Implementing a Support Vector Machine(SVM) to classify images of cats and dogs from the kaggle data.

In [1]:

```
#Import Necessary Libraries
import cv2
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

In [2]:

```
# Set directory paths
cat_dir = 'C:/Users/biswa/Downloads/train/cats'
dog_dir = 'C:/Users/biswa/Downloads/train/dogs'
```

In [3]:

```
# Function to load and preprocess images
def load_images_from_folder(folder, label, image_size=(64, 64)):
    images = []
    labels = []
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)
        img = cv2.imread(img_path)
        if img is not None:
            # Resize image to have consistent dimensions
            img = cv2.resize(img, image_size)
            # Flatten the image into a feature vector
            img_flattened = img.flatten()  # Convert 2D image into 1D arra
            images.append(img_flattened)
            labels.append(label)  # Assign label (1 for dog, 0 for cat, et
    return np.array(images), np.array(labels)
```

In [4]:

```
# Load cat and dog images, label cats as 0 and dogs as 1
cat_images, cat_labels = load_images_from_folder(cat_dir, label=0)
dog_images, dog_labels = load_images_from_folder(dog_dir, label=1)
```

In [5]:

```
# Combine data and labels
X = np.vstack((cat_images, dog_images))
y = np.hstack((cat_labels, dog_labels))
```

In [6]:

```
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
```

In [7]:

```
# Create and train the SVM model
svm_model = SVC(kernel='linear')  # You can try different kernels like 'rb
svm_model.fit(X_train, y_train)
```

Out[7]:

```
SVC(kernel='linear')
```

In [8]:

```
# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 62.14%
```

As we can see here the Accuracy of the fitted classifier is not so good as it is 62.14%, it may classify a new image incorrectly 4 times out of 10 times.

Predict new Images

In [9]:

```
# Function to predict an image using the fitted classifier
def predict_image(image_path, model, image_size=(64, 64)):
    img = cv2.imread(image_path)
    if img is not None:
        img = cv2.resize(img, image_size)
        img_flattened = img.flatten().reshape(1, -1)  # Reshape for predict
        prediction = model.predict(img_flattened)
        return "Dog" if prediction == 1 else "Cat"
    return None

# Test with a new image
result = predict_image("C:/Users/biswa/Downloads/single dog.jpeg", svm_mode
print(f"The image is predicted as: {result}")
result1 = predict_image("C:/Users/biswa/Downloads/single cat.jpeg", svm_mod
print(f"The image is predicted as: {result1}")
```

```
The image is predicted as: Dog
The image is predicted as: Cat
```

In [ ]: