Image Classification Using Scikit-learn

Agenda

01 Importing the Libraries

02 Loading the Data

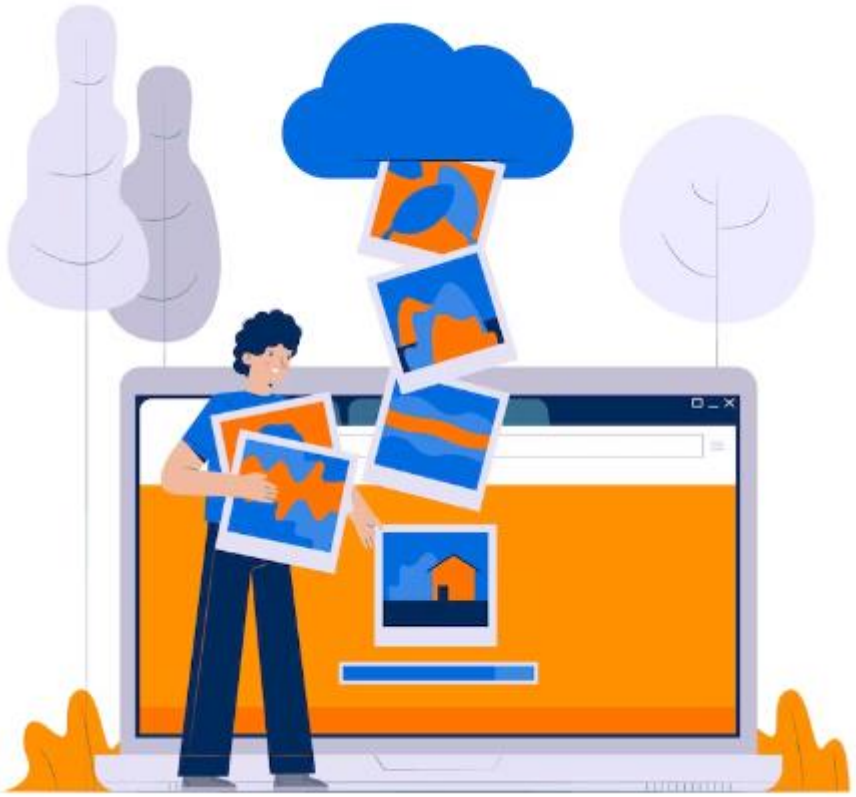
03 Data Preprocessing

04 Implementing ML Algorithms

05 Hyperparameter Tuning

Problem Statement

You are a data-scientist of a global company. As a data-scientist you have to build an image classification model to classify 2 categories of images namely cat and dogs.



Dataset Information

Cats and Dogs dataset contains 1000 images out of which 500 images are of cat and same for the dog.

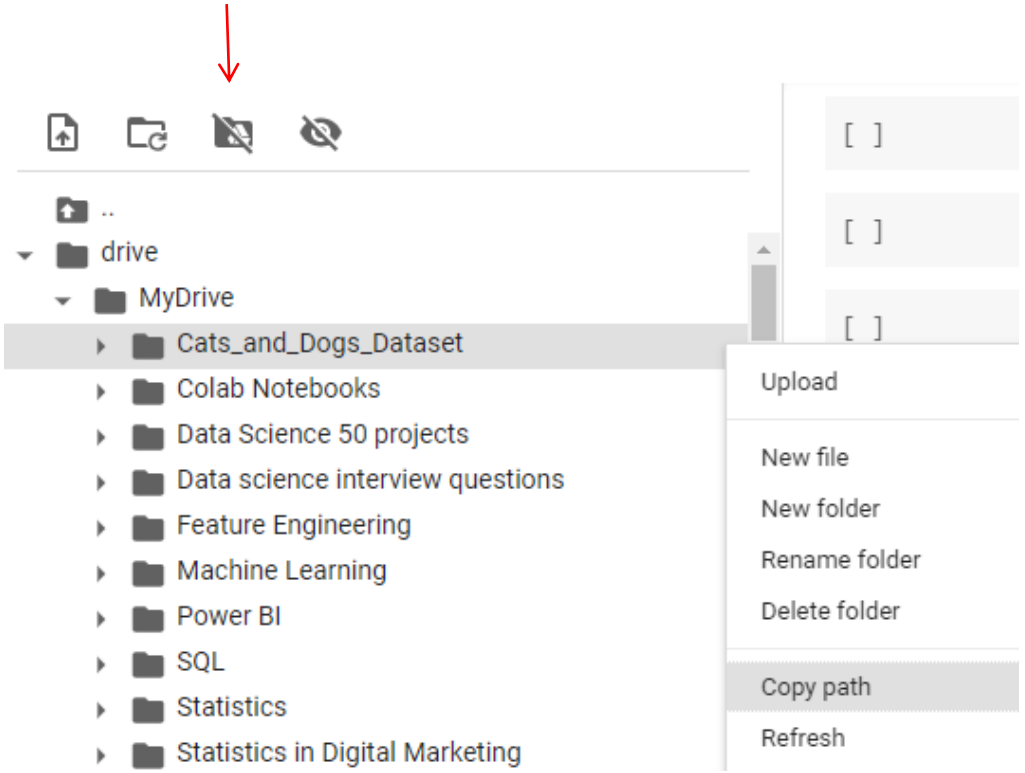
Importing the Libraries

We start off this project by importing all the necessary libraries that will be required for the process.

```
import os
from skimage.io import imread
from skimage.transform import resize
from PIL import Image
import numpy as np
```

Loading the Image Data

Loading the image data by mounting the drive and providing the path of a folder stored in drive.




```
input_dir= "/content/drive/MyDrive/Cats_and_Dogs_Dataset"
```

Displaying the Image

```
categories=["Cat500","Dog500"]
for category in categories:
  for file in os.listdir(os.path.join(input_dir,category)):
    img_path=os.path.join(input_dir,category,file)
    print(img_path)
```

```
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.101.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.102.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.117.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.112.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.104.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.11.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.106.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.111.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.118.jpg
/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.10.jpg
```

To display an image we need a path of each image. Here we are using two functions to handle path .

os.listdir - print a list of names of all the files present in the specified path.

os.path.join - return a combined path by merging arguments.

Displaying the Image

Taking any random path of the previous output to display the image.

```
Image.open("/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.21.jpg")
```

Image conversion to array

Converting the image to numpy array

```
img_path="/content/drive/MyDrive/Cats_and_Dogs_Dataset/Cat500/cat.21.jpg"
```

```
# Converting an image to numpy array
img=imread(img_path)
print(img)
```

```
[[[187 147  95]
  [180 143  91]
  [176 138  89]
  ...
  [239 232 164]
  [246 246 176]
  [253 255 187]]

 [[185 145  93]
  [181 144  92]
  [180 142  93]
  ...
  [239 231 166]
  [245 244 177]
  [252 255 186]]

 [[185 145  93]
  [182 145  92]
  [183 146  94]
  ...
  [243 235 170]
  [247 244 177]
  [250 251 183]]

 ...
```

Image conversion to array

Resizing and flattening the numpy array.

```
img=resize(img,(15,15))
img
```

```
img=img.flatten()
```

```
array([0.74509804, 0.60732026, 0.39947712, 0.76057516, 0.61568627,
       0.41176471, 0.78954248, 0.65228758, 0.47581699, 0.83529412,
       0.68235294, 0.49803922, 0.84705882, 0.69071895, 0.51372549,
       0.86063617, 0.70588235, 0.52156863, 0.88627451, 0.72941176,
       0.54079303, 0.9254902 , 0.78039216, 0.6       , 0.95686275,
       0.83386492, 0.62352941, 0.96470588, 0.88183007, 0.64313725,
       0.96470588, 0.91722876, 0.69479739, 0.96418301, 0.90539434,
       0.67424837, 0.92679739, 0.8130719 , 0.56078431, 0.62138562,
       0.49364706, 0.35628758, 0.65209586, 0.5275817 , 0.36137691,
       0.73694118, 0.58823529, 0.37254902, 0.76470588, 0.61960784,
       0.42054902, 0.77777778, 0.63816993, 0.45385621, 0.84313725,
       0.69019608, 0.50588235, 0.87843137, 0.7254902 , 0.54509804,
       0.8745098 , 0.71764706, 0.52941176, 0.87947712, 0.72156863,
       0.53333333, 0.89411765, 0.77490196, 0.62901961, 0.94509804,
       0.82703268, 0.63633987, 0.94562092, 0.84156863, 0.62352941,
       0.85537255, 0.76470588, 0.56235294, 0.79529412, 0.6896732 ,
       0.49098039, 0.81986928, 0.72235294, 0.51555556, 0.67764706,
       0.56392157, 0.42431373, 0.58122876, 0.45045752, 0.29720261,
       0.76078431, 0.6151634 , 0.40784314, 0.79215686, 0.64313725,
       0.45490196, 0.78039216, 0.6335512 , 0.44705882, 0.82501089,
       0.67058824, 0.47067538, 0.85437908, 0.69411765, 0.50535948,
       0.8745098 , 0.70588235, 0.51372549, 0.85028322, 0.69899782,
       0.53333333, 0.79215686, 0.71503268, 0.65490196, 0.84836601,
       0.74640523, 0.62849673, 0.69132898, 0.59572985, 0.46431373,
       0.48575163, 0.4227451 , 0.35712418, 0.59084967, 0.49429194,
```

Image conversion to array

```
# Creating two empty list as data and labels
data=[]
labels=[]
```

```
for category_idx, category in enumerate(categories):
  for file in os.listdir(os.path.join(input_dir,category)):
    img_path=os.path.join(input_dir,category,file)
    img=imread(img_path)  # img is a numpy array
    img=resize(img,(15,15))
    data.append(img.flatten())
    labels.append(category_idx)
```

```
labels=np.asarray(labels)
data= np.asarray(data)
```

Converting all the images into a 1-D array and storing it into data variable. Here Label contains two categories i.e 0 (Cat) and 1 (Dog)

Logistic Regression

Accuracy obtained by logistic regression is 54 percent.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data,labels,test_size=0.2,shuffle=True, stratify=labels)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
accuracy_score(y_test,y_pred)
```

```
0.545
```

Decision Tree

Accuracy obtained by decision tree model is 56 percent.

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
accuracy_score(y_test,y_pred)
```

```
0.56
```

Random Forest

Accuracy obtained by random forest model is 64.5 percent.

```
from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier(n_estimators=1000)
rfc.fit(x_train,y_train)
y_pred=rfc.predict(x_test)
accuracy_score(y_test,y_pred)
```

```
0.645
```

Hyperparameter Tuning - Random Forest

Accuracy obtained by random forest model is 65.5 percent.

```
rfc=RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(x_train, y_train)
CV_rfc.best_params_
rfc1=RandomForestClassifier(random_state=42, max_features='auto', n_estimators= 200, max_depth=8, criterion='gini')
rfc1.fit(x_train, y_train)
pred=rfc1.predict(x_test)
print("Accuracy for Random Forest on data: ",accuracy_score(y_test,pred))
```

```
Accuracy for Random Forest on data:  0.655
```

Hyperparameter Tuning - Random Forest

Creating the dataframe for actual and predicted value.

```
pd.DataFrame({"Actual_Value":y_test,"Predicted_Value":y_pred})
```

	Actual_Value	Predicted_Value
0	0	0
1	0	0
2	1	1
3	0	0
4	0	1
...	...	...
195	1	1
196	1	1
197	0	0
198	1	1
199	0	0

200 rows × 2 columns

Note - We have trained 1000 images of cats and dogs. Increase in sample image will lead to increase in accuracy.