Dataset Information

The training archive contains 25,000 images of dogs and cats. Train your algorithm on these files and predict the labels

(1 = dog, 0 = cat).

Download Dataset

```
!wget https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_5340.zip
```

```
--2024-07-18 14:58:03--  https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_5340.zip
Resolving download.microsoft.com (download.microsoft.com)... 23.200.181.207, 2600:1417:3f:1388::317f, 2600:1417:3f:138a::317f
Connecting to download.microsoft.com (download.microsoft.com)|23.200.181.207|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 824887076 (787M) [application/octet-stream]
Saving to: ‘kagglecatsanddogs_5340.zip’

kagglecatsanddogs_5 100%[===================>] 786.67M   279MB/s    in 2.8s

2024-07-18 14:58:06 (279 MB/s) - ‘kagglecatsanddogs_5340.zip’ saved [824887076/824887076]
```

Unzip the Dataset

```
!unzip kagglecatsanddogs_5340.zip
```

```
inflating: PetImages/Dog/9102.jpg
inflating: PetImages/Dog/9103.jpg
inflating: PetImages/Dog/9104.jpg
inflating: PetImages/Dog/9105.jpg
inflating: PetImages/Dog/9106.jpg
inflating: PetImages/Dog/9107.jpg
inflating: PetImages/Dog/9108.jpg
inflating: PetImages/Dog/9109.jpg
inflating: PetImages/Dog/911.jpg
inflating: PetImages/Dog/9110.jpg
```

Import Modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import os
import tqdm
import random
from keras.preprocessing.image import load_img
warnings.filterwarnings('ignore')
```

Create Dataframe for Input and Output

```
input_path = []
label = []

for class_name in os.listdir("PetImages"):
    for path in os.listdir("PetImages/"+class_name):
        if class_name == 'Cat':
            label.append(0)
        else:
            label.append(1)
        input_path.append(os.path.join("PetImages", class_name, path))
print(input_path[0], label[0])
```

```
PetImages/Dog/12332.jpg 1
```

```
df = pd.DataFrame()
df['images'] = input_path
df['label'] = label
df = df.sample(frac=1).reset_index(drop=True)
df.head()
```

	images	label
0	PetImages/Cat/7516.jpg	0
1	PetImages/Cat/11045.jpg	0
2	PetImages/Cat/11051.jpg	0
3	PetImages/Dog/11354.jpg	1
4	PetImages/Dog/8025.jpg	1

Next steps: Generate code with df View recommended plots

```
for i in df['images']:
    if '.jpg' not in i:
        print(i)
```

```
PetImages/Dog/Thumbs.db
PetImages/Cat/Thumbs.db
```

```
import PIL
l = []
for image in df['images']:
    try:
        img = PIL.Image.open(image)
    except:
        l.append(image)
l
```

```
['PetImages/Dog/Thumbs.db',
 'PetImages/Dog/11702.jpg',
 'PetImages/Cat/Thumbs.db',
 'PetImages/Cat/666.jpg']
```

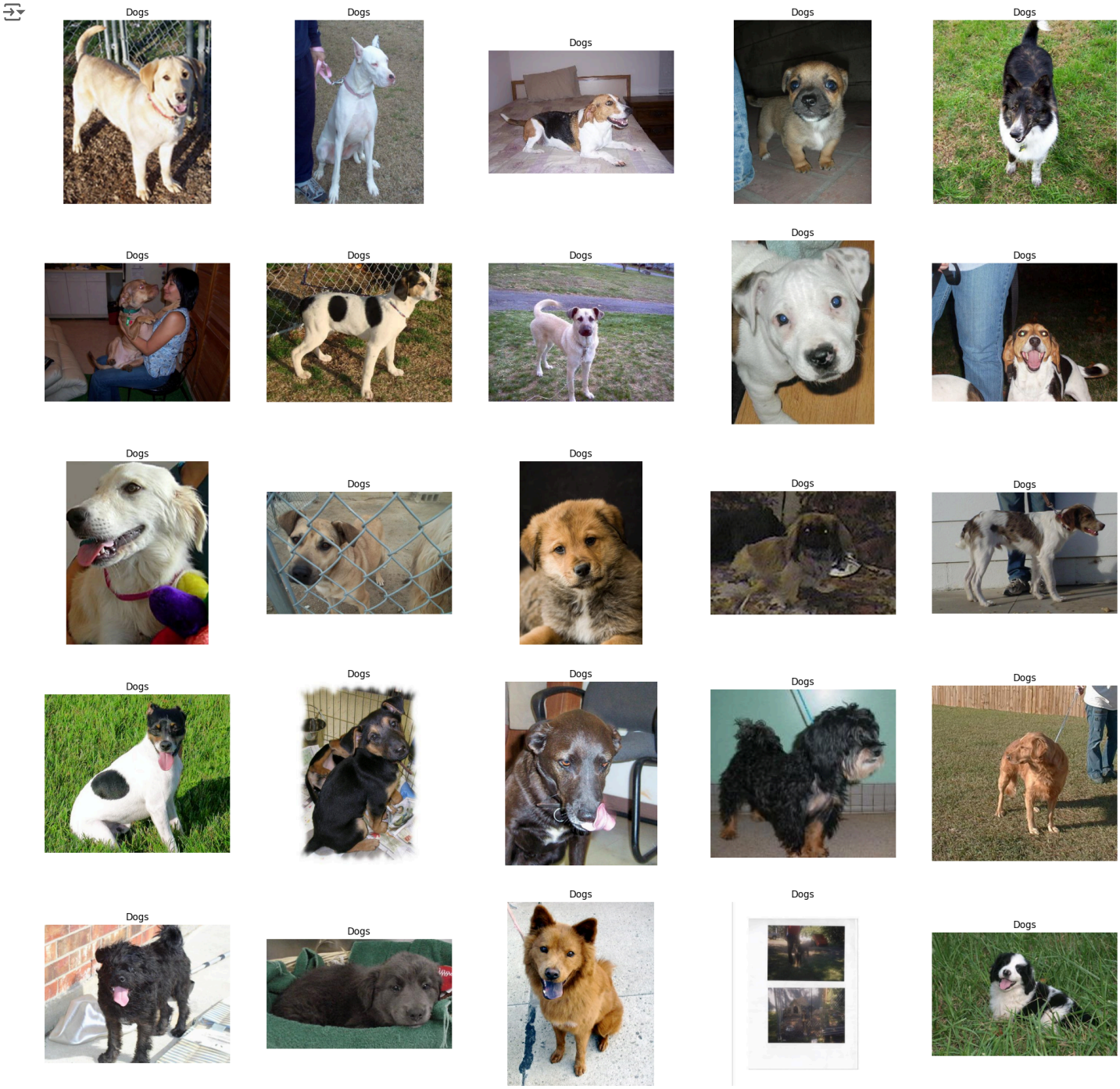
```
# delete db files
df = df[df['images']!='PetImages/Dog/Thumbs.db']
df = df[df['images']!='PetImages/Cat/Thumbs.db']
df = df[df['images']!='PetImages/Cat/666.jpg']
df = df[df['images']!='PetImages/Dog/11702.jpg']
len(df)
```

```
24998
```

Exploratory Data Analysis

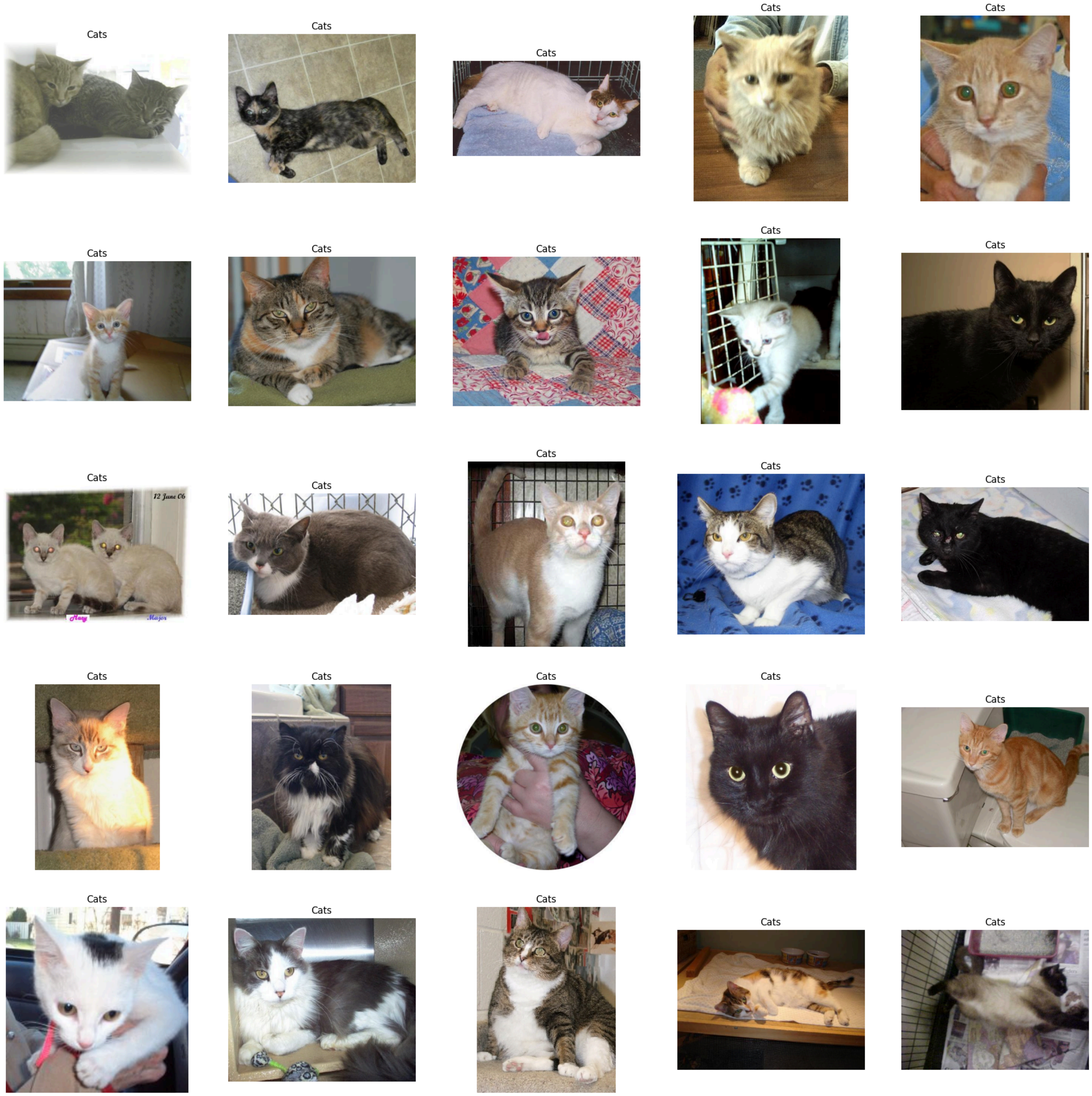
```
# to display grid of images
plt.figure(figsize=(25,25))
temp = df[df['label']==1]['images']
start = random.randint(0, len(temp))
files = temp[start:start+25]

for index, file in enumerate(files):
    plt.subplot(5,5, index+1)
    img = load_img(file)
    img = np.array(img)
    plt.imshow(img)
    plt.title('Dogs')
    plt.axis('off')
```


Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs
Dogs

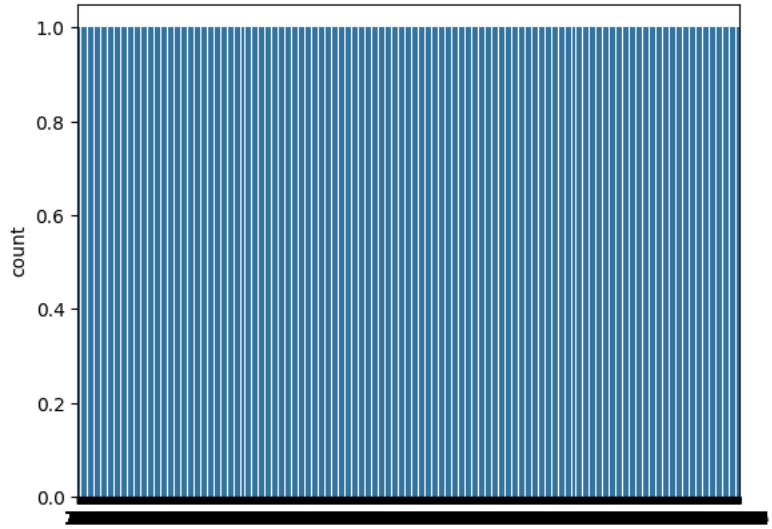
```
# to display grid of images
plt.figure(figsize=(25,25))
temp = df[df['label']==0]['images']
start = random.randint(0, len(temp))
files = temp[start:start+25]

for index, file in enumerate(files):
    plt.subplot(5,5, index+1)
    img = load_img(file)
    img = np.array(img)
    plt.imshow(img)
    plt.title('Cats')
    plt.axis('off')
```


Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
12 June 06
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats
Cats

```
import seaborn as sns
sns.countplot(df['label'])
```

```
<Axes: ylabel='count'>
```



Create DataGenerator for the Images

```
df['label'] = df['label'].astype('str')
```

```
df.head()
```

	images	label
0	PetImages/Cat/7516.jpg	0
1	PetImages/Cat/11045.jpg	0
2	PetImages/Cat/11051.jpg	0
3	PetImages/Dog/11354.jpg	1
4	PetImages/Dog/8025.jpg	1

Next steps: Generate code with df View recommended plots

```
# input split
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.2, random_state=42)


from keras.preprocessing.image import ImageDataGenerator
train_generator = ImageDataGenerator(
    rescale = 1./255,  # normalization of images
    rotation_range = 40, # augmention of images to avoid overfitting
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    fill_mode = 'nearest'
)

val_generator = ImageDataGenerator(rescale = 1./255)

train_iterator = train_generator.flow_from_dataframe(
    train,
    x_col='images',
    y_col='label',
    target_size=(128,128),
    batch_size=512,
    class_mode='binary'
)

val_iterator = val_generator.flow_from_dataframe(
    test,
    x_col='images',
    y_col='label',
    target_size=(128,128),
    batch_size=512,
    class_mode='binary'
)
```

```
Found 19998 validated image filenames belonging to 2 classes.
Found 5000 validated image filenames belonging to 2 classes.
```

Model Creation

```
from keras import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense

model = Sequential([
                    Conv2D(16, (3,3), activation='relu', input_shape=(128,128,3)),
                    MaxPool2D((2,2)),
                    Conv2D(32, (3,3), activation='relu'),
                    MaxPool2D((2,2)),
                    Conv2D(64, (3,3), activation='relu'),
                    MaxPool2D((2,2)),
                    Flatten(),
                    Dense(512, activation='relu'),
                    Dense(1, activation='sigmoid')
])


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448

 max_pooling2d (MaxPooling2  (None, 63, 63, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 64)        18496
```