❖ Day -7: Introduction to SVM

❖ SVM:

Support Vector Machine (SVM) is a machine learning algorithm that finds the best hyperplane to separate classes in a data, with applications in classification and regression.

❖ Types:

I. Linear SVM
II. Non-linear SVM
III. Binary SVM
IV. Multiclass SVM
V. Support Vector Regression

❖ Working:

➢ Support Vector Machines (SVMs) work by:

1.Training: Find the hyperplane that maximizes the margin between classes.

2.kernel Transformation: Handle non-linear patterns using kernel functions.

3.classification:Assign new data points to classes based on their position relative to the hyperplane.

❖ Advantage:

- Effective in high dimensions: performs well in dataset with many features.
- Versatile: handle linear and non-linear relationships using kernel functions.

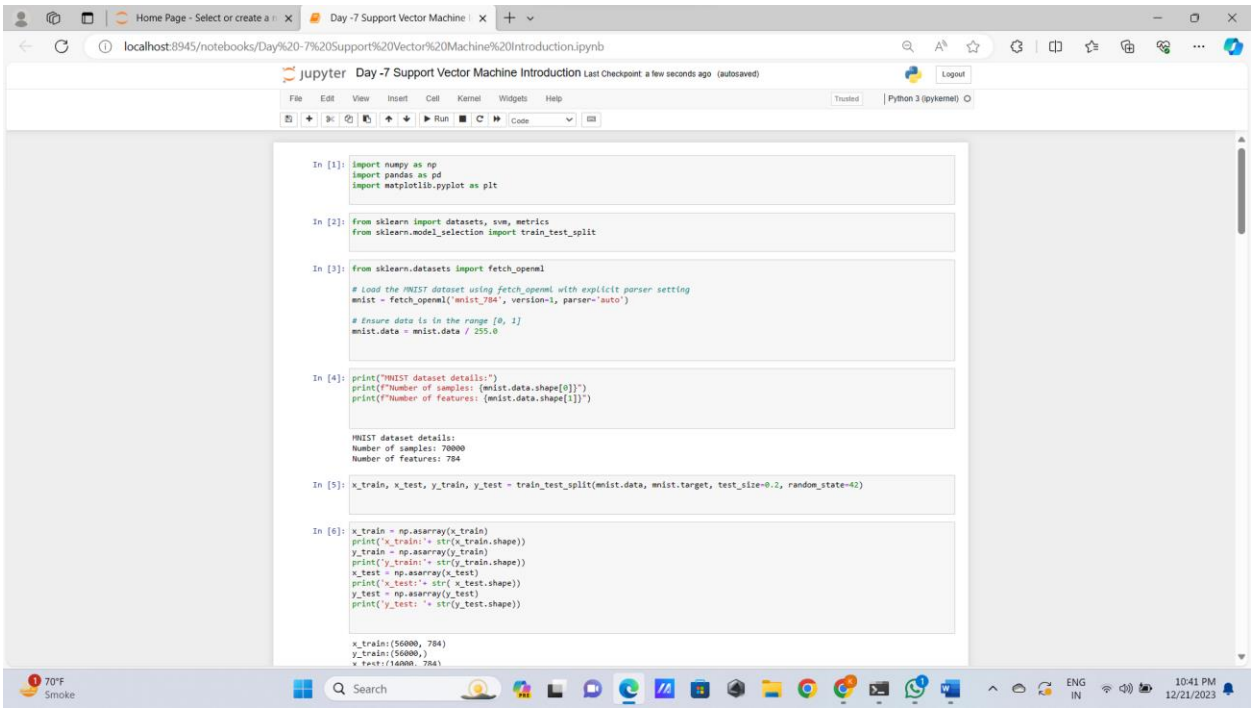
❖ Disadvantage:

- Computational intensity: High computational demand, especially for large datasets.

- Memory Usage: Substantial memory requirements, particularly with non-linear kernels.

❖Description:

I'm thrilled to apply this newfound knowledge to real-world scenarios, and the prospect of further honing these skills through the Skill Boost Internship Program adds an extra layer of excitement. Here's to the journey ahead and the exciting challenges awaiting in the Skill Boost Internship Program(www.Batweb.com).



Home Page - Select or create a n | Day -7 Support Vector Machine I

localhost:8945/notebooks/Day%20-7%20Support%20Vector%20Machine%20Introduction.ipynb

jupyter **Day -7 Support Vector Machine Introduction** Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```
x_train:(56000, 784)
y_train:(56000,)
x_test:(14000, 784)
y_test: (14000,)
```

In [7]:
```
# normalize and scale the data

x_train_flat=x_train/255.0
x_test_flat=x_test/255.0

x_train_flat=x_train.reshape(x_train.shape[0],-1)
x_test_flat=x_test.reshape(x_test.shape[0],-1)

x_train_flat=x_train_flat[0:100][:]
y_train=y_train[0:100]
```

In [8]:
```
# train model
from sklearn.svm import SVC
model_linear=SVC(kernel='linear')
model_linear.fit(x_train_flat,y_train)
```

Out[8]:
```
SVC(kernel='linear')
```

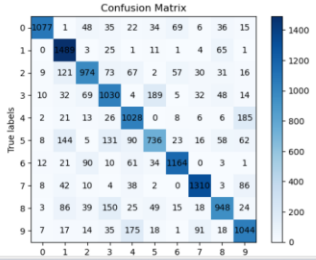
In [9]:
```
# get the accuracy
from sklearn import metrics
#predict
y_pred=model_linear.predict(x_test_flat)

#accuracy
print("accuracy",metrics.accuracy_score(y_test,y_pred),)
```

```
accuracy 0.7714285714285715
```

In [10]:
```
#confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[1077    1   48   35   22   34   69    6   36   15]
 [   0 1489    3   25    1   11    1    4   65    1]
 [   9  121  974   73   67    2   57   30   31   16]
 [  10   32   69 1030    4  189    5   32   48   14]
 [   2   21   13   26 1028    0    8    6    6  185]
 [   8  144    5  131   90  736   23   16   58   62]
 [  12   21   90   10   61   34 1164    0    3    1]
 [   8   42   10    4   38    2    0 1310    3   86]
 [   3   86   39  150   25   49   15   18  948   24]
 [   7   17   14   35  175   18    1   91   18 1044]]
```

In [11]:
```
#Plotting confusion matrix
plt.imshow(cm, interpolation='nearest', cmap='Blues')
plt.title('Confusion Matrix')
plt.colorbar()

# Add labels
classes = np.arange(len(cm))
plt.xticks(classes, classes)
plt.yticks(classes, classes)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')

# Add annotations
for i in range(len(classes)):
    for j in range(len(classes)):
        plt.text(j, i, str(cm[i, j]), ha='center', va='center', color='black')

plt.show()
```



70°F Smoke

Search

ENG IN

10:42 PM 12/21/2023

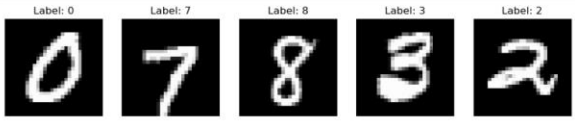
Home Page - Select or create a n | Day -7 Support Vector Machine | localhost:8945/notebooks/Day%20-7%20Support%20Vector%20Machine%20Introduction.ipynb

jupyter Day -7 Support Vector Machine Introduction Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

Predicted labels

In [12]:
```
# Choose a random sample of images to plot
num_samples = 5
sample_indices = np.random.choice(mnist.data.shape[0], num_samples, replace=False)

# Plot the sampled images
plt.figure(figsize=(12, 4))
for i, index in enumerate(sample_indices, 1):
    plt.subplot(1, num_samples, i)
    plt.imshow(mnist.data.iloc[index].values.reshape(28, 28), cmap='gray')  # Use .values to access NumPy array
    plt.title(f"Label: {mnist.target.iloc[index]}")
    plt.axis('off')

plt.show()
```



In [13]:
```
from sklearn.decomposition import PCA
```

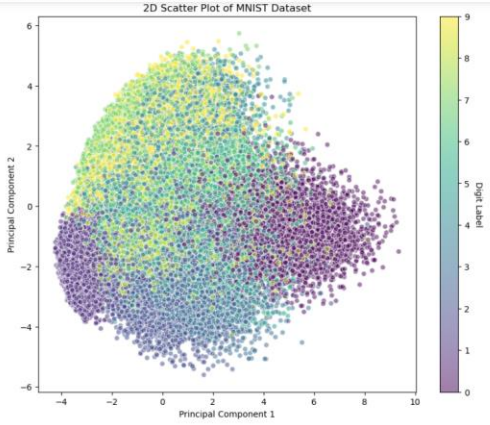
In [14]:
```
# Reduce dimensionality using PCA
pca = PCA(n_components=2)
mnist_2d = pca.fit_transform(mnist.data)

# Create a scatter plot
plt.figure(figsize=(10, 8))
scatter = plt.scatter(mnist_2d[:, 0], mnist_2d[:, 1], c=mnist.target.astype(int), cmap='viridis', alpha=0.5, edgecolors='w')

# Add colorbar
cbar = plt.colorbar(scatter)
cbar.set_label('Digit Label', rotation=270, labelpad=15)

plt.title('2D Scatter Plot of MNIST Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

70°F Smoke | Search | ENG IN | 10:42 PM 12/21/2023

Home Page - Select or create a n | Day -7 Support Vector Machine | localhost:8945/notebooks/Day%20-7%20Support%20Vector%20Machine%20Introduction.ipynb

jupyter Day -7 Support Vector Machine Introduction Last Checkpoint: a minute ago (autosaved)

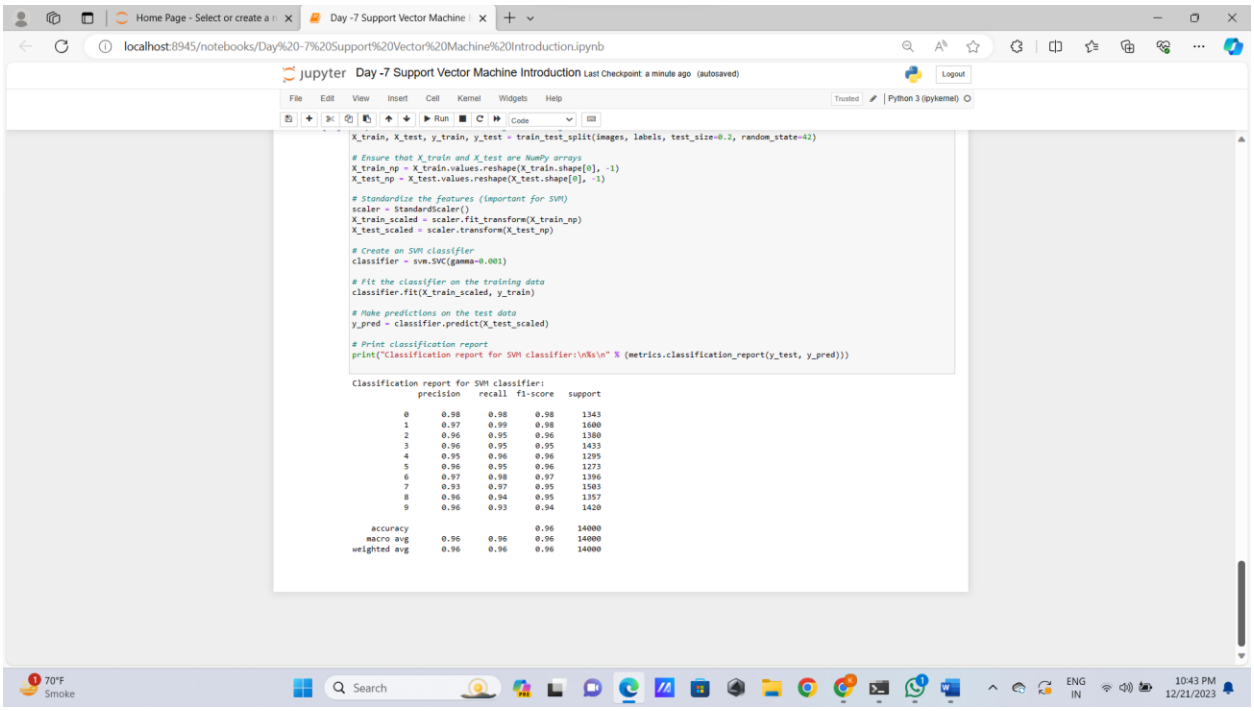
File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)



In [15]:
```
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [18]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

# Ensure that X_train and X_test are NumPy arrays
```

70°F Smoke | Search | ENG IN | 10:43 PM 12/21/2023


Home Page - Select or create a n
Day -7 Support Vector Machine I
localhost:8945/notebooks/Day%20-7%20Support%20Vector%20Machine%20Introduction.ipynb
jupyter Day -7 Support Vector Machine Introduction Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)
# Ensure that X_train and X_test are NumPy arrays
X_train_np = X_train.values.reshape(X_train.shape[0], -1)
X_test_np = X_test.values.reshape(X_test.shape[0], -1)
# Standardize the features (important for SVM)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_np)
X_test_scaled = scaler.transform(X_test_np)
# Create an SVM classifier
classifier = svm.SVC(gamma=0.001)
# Fit the classifier on the training data
classifier.fit(X_train_scaled, y_train)
# Make predictions on the test data
y_pred = classifier.predict(X_test_scaled)
# Print classification report
print("Classification report for SVM classifier:\n%s\n" % (metrics.classification_report(y_test, y_pred)))
Classification report for SVM classifier:
precision recall f1-score support
0 0.98 0.98 0.98 1343
1 0.97 0.99 0.98 1600
2 0.96 0.95 0.96 1380
3 0.96 0.95 0.95 1433
4 0.95 0.96 0.96 1295
5 0.96 0.95 0.96 1273
6 0.97 0.98 0.97 1396
7 0.93 0.97 0.95 1503
8 0.96 0.94 0.95 1357
9 0.96 0.93 0.94 1420
accuracy 0.96 14000
macro avg 0.96 0.96 0.96 14000
weighted avg 0.96 0.96 0.96 14000
70°F
Smoke
Search
ENG
IN
10:43 PM
12/21/2023