NEGI AMIT SINGH
23552006

Support Vector Machine (SVM) Example with Handwritten Digit Recognition

Problem statement: In Scikit-Learn handwritten dataset, we have 1797 images of 8x8 pixels. •By using Support Vector Machine (SVM), We need to train the model in such a way that the model can identify a new handwritten digit which is not available in training dataset but available in test dataset.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn import svm


# Load  the dastset
digits = load_digits()


# shape of dataset
digits.data.shape


# shape of image
digits.images.shape
```

NEGI AMIT SINGH
23552006

Visualization of first image in the datatset

```
# Get the first five images and their corresponding labels
x = digits.data[:5].reshape(-1, 8, 8)
y = digits.target[:5]

# Plot the first five images
fig, axes = plt.subplots(1, 5, figsize=(12, 3))

for i in range(5):
    axes[i].imshow(x[i], cmap='gray', interpolation='nearest')
    axes[i].set_title(f"Label: {y[i]}")
    axes[i].axis('off')

plt.show()

print(digits['DESCR'])
```

 # Data Export

```
df = pd.DataFrame(data=digits.data, columns=[f'pixel_{i}' for i in range(digits.data.shape[1])])
df['target'] = digits.target

# Specify the path where you want to save the CSV file
csv_path = 'C:/Users/negia/Downloads/Assignment 2/digits_dataset.csv'

# Save the DataFrame to a CSV file
```

```
df.to_csv(csv_path, index=False)

print(f"Digits dataset exported to: {csv_path}")

df=pd.read_csv('digits_dataset.csv')

X = df.drop('target', axis=1)
y = df['target']
```

Model Training

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 100)
X_train.shape
```

Data preprocessing

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# fit_transform use to do some calculation and then do transformation
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Model construct

```
model = svm.SVC
rbf_model = model(kernel='rbf')
rbf_model.fit(X_train, y_train)
```

```
y_rbf_pred = rbf_model.predict(X_test)

y_rbf_pred

print("Predictad Values :\n" ,y_rbf_pred[10:15])
print ("Actual Values :\n",y_test[10:15])

# Accuracy

from sklearn import metrics
acc_rbf= metrics.accuracy_score(y_test, y_rbf_pred)
print("Accuracy:","{:.2f}".format(acc_rbf*100),"%")
```

ACCURACY :- 98.89%

```
# Testing the model using random input
Input:-
        for i in (np.random.randint(0,len(X_test),4)):
          two_d = (np.reshape(X_test[i], (8, 8)))
          plt.title('predicted label: {0}'. format(y_rbf_pred[i]))
          plt.imshow(two_d, cmap='gray')
          plt.show()
```

NEGI AMIT SINGH
23552006

OUTPUT