```
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
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

```
# extract dataset
from zipfile import ZipFile

dataset_train = "train.zip"

with ZipFile(dataset_train, 'r') as zip:
    zip.extractall()

folder_path = f"Dataset/"
os.makedirs(folder_path, exist_ok=True)

# define path
confusion_image_path = os.path.join(folder_path, 'confusion
matrix.png')
classification_file_path = os.path.join(folder_path,
'classification_report.txt')
model_file_path = os.path.join(folder_path, "svm_model.pkl")

# Path dataset
dataset_dir = "Dataset/"
train_dir = os.path.join(dataset_dir, "train")
test_dir = os.path.join(dataset_dir, "test1")

# load data, preprocessing data, and labeling
# dog = 1, cat = 0
train_images = os.listdir(train_dir)
features = []
labels = []
image_size = (50, 50)
```

```
# Proses train images
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 [00:43<00:00,
575.99it/s]
```

```
del train_images

features = np.asarray(features)
labels = np.asarray(labels)

# train test split
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, shuffle=True, random_state=42)

del features
del labels

# PCA, SVM, & Pipeline
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)
])

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

# Hitung waktu training
start_time = time.time()

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

# Hitung waktu training
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=0.527
total time=  16.8s
[CV 2/3] END pca__n_components=2, svm__kernel=linear;, score=0.533
total time=  15.6s
[CV 3/3] END pca__n_components=2, svm__kernel=linear;, score=0.530
total time=  16.2s
[CV 1/3] END pca__n_components=2, svm__kernel=rbf;, score=0.566 total
time=  12.9s
[CV 2/3] END pca__n_components=2, svm__kernel=rbf;, score=0.573 total
time=  12.8s
[CV 3/3] END pca__n_components=2, svm__kernel=rbf;, score=0.565 total
time=  12.6s
[CV 1/3] END pca__n_components=2, svm__kernel=poly;, score=0.496 total
time=  11.2s
[CV 2/3] END pca__n_components=2, svm__kernel=poly;, score=0.512 total
time=  11.3s
[CV 3/3] END pca__n_components=2, svm__kernel=poly;, score=0.496 total
time=  11.2s
[CV 1/3] END pca__n_components=2, svm__kernel=sigmoid;, score=0.501
total time=  13.1s
[CV 2/3] END pca__n_components=2, svm__kernel=sigmoid;, score=0.509
total time=  12.0s
[CV 3/3] END pca__n_components=2, svm__kernel=sigmoid;, score=0.498
total time=  14.0s
[CV 1/3] END pca__n_components=1, svm__kernel=linear;, score=0.519
total time=  12.1s
[CV 2/3] END pca__n_components=1, svm__kernel=linear;, score=0.518
total time=  11.3s
[CV 3/3] END pca__n_components=1, svm__kernel=linear;, score=0.514
total time=  11.8s
[CV 1/3] END pca__n_components=1, svm__kernel=rbf;, score=0.531 total
time=  12.8s
[CV 2/3] END pca__n_components=1, svm__kernel=rbf;, score=0.530 total
time=  12.6s
[CV 3/3] END pca__n_components=1, svm__kernel=rbf;, score=0.532 total
time=  13.3s
[CV 1/3] END pca__n_components=1, svm__kernel=poly;, score=0.499 total
time=  10.7s
[CV 2/3] END pca__n_components=1, svm__kernel=poly;, score=0.503 total
time=  10.1s
[CV 3/3] END pca__n_components=1, svm__kernel=poly;, score=0.499 total
time=  10.7s
[CV 1/3] END pca__n_components=1, svm__kernel=sigmoid;, score=0.506
total time=  11.3s
[CV 2/3] END pca__n_components=1, svm__kernel=sigmoid;, score=0.507
total time=  11.1s
[CV 3/3] END pca__n_components=1, svm__kernel=sigmoid;, score=0.506
total time=  11.2s
[CV 1/3] END pca__n_components=0.9, svm__kernel=linear;, score=0.608
```

```
total time=14.6min
[CV 2/3] END pca__n_components=0.9, svm__kernel=linear;, score=0.606
total time=14.6min
[CV 3/3] END pca__n_components=0.9, svm__kernel=linear;, score=0.605
total time=17.6min
[CV 1/3] END pca__n_components=0.9, svm__kernel=rbf;, score=0.674
total time= 4.4min
[CV 2/3] END pca__n_components=0.9, svm__kernel=rbf;, score=0.680
total time= 4.4min
[CV 3/3] END pca__n_components=0.9, svm__kernel=rbf;, score=0.673
total time= 4.3min
[CV 1/3] END pca__n_components=0.9, svm__kernel=poly;, score=0.606
total time= 4.4min
[CV 2/3] END pca__n_components=0.9, svm__kernel=poly;, score=0.610
total time= 4.6min
[CV 3/3] END pca__n_components=0.9, svm__kernel=poly;, score=0.605
total time= 4.5min
[CV 1/3] END pca__n_components=0.9, svm__kernel=sigmoid;, score=0.521
total time= 3.8min
[CV 2/3] END pca__n_components=0.9, svm__kernel=sigmoid;, score=0.516
total time= 3.8min
[CV 3/3] END pca__n_components=0.9, svm__kernel=sigmoid;, score=0.521
total time= 3.9min
[CV 1/3] END pca__n_components=0.8, svm__kernel=linear;, score=0.587
total time= 5.5min
[CV 2/3] END pca__n_components=0.8, svm__kernel=linear;, score=0.587
total time= 5.3min
[CV 3/3] END pca__n_components=0.8, svm__kernel=linear;, score=0.589
total time= 5.2min
[CV 1/3] END pca__n_components=0.8, svm__kernel=rbf;, score=0.663
total time= 3.5min
[CV 2/3] END pca__n_components=0.8, svm__kernel=rbf;, score=0.668
total time= 3.5min
[CV 3/3] END pca__n_components=0.8, svm__kernel=rbf;, score=0.659
total time= 3.4min
[CV 1/3] END pca__n_components=0.8, svm__kernel=poly;, score=0.597
total time= 3.4min
[CV 2/3] END pca__n_components=0.8, svm__kernel=poly;, score=0.606
total time= 3.4min
[CV 3/3] END pca__n_components=0.8, svm__kernel=poly;, score=0.592
total time= 3.4min
[CV 1/3] END pca__n_components=0.8, svm__kernel=sigmoid;, score=0.517
total time= 3.4min
[CV 2/3] END pca__n_components=0.8, svm__kernel=sigmoid;, score=0.516
total time= 3.4min
[CV 3/3] END pca__n_components=0.8, svm__kernel=sigmoid;, score=0.520
total time= 3.3min
```

```
del X_train
del y_train
```

```python
# Mendapatkan model terbaik dan parameter terbaik
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:  0.6756998783724181
```

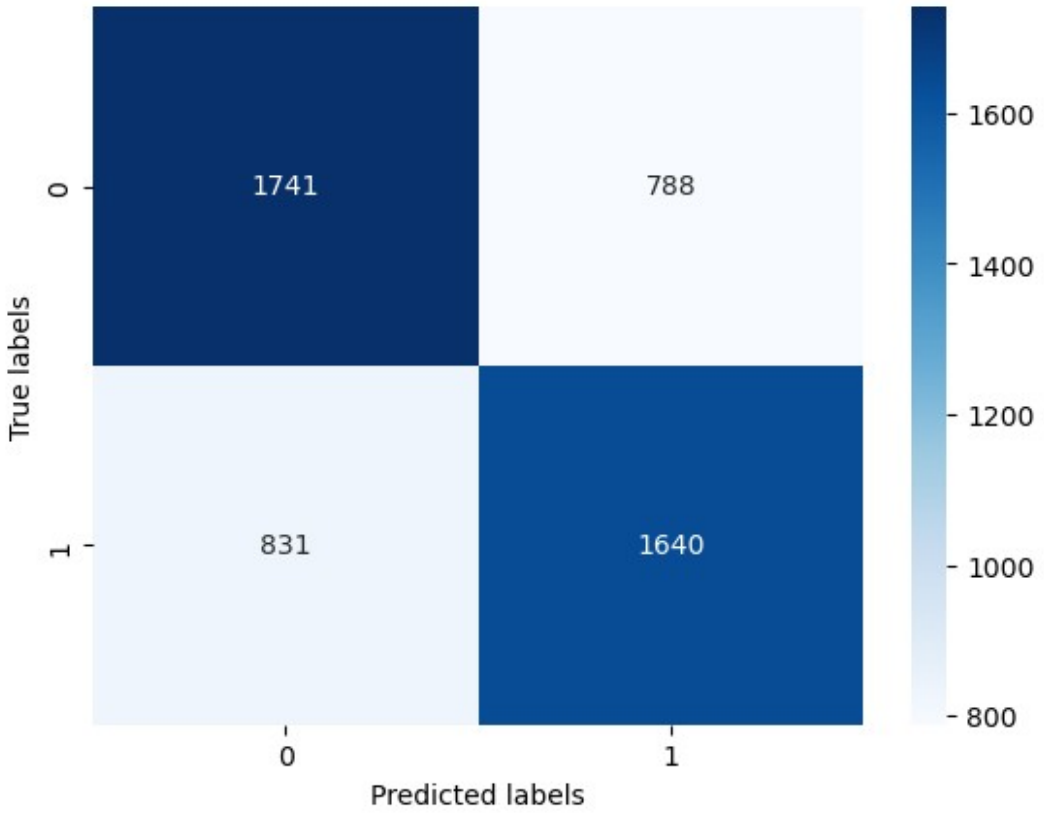
```python
# Evaluation on test dataset
accuracy = best_pipeline.score(X_test, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6762
```

```python
y_pred = best_pipeline.predict(X_test)

# classification report
target_names = ['Cat', 'Dog']
classification_rep = classification_report(y_test, y_pred,
target_names=target_names)
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       0.68      0.69      0.68      2529
         Dog       0.68      0.66      0.67      2471

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

```python
# Confusion matrix
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()
```


1741
788
831
1640
0
1
True labels
0
1
Predicted labels
1600
1400
1200
1000
800