Code for SVM Classifier for Cats vs Dogs

```
import os
import zipfile
import numpy as np
import pandas as pd
from sklearn import svm
from sklearn.metrics import accuracy_score
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from sklearn.model_selection import train_test_split
from PIL import Image
import shutil

# Step 1: Unzip the datasets
def unzip_data(zip_path, extract_to):
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_to)

data_paths = [
    '/Users/ronitvyas/Downloads/dogs-vs-cats/train.zip',
    '/Users/ronitvyas/Downloads/dogs-vs-cats/test1.zip'
]
extract_paths = '/Users/ronitvyas/Downloads/dogs-vs-cats/extracted/'

for zip_path in data_paths:
    unzip_data(zip_path, extract_paths)
```

```
# Custom function to validate images
def is_valid_image(file_path):
    try:
        Image.open(file_path)
        return True
    except:
        return False

# Remove invalid files
for root, dirs, files in os.walk(extract_paths):
    for file in files:
        file_path = os.path.join(root, file)
        if not is_valid_image(file_path):
            os.remove(file_path)

# Function to organize images into subdirectories based on class
def organize_images(directory):
    if not os.path.exists(directory):
        return

    for root, dirs, files in os.walk(directory):
        for file in files:
            if 'cat' in file.lower():
                class_dir = os.path.join(directory, 'cat')
            elif 'dog' in file.lower():
```

```
            class_dir = os.path.join(directory, 'dog')
        else:
            continue
        if not os.path.exists(class_dir):
            os.makedirs(class_dir)
        shutil.move(os.path.join(root, file), os.path.join(class_dir, file))

# Organize images in train and test1 directories
organize_images(os.path.join(extract_paths, 'train'))
organize_images(os.path.join(extract_paths, 'test1'))

# Print directory structure
def print_directory_structure(directory):
    for root, dirs, files in os.walk(directory):
        level = root.replace(directory, '').count(os.sep)
        indent = ' ' * 4 * (level)
        print('{}{}/'.format(indent, os.path.basename(root)))
        sub_indent = ' ' * 4 * (level + 1)
        for f in files:
            print('{}{}'.format(sub_indent, f))

print("Train directory structure:")
print_directory_structure(os.path.join(extract_paths, 'train'))

print("Test directory structure:")
print_directory_structure(os.path.join(extract_paths, 'test1'))
```

```
# Step 2: Load and preprocess the images
image_size = (224, 224)
batch_size = 32

# Function to load images and extract features manually
def load_images_and_flatten(directories):
    datagen = ImageDataGenerator(rescale=1./255)
    images_list = []
    labels_list = []

    for directory in directories:
        generator = datagen.flow_from_directory(
            directory,
            target_size=image_size,
            batch_size=batch_size,
            class_mode='binary',
            shuffle=False
        )

        num_images = generator.samples
        if num_images == 0:
                raise ValueError(f"No images found in directory {directory}. Please check the directory structure.")

        for batch_images, batch_labels in generator:
```

```
            for img in batch_images:
                img_flatten = img_to_array(img).flatten()
                images_list.append(img_flatten)
            labels_list.extend(batch_labels)

    return np.array(images_list), np.array(labels_list)

directories = [os.path.join(extract_paths, 'train'), os.path.join(extract_paths, 'test1')]

# Load and extract features from all directories
X, y = load_images_and_flatten(directories)

# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Train an SVM classifier
svm_classifier = svm.SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)

# Step 4: Evaluate the classifier
y_pred = svm_classifier.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Save the model
import joblib
```

```
joblib.dump(svm_classifier, '/Users/ronitvyas/Downloads/dogs-vs-cats/svm_cat_dog_classifier.pkl')


# Save the evaluation results

results = pd.DataFrame({'True Label': y_val, 'Predicted Label': y_pred})

results.to_csv('/Users/ronitvyas/Downloads/dogs-vs-cats/svm_evaluation_results.csv', index=False)
```