TASK 3

Implement a support vector machine (SVM) to classify images of cats and dogs from the Kaggle dataset.

In [ ]:

```
import os
import numpy as np
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
from tqdm import tqdm
import joblib
from sklearn.model_selection import GridSearchCV
import cv2
import seaborn as sns
import time
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

In [2]:

```
folder_path = f"/Users/manojt/Downloads/Data"
os.makedirs(folder_path, exist_ok=True)

confusion_image_path = os.path.join(folder_path, 'confusion matrix.png
classification_file_path = os.path.join(folder_path, 'classification_
model_file_path = os.path.join(folder_path, "svm_model.pkl")

dataset_dir = "/Users/manojt/Downloads/Data"
train_dir = os.path.join(dataset_dir, "train")
test_dir = os.path.join(dataset_dir, "test1")
```

In [3]:

```
train_images = os.listdir(train_dir)
features = []
labels = []
image_size = (50, 50)

for image in tqdm(train_images, desc="Processing Train Images"):
    if image[0:3] == 'cat' :
        label = 0
    else :
        label = 1
    image_read = cv2.imread(train_dir+"/"+image)
    image_resized = cv2.resize(image_read, image_size)
    image_normalized = image_resized / 255.0
    image_flatten = image_normalized.flatten()
    features.append(image_flatten)
    labels.append(label)
```

```
Processing Train Images: 100%|██████████| 25000/25000 [05:46<00:0
0, 72.11it/s]
```

In [4]:
```
del train_images
```

In [5]:
```
features = np.asarray(features)
labels = np.asarray(labels)

X_train, X_test, y_train, y_test = train_test_split(features, labels,
```

In [6]:
```
del features
del labels
```

In [7]:
```
n_components = 0.8
pca = PCA(n_components=n_components)
svm = SVC()
pca = PCA(n_components=n_components, random_state=42)
pipeline = Pipeline([
    ('pca', pca),
    ('svm', svm)
])
```

In [8]:
```
param_grid = {
    'pca__n_components': [2, 1, 0.9, 0.8],
    'svm__kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
}
```

In [9]:

```
start_time = time.time()

grid_search = GridSearchCV(pipeline, param_grid, cv=3, verbose=4)
grid_search.fit(X_train, y_train)

end_time = time.time()
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV 1/3] END pca__n_components=2, svm__kernel=linear;, score=0.529 t
otal time=  15.0s
[CV 2/3] END pca__n_components=2, svm__kernel=linear;, score=0.522 t
otal time=  13.9s
[CV 3/3] END pca__n_components=2, svm__kernel=linear;, score=0.539 t
otal time=  14.5s
[CV 1/3] END pca__n_components=2, svm__kernel=rbf;, score=0.572 tota
l time=  10.0s
[CV 2/3] END pca__n_components=2, svm__kernel=rbf;, score=0.560 tota
l time=  10.0s
[CV 3/3] END pca__n_components=2, svm__kernel=rbf;, score=0.564 tota
l time=   9.8s
[CV 1/3] END pca__n_components=2, svm__kernel=poly;, score=0.494 tot
al time=   8.9s
[CV 2/3] END pca__n_components=2, svm__kernel=poly;, score=0.492 tot
al time=   8.8s
[CV 3/3] END pca__n_components=2, svm__kernel=poly;, score=0.498 tot
al time=   8.6s
[CV 1/3] END pca__n_components=2, svm__kernel=sigmoid;, score=0.502 
total time=   9.5s
[CV 2/3] END pca__n_components=2, svm__kernel=sigmoid;, score=0.508 
total time=   9.7s
[CV 3/3] END pca__n_components=2, svm__kernel=sigmoid;, score=0.510 
total time=  10.2s
[CV 1/3] END pca__n_components=1, svm__kernel=linear;, score=0.521 t
otal time=   9.9s
[CV 2/3] END pca__n_components=1, svm__kernel=linear;, score=0.517 t
otal time=   9.4s
[CV 3/3] END pca__n_components=1, svm__kernel=linear;, score=0.518 t
otal time=   9.7s
[CV 1/3] END pca__n_components=1, svm__kernel=rbf;, score=0.527 tota
l time=  10.0s
[CV 2/3] END pca__n_components=1, svm__kernel=rbf;, score=0.527 tota
l time=  10.5s
[CV 3/3] END pca__n_components=1, svm__kernel=rbf;, score=0.527 tota
l time=  10.3s
[CV 1/3] END pca__n_components=1, svm__kernel=poly;, score=0.501 tot
al time=   8.2s
[CV 2/3] END pca__n_components=1, svm__kernel=poly;, score=0.496 tot
al time=   8.2s
[CV 3/3] END pca__n_components=1, svm__kernel=poly;, score=0.500 tot
al time=   8.2s
[CV 1/3] END pca__n_components=1, svm__kernel=sigmoid;, score=0.502 
total time=   8.8s
[CV 2/3] END pca__n_components=1, svm__kernel=sigmoid;, score=0.508 
total time=   8.8s
[CV 3/3] END pca__n_components=1, svm__kernel=sigmoid;, score=0.504 
total time=   8.9s
[CV 1/3] END pca__n_components=0.9, svm__kernel=linear;, score=0.602 
total time=15.2min
[CV 2/3] END pca__n_components=0.9, svm__kernel=linear;, score=0.613 
total time=15.5min
[CV 3/3] END pca__n_components=0.9, svm__kernel=linear;, score=0.610 
total time=15.5min
[CV 1/3] END pca__n_components=0.9, svm__kernel=rbf;, score=0.675 to
tal time= 6.4min
[CV 2/3] END pca__n_components=0.9, svm__kernel=rbf;, score=0.674 to
tal time= 6.4min
[CV 3/3] END pca__n_components=0.9, svm__kernel=rbf;, score=0.673 to
tal time= 6.5min
```

```
[CV 1/3] END pca__n_components=0.9, svm__kernel=poly;, score=0.612 t
otal time= 6.4min
[CV 2/3] END pca__n_components=0.9, svm__kernel=poly;, score=0.607 t
otal time= 6.5min
[CV 3/3] END pca__n_components=0.9, svm__kernel=poly;, score=0.600 t
otal time= 6.5min
[CV 1/3] END pca__n_components=0.9, svm__kernel=sigmoid;, score=0.51
8 total time= 6.1min
[CV 2/3] END pca__n_components=0.9, svm__kernel=sigmoid;, score=0.53
0 total time= 6.2min
[CV 3/3] END pca__n_components=0.9, svm__kernel=sigmoid;, score=0.52
4 total time= 6.1min
[CV 1/3] END pca__n_components=0.8, svm__kernel=linear;, score=0.585
total time= 8.9min
[CV 2/3] END pca__n_components=0.8, svm__kernel=linear;, score=0.591
total time= 8.8min
[CV 3/3] END pca__n_components=0.8, svm__kernel=linear;, score=0.594
total time= 9.0min
[CV 1/3] END pca__n_components=0.8, svm__kernel=rbf;, score=0.665 to
tal time= 5.9min
[CV 2/3] END pca__n_components=0.8, svm__kernel=rbf;, score=0.662 to
tal time= 5.9min
[CV 3/3] END pca__n_components=0.8, svm__kernel=rbf;, score=0.667 to
tal time= 5.9min
[CV 1/3] END pca__n_components=0.8, svm__kernel=poly;, score=0.595 t
otal time= 5.8min
[CV 2/3] END pca__n_components=0.8, svm__kernel=poly;, score=0.600 t
otal time= 5.9min
[CV 3/3] END pca__n_components=0.8, svm__kernel=poly;, score=0.593 t
otal time= 5.9min
[CV 1/3] END pca__n_components=0.8, svm__kernel=sigmoid;, score=0.51
6 total time= 5.8min
[CV 2/3] END pca__n_components=0.8, svm__kernel=sigmoid;, score=0.52
5 total time= 5.8min
[CV 3/3] END pca__n_components=0.8, svm__kernel=sigmoid;, score=0.52
1 total time= 5.8min
```

In [10]:

```python
del X_train
del y_train
```

In [11]:

```python
best_pipeline = grid_search.best_estimator_
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters: ", best_params)
print("Best Score: ", best_score)
```

```
Best Parameters:  {'pca__n_components': 0.9, 'svm__kernel': 'rbf'}
Best Score:  0.673899930866043
```

In [12]:

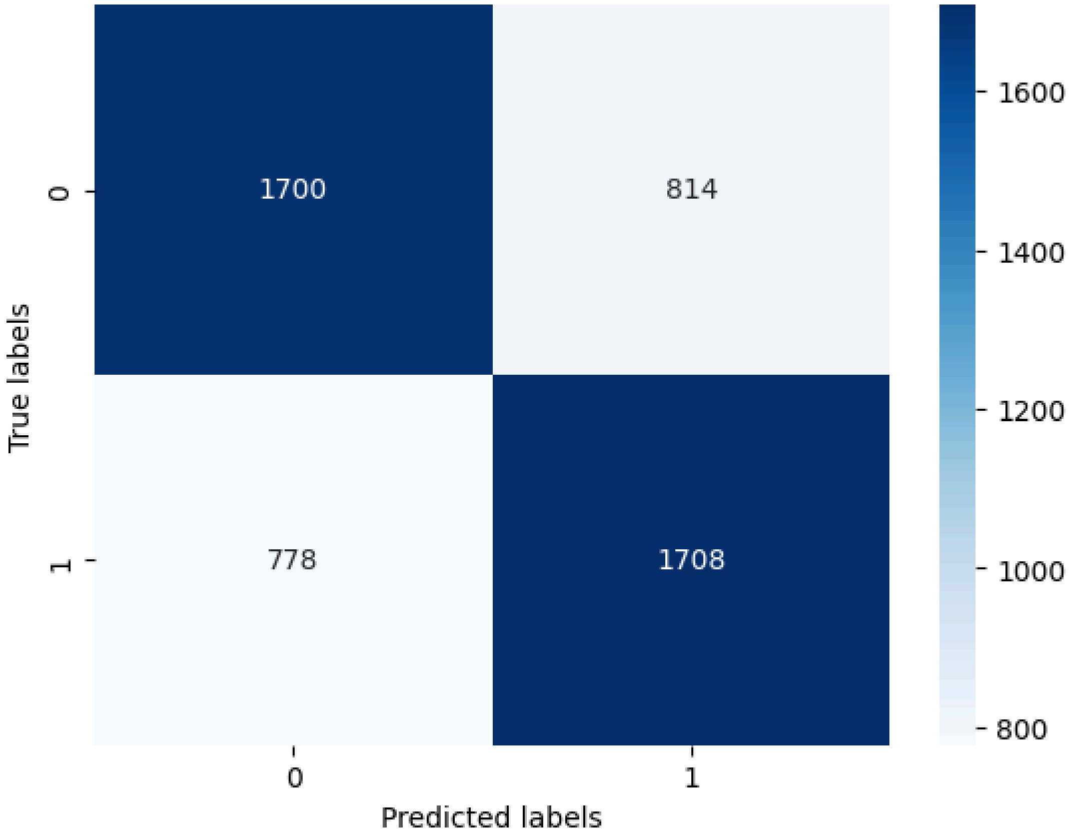
```python
accuracy = best_pipeline.score(X_test, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6816
```

In [13]:
```
y_pred = best_pipeline.predict(X_test)
target_names = ['Cat', 'Dog']
classification_rep = classification_report(y_test, y_pred, target_name
print("Classification Report:\n", classification_rep)

with open(classification_file_path, 'w') as file:
    file.write(classification_rep)
```

```
Classification Report:
               precision    recall  f1-score   support

         Cat       0.69      0.68      0.68      2514
         Dog       0.68      0.69      0.68      2486

    accuracy                           0.68      5000
   macro avg       0.68      0.68      0.68      5000
weighted avg       0.68      0.68      0.68      5000
```

In [14]:
```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.savefig(confusion_image_path)
plt.show()
```



In [ ]: