SVM Model

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.calibration import CalibratedClassifierCV
import os
from PIL import Image
from tqdm import tqdm
from time import time
```

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

```
--2024-12-11 17:45:18--  https://download.microsoft.com/download/3/E/1/3E1C3F2
Resolving download.microsoft.com (download.microsoft.com)... 23.197.2.18, 2600
Connecting to download.microsoft.com (download.microsoft.com)|23.197.2.18|:443
HTTP request sent, awaiting response... 200 OK
Length: 824887076 (787M) [application/octet-stream]
Saving to: ‘kagglecatsanddogs_5340.zip’

kagglecatsanddogs_5 100%[===================>] 786.67M   148MB/s    in 5.6s

2024-12-11 17:45:24 (140 MB/s) - ‘kagglecatsanddogs_5340.zip’ saved [824887076
```

```
!unzip kagglecatsanddogs_5340.zip
```

Afficher la sortie masquée

```
# Set paths to the dataset folders
cat_folder = '/content/PetImages/Cat'
dog_folder = '/content/PetImages/Dog'

# Function to load and preprocess images
def load_images(folder, label, target_size=(64, 64)):
    images = []
    labels = []

    for filename in tqdm(os.listdir(folder)):
        if filename.endswith(('.jpg', '.jpeg', '.png')):
            try:
                # Load and resize image
                img_path = os.path.join(folder, filename)
                img = Image.open(img_path).convert('RGB')
                img = img.resize(target_size)

                # Convert to array and flatten
                img_array = np.array(img)
                img_flat = img_array.flatten()

                images.append(img_flat)
                labels.append(label)
            except Exception as e:
                print(f"Error processing {filename}: {str(e)}")
                continue

    return np.array(images), np.array(labels)
```

```
# Load cat and dog images
print("Loading cat images...")
X_cats, y_cats = load_images(cat_folder, 0)  # 0 for cats

print("\nLoading dog images...")
X_dogs, y_dogs = load_images(dog_folder, 1)  # 1 for dogs

# Combine datasets
X = np.vstack((X_cats, X_dogs))
y = np.hstack((y_cats, y_dogs))

print(f"\nTotal number of images: {len(X)}")
print(f"Number of cats: {len(X_cats)}")
print(f"Number of dogs: {len(X_dogs)}")
```

```
Loading cat images...
 32%|███▏      | 4038/12501 [00:15<01:00, 139.68it/s]Error processing 666.jpg
100%|██████████| 12501/12501 [00:48<00:00, 259.75it/s]

Loading dog images...
 19%|█▉        | 2361/12501 [00:09<00:36, 275.09it/s]/usr/local/lib/python3.1
  warnings.warn(str(msg))
 85%|████████▍ | 10583/12501 [00:41<00:06, 289.51it/s]Error processing 11702.
100%|██████████| 12501/12501 [00:48<00:00, 255.69it/s]

Total number of images: 24998
Number of cats: 12499
Number of dogs: 12499
```

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

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Train SVM model
print("\nTraining SVM model...")

# Use LinearSVC for faster training with optimized parameters
start_time = time()
base_svm = LinearSVC(
    random_state=42,
    max_iter=5000,  # Increased iterations
    C=0.1,         # Stronger regularization
    class_weight='balanced',  # Handle class imbalance
    dual=False     # Faster for n_samples > n_features
)
# Wrap it in CalibratedClassifierCV to get probability estimates
svm_model = CalibratedClassifierCV(base_svm, cv=3)
```

```
Training SVM model...
```

```
print("Starting training with optimized parameters... This may take a few minutes.
svm_model.fit(X_train, y_train)
training_time = time() - start_time
print(f"Training completed in {training_time:.2f} seconds")

# Make predictions
y_pred = svm_model.predict(X_test)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Cat', 'Dog']))
```

```
Starting training with optimized parameters... This may take a few minutes.
```

```
# Evaluate the model
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Cat', 'Dog']))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```

```
# Function to predict a single image
def predict_image(image_path, model, scaler):
    # Load and preprocess the image
    img = Image.open(image_path).convert('RGB')
    img = img.resize((64, 64))
    img_array = np.array(img)
    img_flat = img_array.flatten().reshape(1, -1)

    # Scale the features
    img_scaled = scaler.transform(img_flat)

    # Make prediction
    prediction = model.predict(img_scaled)[0]

    return 'Dog' if prediction == 1 else 'Cat'
```

```
# To test the model with a new image, we use the following code:

image_path = 'path_to_test_image.jpg'
result = predict_image(image_path, svm_model, scaler)
print(f'Prediction: {result}')
```