In [21]:
```
import os
import cv2
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import F1Score,Precision,Recall
```

In [7]:
```
data_dir='/Users/KIIT/Downloads/CatsDogs/train'

folds=os.listdir(data_dir)
file_paths=[]
labels=[]

for folds in folds:
  foldpath=os.path.join(data_dir,folds)
  files=os.listdir(foldpath)
  for files in files:
    file_path=os.path.join(foldpath,files)
    file_paths.append(file_path)
    labels.append(folds)

df_train=pd.DataFrame(data={'file_path':file_paths,'label':labels})
```

In [9]:
```
df_train.head()
```

Out[9]:

	file_path	label
0	/Users/KIIT/Downloads/CatsDogs/train\cats\cat_...	cats
1	/Users/KIIT/Downloads/CatsDogs/train\cats\cat_...	cats
2	/Users/KIIT/Downloads/CatsDogs/train\cats\cat_...	cats
3	/Users/KIIT/Downloads/CatsDogs/train\cats\cat_...	cats
4	/Users/KIIT/Downloads/CatsDogs/train\cats\cat_...	cats

In [11]:
```
data_dir='/Users/KIIT/Downloads/CatsDogs/test'

folds=os.listdir(data_dir)
file_paths=[]
labels=[]

for folds in folds:
  foldpath=os.path.join(data_dir,folds)
  files=os.listdir(foldpath)
  for files in files:
    file_path=os.path.join(foldpath,files)
    file_paths.append(file_path)
    labels.append(folds)

df_test=pd.DataFrame(data={'file_path':file_paths,'label':labels})
```

In [15]:
```
df_test.tail()
```

Out[15]:

	file_path	label
137	/Users/KIIT/Downloads/CatsDogs/test\dogs\dog_6...	dogs
138	/Users/KIIT/Downloads/CatsDogs/test\dogs\dog_7...	dogs
139	/Users/KIIT/Downloads/CatsDogs/test\dogs\dog_8...	dogs
140	/Users/KIIT/Downloads/CatsDogs/test__MACOSX\cats	__MACOSX
141	/Users/KIIT/Downloads/CatsDogs/test__MACOSX\dogs	__MACOSX

In [17]:
```
# Image preprocessing function
def preprocess_image(image_path, size=(64, 64)):
    image = cv2.imread(image_path, cv2.IMREAD_COLOR)  # Read image
    if image is None:
        return np.zeros(size[0] * size[1] * 3)  # Handle missing images
    image = cv2.resize(image, size)  # Resize to 64x64
    return image.flatten()  # Flatten to 1D
```

In [23]:
```
# Apply preprocessing
X = np.array([preprocess_image(path) for path in df_train["file_path"]])
y = df_train["label"].apply(lambda x: 1 if x == "dogs" else 0).values  # Encode labels
```

In [25]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [29]:
```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Train SVM
svm_model = SVC(kernel="linear")
svm_model.fit(X_train, y_train)

# Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)

# Train Logistic Regression
lr_model = LogisticRegression(max_iter=5000)
lr_model.fit(X_train, y_train)

# Evaluate models
print("SVM Accuracy:", accuracy_score(y_test, svm_model.predict(X_test)))
print("Random Forest Accuracy:", accuracy_score(y_test, rf_model.predict(X_test)))
print("Logistic Regression Accuracy:", accuracy_score(y_test, lr_model.predict(X_test)))
```

```
SVM Accuracy: 0.6071428571428571
Random Forest Accuracy: 0.5267857142857143
Logistic Regression Accuracy: 0.6071428571428571
```

In [31]:
```
import joblib

joblib.dump(svm_model, "svm_model.pkl")
joblib.dump(rf_model, "rf_model.pkl")
joblib.dump(lr_model, "lr_model.pkl")
```

Out[31]:
```
['lr_model.pkl']
```

In [ ]: