

```
In [ ]: ## Name : Patil Kiran Prakash
        ## Roll no:067
        ## Subject:LP-IV(DL)
```

```
In [1]: from tensorflow.keras.preprocessing.image import load_img
        from tensorflow.keras.preprocessing.image import img_to_array
        from keras.applications.vgg16 import preprocess_input
        from keras.applications.vgg16 import decode_predictions
        from keras.applications.vgg16 import VGG16
        # load an image from file
        image = load_img('download.jpg', target_size=(224, 224))
        # convert the image pixels to a numpy array
        image = img_to_array(image)
        # reshape data for the model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        # prepare the image for the VGG model
        image = preprocess_input(image)
        # load the model
        model = VGG16()
        # predict the probability across all output classes
        yhat = model.predict(image)
        # convert the probabilities to class labels
        label = decode_predictions(yhat)
        # retrieve the most likely result, e.g. highest probability
        label = label[0][0]
        # print the classification
        print('%s (%.2f%%)' % (label[1], label[2]*100))
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [=====] - 151s 0us/step
1/1 [=====] - 2s 2s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [=====] - 0s 0us/step
castle (34.03%)

```
In [2]: # load an image from file
        image = load_img('download2.png', target_size=(224, 224))
        # convert the image pixels to a numpy array
        image = img_to_array(image)
        # reshape data for the model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        # prepare the image for the VGG model
        image = preprocess_input(image)
        # load the model
        model = VGG16()
        # predict the probability across all output classes
        yhat = model.predict(image)
        # convert the probabilities to class labels
        label = decode_predictions(yhat)
        # retrieve the most likely result, e.g. highest probability
        label = label[0][0]
        # print the classification
        print('%s (%.2f%%)' % (label[1], label[2]*100))
```

1/1 [=====] - 1s 672ms/step
valley (44.85%)

```
In [3]: # load an image from file
        img('download3.jpg', target_size=(224, 224))
```

```
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load the model
model = VGG16()
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
1/1 [=====] - 2s 2s/step
golden_retriever (84.78%)
```

In []: