake = (430, 571)
city = (750, 550)
size1 = (200, 200)

path_start='data/'
for filename in os.listdir(path_start):
    image = np.clip(cv2.imread(os.path.join(path_start, filename), cv2.IMREAD_UNCHANGED)*3, 0, 1)

    images.append(cv2.getRectSubPix(image, size1, lavina))
    labels.append('lavina')

    images.append(cv2.getRectSubPix(image, size1, airport))
    labels.append('airport')

    images.append(cv2.getRectSubPix(image, size1, retr))
    labels.append('retroville')

    images.append(cv2.getRectSubPix(image, size1, lake))
    labels.append('lake')

    images.append(cv2.getRectSubPix(image, size1, city))
    labels.append('city')
```

```
images=np.array(images)
labels=np.array(labels)

print(images.shape)
print(labels.shape)
```

```
(120, 200, 200, 3)
(120,)
```

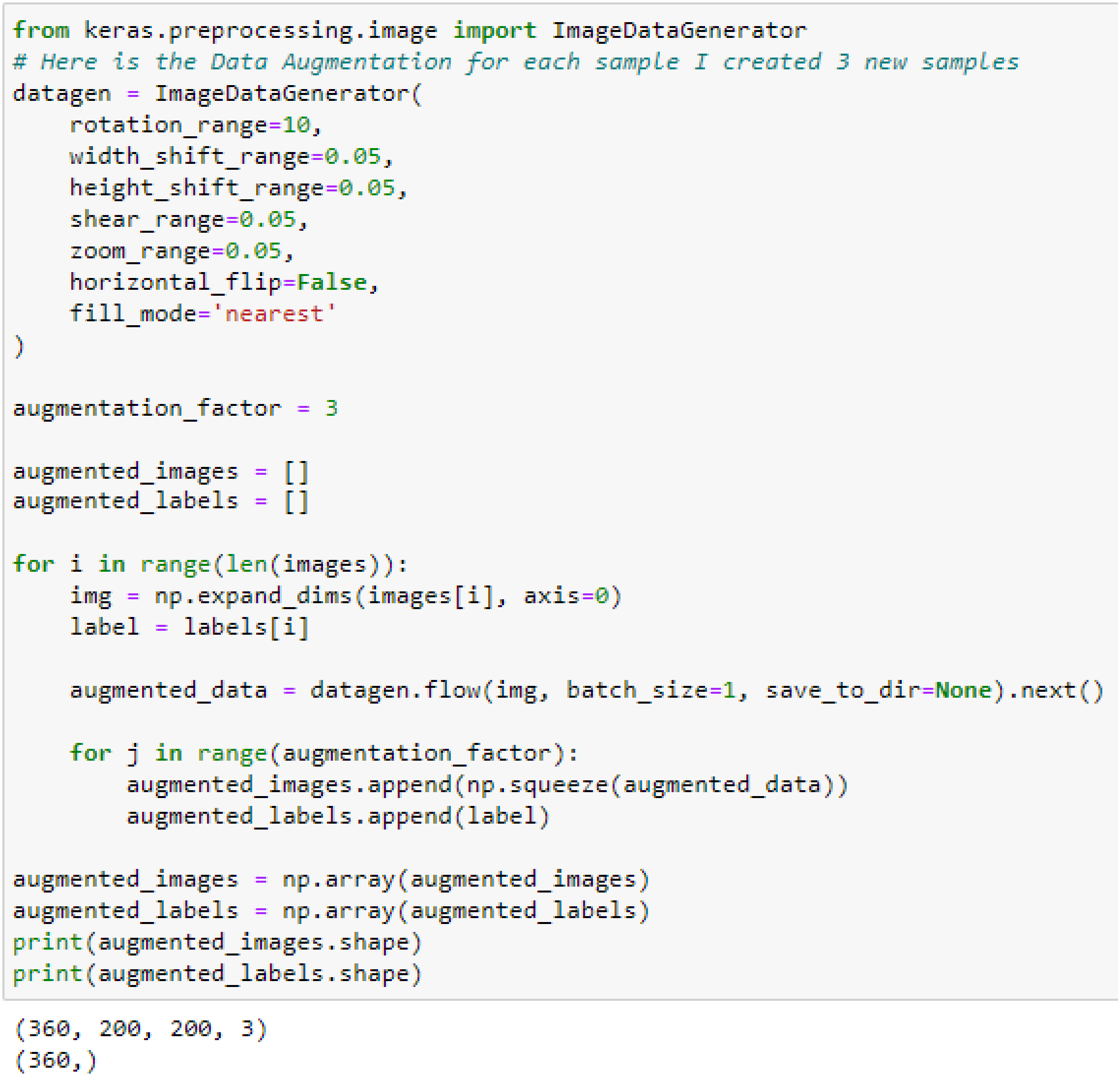
```
np.where(labels=='lavina')
```

```
(array([  0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60,
         65,  70,  75,  80,  85,  90,  95, 100, 105, 110, 115], dtype=int64),)
```

```
fig,axs=plt.subplots(nrows=2, ncols=5)
for i in range(5):
    axs[0,i].imshow(images[5*i])
    axs[1,i].imshow(images[25+(5*i)])
    axs[0, i].axis('off')
    axs[1, i].axis('off')

plt.show()
```

```
C:\Users\koles\anaconda3\lib\site-packages\matplotlib\cm.py:440: RuntimeWarning: invalid value encountered in cast
  xx = (xx * 255).astype(np.uint8)
```

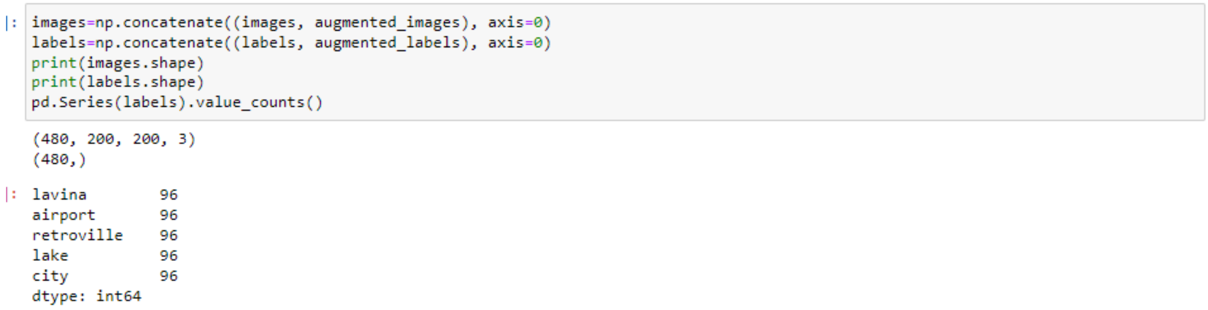
Data augmentation to increase number of samples.



```
from keras.preprocessing.image import ImageDataGenerator
# Here is the Data Augmentation for each sample I created 3 new samples
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.05,
    height_shift_range=0.05,
    shear_range=0.05,
    zoom_range=0.05,
    horizontal_flip=False,
    fill_mode='nearest'
)

augmentation_factor = 3

augmented_images = []
augmented_labels = []

for i in range(len(images)):
    img = np.expand_dims(images[i], axis=0)
    label = labels[i]

    augmented_data = datagen.flow(img, batch_size=1, save_to_dir=None).next()

    for j in range(augmentation_factor):
        augmented_images.append(np.squeeze(augmented_data))
        augmented_labels.append(label)

augmented_images = np.array(augmented_images)
augmented_labels = np.array(augmented_labels)
print(augmented_images.shape)
print(augmented_labels.shape)
```

```
(360, 200, 200, 3)
(360,)
```

As result I`ve got 480 samples(120 was before augmentation)

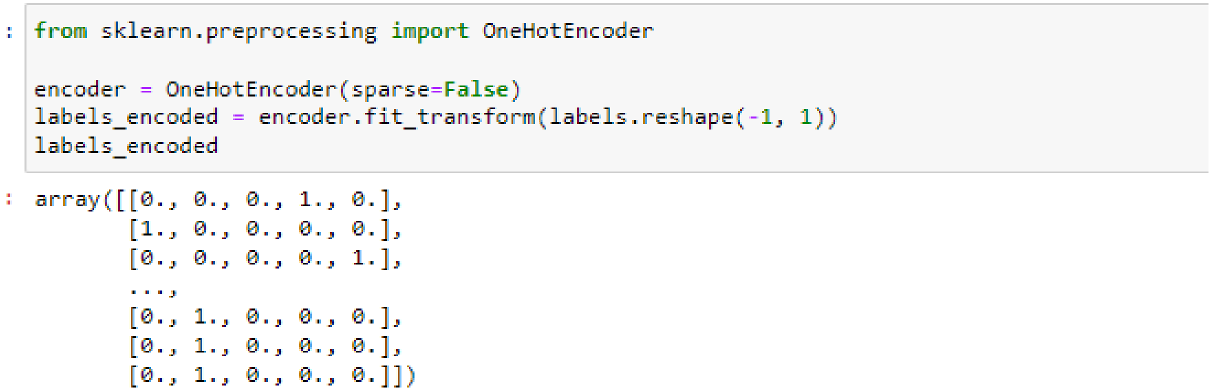


```
images=np.concatenate((images, augmented_images), axis=0)
labels=np.concatenate((labels, augmented_labels), axis=0)
print(images.shape)
print(labels.shape)
pd.Series(labels).value_counts()
```

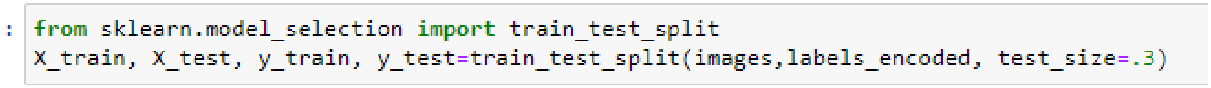
```
(480, 200, 200, 3)
(480,)
```

```
lavina         96
airport        96
retroville     96
lake           96
city           96
dtype: int64
```

Preparing data for model learning.

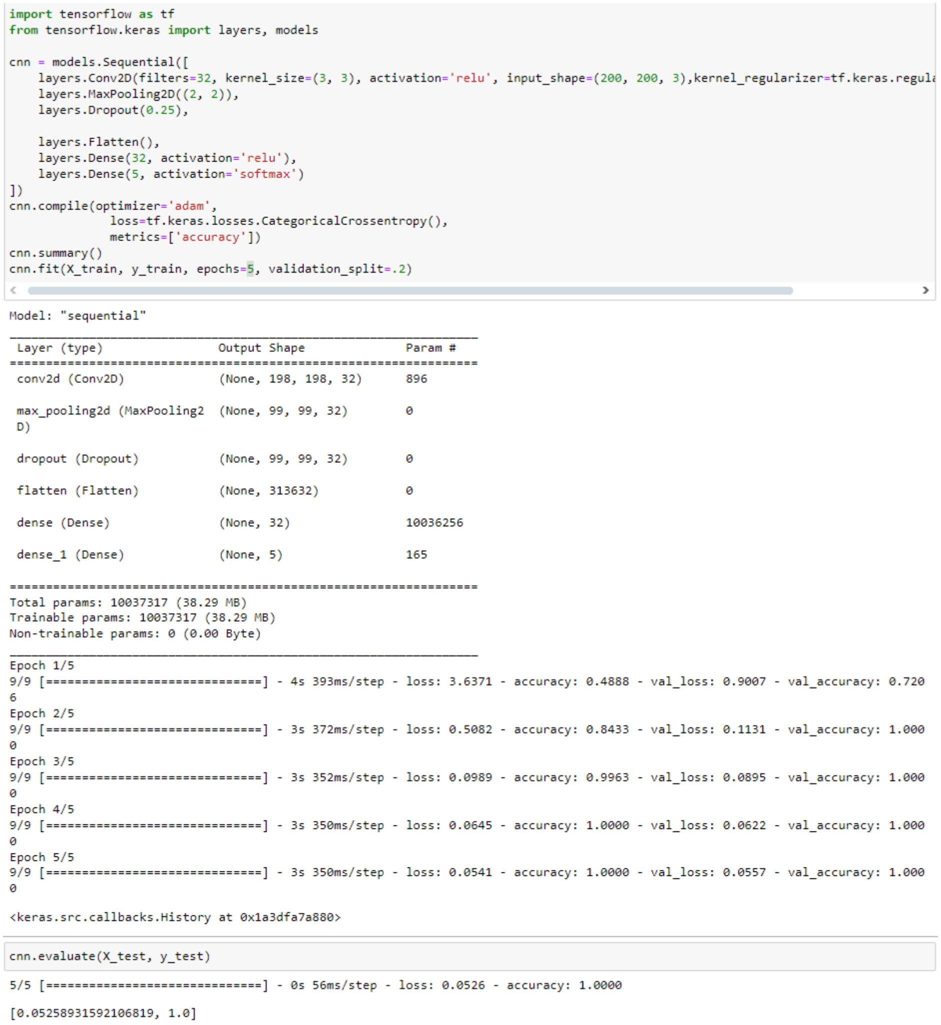


```
: from sklearn.preprocessing import OneHotEncoder

  encoder = OneHotEncoder(sparse=False)
  labels_encoded = encoder.fit_transform(labels.reshape(-1, 1))
  labels_encoded

: array([[0., 0., 0., 1., 0.],
         [1., 0., 0., 0., 0.],
         [0., 0., 0., 0., 1.],
         ...,
         [0., 1., 0., 0., 0.],
         [0., 1., 0., 0., 0.],
         [0., 1., 0., 0., 0.]])
```



```
: from sklearn.model_selection import train_test_split
  X_train, X_test, y_train, y_test=train_test_split(images,labels_encoded, test_size=.3)
```

I decided to build CNN model to classify images



```
import tensorflow as tf
from tensorflow.keras import layers, models

cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(200, 200, 3),kernel_regularizer=tf.keras.regula
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),

    layers.Flatten(),
    layers.Dense(32, activation='relu'),
    layers.Dense(5, activation='softmax')
])
cnn.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(),
              metrics=['accuracy'])
cnn.summary()
cnn.fit(X_train, y_train, epochs=5, validation_split=.2)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      896

 max_pooling2d (MaxPooling2  (None, 99, 99, 32)        0
 D)

 dropout (Dropout)           (None, 99, 99, 32)        0

 flatten (Flatten)           (None, 313632)            0

 dense (Dense)               (None, 32)                10036256

 dense_1 (Dense)             (None, 5)                 165

=================================================================
Total params: 10037317 (38.29 MB)
Trainable params: 10037317 (38.29 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/5
9/9 [==============================] - 4s 393ms/step - loss: 3.6371 - accuracy: 0.4888 - val_loss: 0.9007 - val_accuracy: 0.720
6
Epoch 2/5
9/9 [==============================] - 3s 372ms/step - loss: 0.5082 - accuracy: 0.8433 - val_loss: 0.1131 - val_accuracy: 1.000
0
Epoch 3/5
9/9 [==============================] - 3s 352ms/step - loss: 0.0989 - accuracy: 0.9963 - val_loss: 0.0895 - val_accuracy: 1.000
0
Epoch 4/5
9/9 [==============================] - 3s 350ms/step - loss: 0.0645 - accuracy: 1.0000 - val_loss: 0.0622 - val_accuracy: 1.000
0
Epoch 5/5
9/9 [==============================] - 3s 350ms/step - loss: 0.0541 - accuracy: 1.0000 - val_loss: 0.0557 - val_accuracy: 1.000
0

<keras.src.callbacks.History at 0x1a3dfa7a880>
```

```
cnn.evaluate(X_test, y_test)
```

```
5/5 [==============================] - 0s 56ms/step - loss: 0.0526 - accuracy: 1.0000

[0.05258931592106819, 1.0]
```

Accuracy 100