PROJECT IMPLEMENTATION

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

```
Image_SIZE = [180,180]
```

```
! pip install tensorflow
```

```
import keras.applications
```

```
from tensorflow.keras.applications import resnet
```

```
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import ImageFile
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2,
zoom_range =[.99, 1.01], brightness_range = [0.8,
1.2],data_format="channels_last",fill_mode="constant",horizontal_flip =
True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import ImageFile
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range= 0.2,
zoom_range = [.99, 1.01], brightness_range= [0.8, 1.2], data_format=
"channels_last", fill_mode="constant", horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/archive
(6)/dataset', target_size = (180, 180), batch_size = 64, class_mode =
'categorical')
test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/archive
(6)/dataset',target_size = (180, 180), batch_size = 64, class_mode =
'categorical')
```

```
VGG19 = VGG19(input_shape = Image_SIZE +[3],weights =
'imagenet',include_top = False )
```

```
for layer in VGG19.layers:
  layer.trainable = False
x = Flatten()(VGG19.output)
prediction = Dense(6, activation = 'softmax')(x)
model = Model(inputs = VGG19.input, outputs = prediction)
```

```
model.compile(optimizer="sgd",
              loss="categorical_crossentropy",
              metrics=[keras.metrics.Precision(), keras.metrics.Recall(),
keras.metrics.SpecificityAtSensitivity(0.5),
keras.metrics.SensitivityAtSpecificity(0.5), 'accuracy'])
```

```
r = model.fit(
    training_set,
    validation_data=test_set,
    epochs=5,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)
```

```
loss,accuracy = model.evaluate(test_set,
                               steps = 11,
                               verbose = 2,
                               use_multiprocessing = True,
                               workers = 2)
```

```
printf('Model performance on test images:\nAccuracy = {accuracy}\nLoss =
{loss}')
```

```
model.save('WCV.h5')
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
model = load_model('/content/WCV.h5')
```

```
import numpy as np
import os
from flask import Flask,request,render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_inpu
```

```
model = load_model(r"WCV.h5")
app = Flask(__name__)
```

```
@app.route('/')
def index():
  return render_template('index.html')

@app.route('/home')
def home():
  return render_template('index.html')

@app.route('/input')
def input1():
  return render_template('index.html')
```

```
@app.route('/predict', methods = ["GET","POST"])
def res():
  if request.method == "POST":
    f = request.files['image']
    basepath = os.path.dirname(__file__)
    filepath = os.path.join(basepath,'uploads',filename)
    f.save(filepath)

    img = image.load_img(filepath,target_size = (224,224,3))
    x = image.img_to_array(img)
```

```
    x = np.expand_dims(x,axis = 0)

    img_data = preprocess_input(x)
    prediction = np.argmax(model.predict(img_data),axis = 1)


    index = ['alien_test','cloudy','foggy','rainy','sunrise']

    result = str(index[prediction[0]])
    print(result)
    return render_template('output.html',prediction = result)
```

```
if __name__ == "__main__":
  app.run(debug = False)
```

https://colab.research.google.com/drive/1dtBXBDsiRozPhJMDmQfUljUiK9K5uoWZ?usp=sharing