+ Code + Text

[1] 3s
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

[2] 5s
```
#Load the Libraries
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

[3] 0s
```
# Load and preprocess the image data
data = ImageDataGenerator(
    rescale = 1.0/255.0,
    validation_split = 0.2
)
```

[4] 0s
```
df1 = "/content/drive/MyDrive/Cats_and_Dogs_dataset"
```

```
from skimage.io import imread
from PIL import Image

categories=["cats", "dogs"]
for category in categories:
  for file in os.listdir (os.path.join(df1, category)):
    img_path=os.path.join(df1, category,file)
    print(img_path)
```

```
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.123.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.160.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.17.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.182.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.164.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.200.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.122.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.112.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.127.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.11.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.134.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.148.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.114.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.135.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.174.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.116.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.13.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.117.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.145.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.130.jpg
/content/drive/MyDrive/Cats_and_Dogs_dataset/dogs/dog.104.jpg
```

```
[18] #Taking any random path of the previous output to display the image
     Image.open("/content/drive/MyDrive/Cats_and_Dogs_dataset/cats/cat.21.jpg")
```

✓ 1s completed at 6:01 PM

[5] 1s

```
# Extracting a batch of data from generator object
x_batch, y_batch = next(data.flow_from_directory(
    df1,
    target_size=(224,224),
    batch_size=50,
    class_mode='categorical',
    subset='training'
))
```

```
Found 1600 images belonging to 2 classes.
```

[6] ✓ 2s

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Splitting data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x_batch.reshape(x_batch.shape[0], -1), y_batch.argmax(axis=1), test_size=0.2, random_state=4

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

lr = LogisticRegression()
lr.fit(x_train, y_train)

y_pred = lr.predict(x_test)
accuracy_score(y_test, y_pred)
```

```
0.6
```

[10]

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

y_pred = dt.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("The accuracy of the decision tree classifier is:",accuracy)
```

```
The accuracy of the decision tree classifier is: 0.4
```

[11] 3s

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

rfc = RandomForestClassifier(n_estimators=1000)
rfc.fit(x_train, y_train)

y_pred = rfc.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("The accuracy of the random forest classifier is :",accuracy)
```

```
The accuracy of the random forest classifier is : 0.4
```

[12]

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

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)
print(f"The best parameters for the random forest classifier are {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)
accuracy = accuracy_score(y_test, pred)
print("The accuracy of the random forest classifier is :",accuracy)
```

```
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and
  warn(
```

[12]
```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in
  warn(
The best parameters for the random forest classifier are {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'n_estimators'
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in
  warn(
The accuracy of the random forest classifier is : 0.3
```

[14]
```
#Creating the dataframe for actual and predicted value.

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

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

```
   Actual_Value  Predicted_Value
0             1                1
1             1                0
2             1                1
3             0                1
4             1                0
5             0                0
6             0                1
7             1                0
8             0                0
9             0                1
```