In [162…

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from scipy.io import loadmat
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin
```

In [ ]:

```
# Step 1: Load digits012.mat data and split between x/y.
digits = loadmat('/content/digits012.mat')
X = digits['x']
y = digits['y'][0,:]
```

In [164…

```
# Step 2: Split the data into train and test sets using a 70-30 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Fit an SVM classifier
y_train[np.where(y_train == '0')[0]] = 0
y_train[np.where(y_train == '1')[0]] = 1
y_train[np.where(y_train == '2')[0]] = 2

y_test[np.where(y_test == '0')[0]] = 0
y_test[np.where(y_test == '1')[0]] = 1
y_test[np.where(y_test == '2')[0]] = 2

y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

In [165…

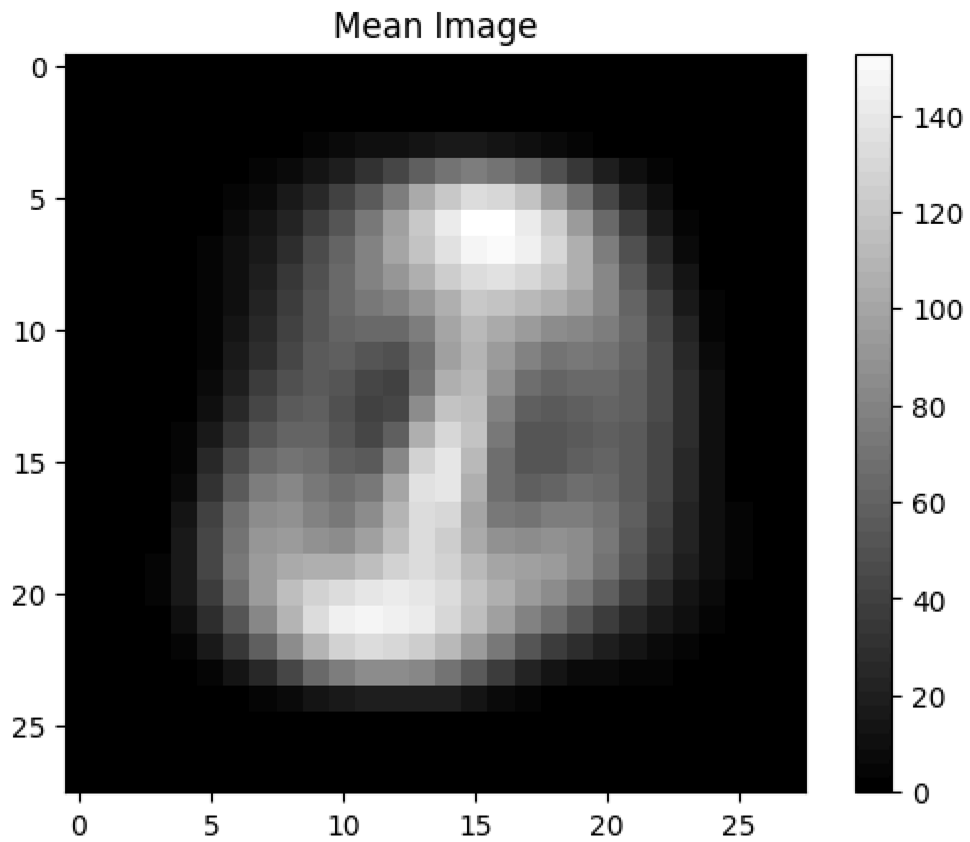
```
# Step 3: Perform PCA analysis and take out the best number of components that will specify ~95%
# of information from original dataset features.
var_thr = 0.95
pca = PCA(n_components=784).fit(X_train)
pca_cumsum = np.cumsum(pca.explained_variance_ratio_)

var_comp = np.where(pca_cumsum >= var_thr)[0]
var_comp = var_comp[0]
print('No. of required components for explaining {}% variance: {}'.format(int(var_thr*100),var_comp))
```

```
No. of required components for explaining 95% variance: 125
```

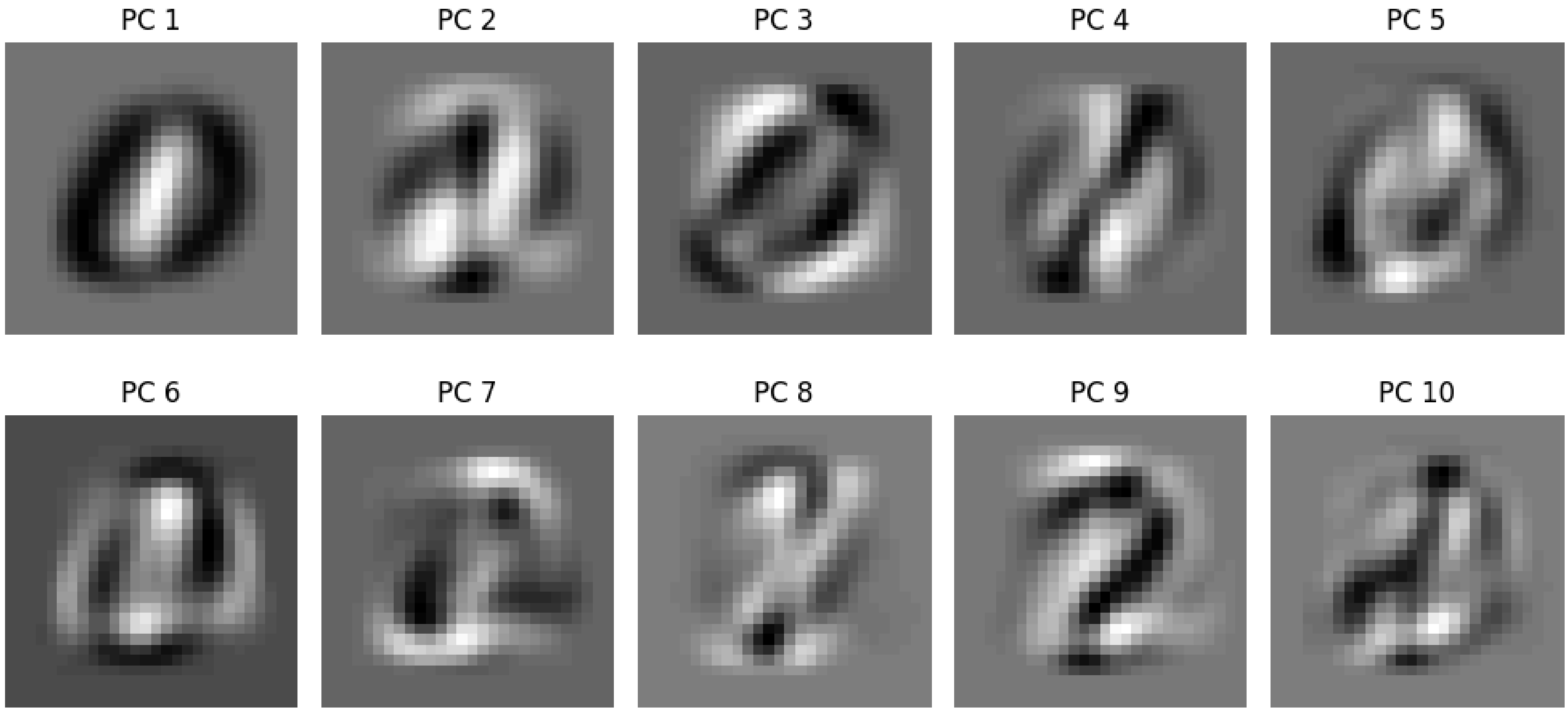
In [166…

```
# Step 4: Plot the Mean Image
mean_image = pca.mean_.reshape(28, 28)
plt.imshow(mean_image, cmap='gray')
plt.title("Mean Image")
plt.colorbar()
plt.show()
```



In [167…

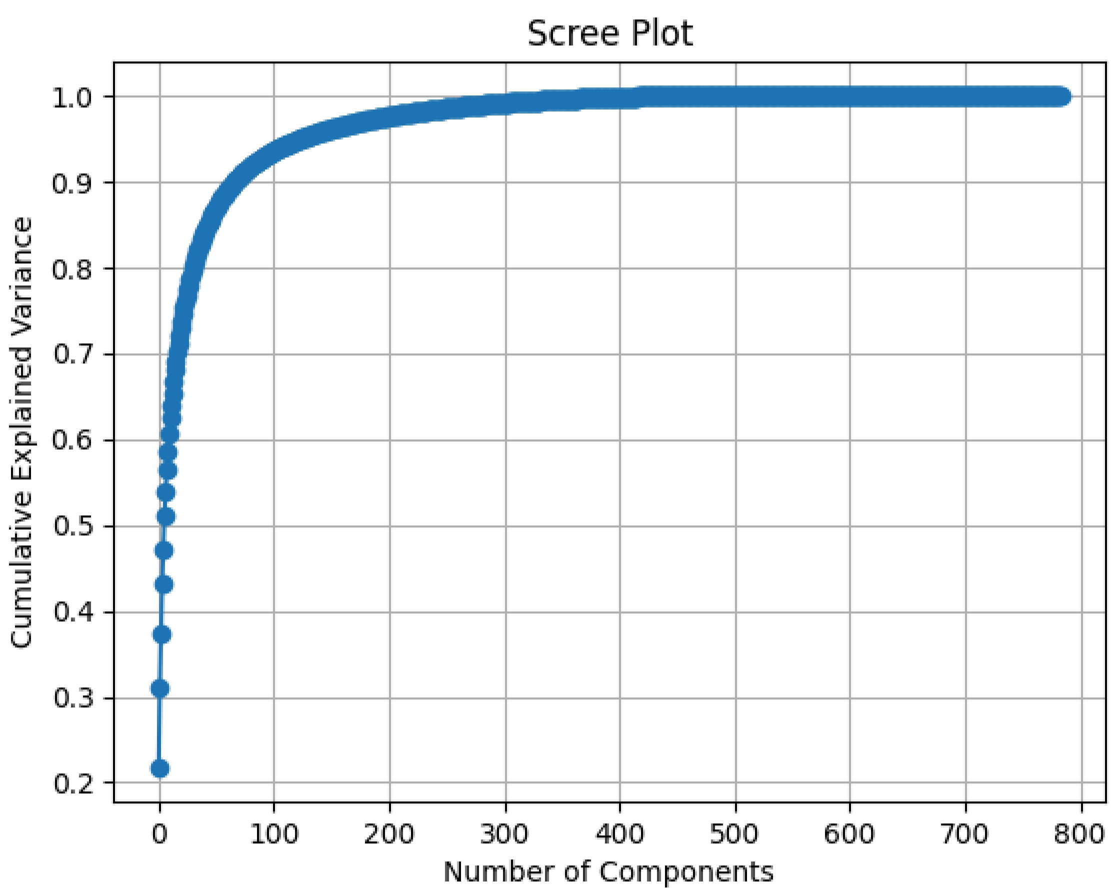
```
# Step 5: Plot First 10 Principal Components
fig, axes = plt.subplots(2, 5, figsize=(10, 5))
for i, ax in enumerate(axes.flat):
    component = pca.components_[i].reshape(28, 28)
    ax.imshow(component, cmap='gray')
    ax.set_title(f"PC {i+1}")
    ax.axis('off')
plt.tight_layout()
plt.show()
```



In [168…

```
# Step 6: Scree Plot
plt.plot(pca_cumsum, marker='o')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("Scree Plot")
plt.grid()
plt.show()

# Print the explained variance ratio for each principal component
print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```



```
Explained Variance Ratio: [2.17514496e-01 9.21426246e-02 6.42716363e-02 5.67331234e-02
 4.15734560e-02 3.86336066e-02 2.85137500e-02 2.48212183e-02
 2.21539401e-02 1.99043052e-02 1.80201236e-02 1.47820833e-02
 1.47345147e-02 1.32254364e-02 1.26767578e-02 1.13378686e-02
 1.06021142e-02 9.85061213e-03 9.26799187e-03 8.52017665e-03
 8.20551945e-03 7.75528358e-03 7.46124753e-03 6.90451287e-03
 6.85472158e-03 6.53441701e-03 6.37941739e-03 5.77081646e-03
 5.55700341e-03 5.34729307e-03 5.13457053e-03 4.99240894e-03
 4.63826651e-03 4.55376598e-03 4.34154520e-03 3.93304153e-03
 3.86386641e-03 3.69116422e-03 3.45695070e-03 3.37977523e-03
 3.26581084e-03 3.08209483e-03 3.03508295e-03 3.02781826e-03
 2.97075430e-03 2.86604831e-03 2.81317907e-03 2.72995352e-03
 2.63797247e-03 2.50139094e-03 2.44080419e-03 2.37345958e-03
 2.23588009e-03 2.20625910e-03 2.15348145e-03 2.10400548e-03
 2.04127974e-03 1.93161675e-03 1.88084368e-03 1.81988283e-03
 1.79424636e-03 1.77497273e-03 1.66753435e-03 1.60113694e-03
 1.56756810e-03 1.56421096e-03 1.51959157e-03 1.46401442e-03
 1.44598172e-03 1.40305160e-03 1.36745456e-03 1.35733494e-03
 1.30842993e-03 1.28565927e-03 1.24751289e-03 1.19990816e-03
 1.17590995e-03 1.14763839e-03 1.12699276e-03 1.09936573e-03
 1.08505544e-03 1.07434889e-03 1.05418153e-03 1.04793197e-03
 1.00114707e-03 9.69037190e-04 9.63102397e-04 9.50562631e-04
 9.31806743e-04 9.25367174e-04 9.07052790e-04 8.79246799e-04
 8.73517175e-04 8.46289717e-04 8.21461954e-04 8.11400484e-04
 8.00735947e-04 7.86624847e-04 7.62436909e-04 7.49527147e-04
 7.43273207e-04 7.26496012e-04 7.18205495e-04 7.10455594e-04
 6.97653821e-04 6.78502970e-04 6.59459062e-04 6.49889182e-04
 6.43643439e-04 6.35702753e-04 6.28084227e-04 6.21111992e-04
 6.06518948e-04 6.01093669e-04 5.97033212e-04 5.92773664e-04
 5.78920241e-04 5.58318350e-04 5.54416366e-04 5.53057678e-04
 5.42157786e-04 5.30525633e-04 5.26470751e-04 5.19856855e-04
 5.09162409e-04 5.02032028e-04 4.96239109e-04 4.90722513e-04
 4.86262302e-04 4.79996528e-04 4.76433953e-04 4.74427414e-04
 4.63149733e-04 4.59940340e-04 4.54287800e-04 4.52233187e-04
 4.45362597e-04 4.34357745e-04 4.32867748e-04 4.26455893e-04
 4.20892605e-04 4.15648350e-04 4.05254470e-04 4.03483594e-04
 3.99672972e-04 3.98019838e-04 3.94306960e-04 3.87492172e-04
 3.84997769e-04 3.74222916e-04 3.68458118e-04 3.66475623e-04
 3.63632630e-04 3.61711130e-04 3.55255540e-04 3.53849373e-04
 3.50697936e-04 3.43595902e-04 3.40585980e-04 3.38015320e-04
 3.34473395e-04 3.33583467e-04 3.30619156e-04 3.26831111e-04
 3.25457123e-04 3.23135147e-04 3.19104139e-04 3.13678210e-04
 3.11470776e-04 3.08250773e-04 3.03660152e-04 3.00529492e-04
 2.97547267e-04 2.95725421e-04 2.91671599e-04 2.89760202e-04
 2.87544379e-04 2.84648395e-04 2.83712015e-04 2.80029360e-04
 2.78175023e-04 2.76198906e-04 2.71823453e-04 2.70183463e-04
 2.66461547e-04 2.62568925e-04 2.59145262e-04 2.56187546e-04
 2.55909861e-04 2.52621525e-04 2.50572705e-04 2.47647345e-04
 2.44182907e-04 2.42381210e-04 2.42162408e-04 2.39756536e-04
 2.37263917e-04 2.35326544e-04 2.34199605e-04 2.30061430e-04
 2.28020957e-04 2.26774683e-04 2.24578105e-04 2.21588809e-04
 2.20274161e-04 2.18367406e-04 2.15922710e-04 2.13503488e-04
 2.12224835e-04 2.11341621e-04 2.09939838e-04 2.09086578e-04
 2.06845606e-04 2.06470587e-04 2.04852981e-04 2.02776677e-04
 1.99715051e-04 1.98158159e-04 1.94820936e-04 1.93607125e-04
 1.93109071e-04 1.91073404e-04 1.89144101e-04 1.86447186e-04
 1.85785680e-04 1.84118565e-04 1.81279262e-04 1.79543430e-04
 1.78776209e-04 1.77335021e-04 1.76455016e-04 1.74914501e-04
 1.73349459e-04 1.71639665e-04 1.70069042e-04 1.68134533e-04
 1.67415743e-04 1.65896362e-04 1.64625893e-04 1.62497436e-04
 1.60713835e-04 1.60284488e-04 1.60112795e-04 1.56819532e-04
 1.56547335e-04 1.56177326e-04 1.54398058e-04 1.53513520e-04
 1.51463173e-04 1.50064271e-04 1.49647920e-04 1.47950887e-04
 1.47103046e-04 1.46338109e-04 1.45163330e-04 1.42967023e-04
 1.42361400e-04 1.42010300e-04 1.39938357e-04 1.39485857e-04
 1.37924162e-04 1.36736095e-04 1.35280735e-04 1.34380158e-04
 1.33483118e-04 1.33126128e-04 1.31414299e-04 1.30606936e-04
 1.29927408e-04 1.29234623e-04 1.27788675e-04 1.26535772e-04
 1.25387784e-04 1.24834307e-04 1.23601836e-04 1.22133141e-04
 1.20594901e-04 1.20018650e-04 1.19347834e-04 1.18641523e-04
 1.18567817e-04 1.16822856e-04 1.16084868e-04 1.14808157e-04
 1.14652734e-04 1.12890130e-04 1.11805048e-04 1.10552458e-04
 1.09594716e-04 1.09114492e-04 1.08140396e-04 1.07937779e-04
 1.07360143e-04 1.06283864e-04 1.06203833e-04 1.04549872e-04
 1.04392092e-04 1.03983433e-04 1.03063306e-04 1.02350980e-04
 1.01997138e-04 1.01265370e-04 9.98035024e-05 9.89772726e-05
 9.75289430e-05 9.72660882e-05 9.65610644e-05 9.57558687e-05
 9.51972467e-05 9.45617976e-05 9.38691999e-05 9.32864695e-05
 9.23376603e-05 9.15742209e-05 9.11642406e-05 9.08209345e-05
 9.00593590e-05 8.91829765e-05 8.89731915e-05 8.77406837e-05
 8.68107094e-05 8.65501891e-05 8.58748864e-05 8.50830409e-05
 8.45403099e-05 8.35329308e-05 8.33061178e-05 8.28603850e-05
 8.24282414e-05 8.14824735e-05 8.08400584e-05 7.96904876e-05
 7.96762900e-05 7.91057222e-05 7.79842343e-05 7.75770844e-05
 7.66031702e-05 7.59614765e-05 7.45095919e-05 7.43545904e-05
 7.37723579e-05 7.37538847e-05 7.29712116e-05 7.26650500e-05
 7.24152290e-05 7.16328068e-05 7.12644126e-05 7.05429396e-05
 7.01811880e-05 6.96895510e-05 6.91840434e-05 6.83719962e-05
 6.76099322e-05 6.67724933e-05 6.63984577e-05 6.62962904e-05
 6.55926170e-05 6.47371713e-05 6.39880422e-05 6.31414591e-05
 6.28718459e-05 6.27047239e-05 6.17474584e-05 6.08894934e-05
 6.06941657e-05 6.00801580e-05 5.93560779e-05 5.87773679e-05
 5.82629231e-05 5.75477089e-05 5.65813620e-05 5.65537901e-05
 5.58738348e-05 5.55784910e-05 5.53757999e-05 5.40915535e-05
 5.37993250e-05 5.33577242e-05 5.26070441e-05 5.21985108e-05
 5.11411924e-05 5.08419594e-05 5.00492224e-05 4.97244967e-05
```

```
4.94579524e-05 4.88362139e-05 4.79024821e-05 4.70924734e-05
4.64520798e-05 4.53447036e-05 4.48122113e-05 4.46549488e-05
4.43454128e-05 4.36242199e-05 4.30548835e-05 4.25392346e-05
4.23978857e-05 4.03677408e-05 4.01625142e-05 3.95147441e-05
3.93341333e-05 3.89114942e-05 3.84896686e-05 3.76704958e-05
3.74517083e-05 3.64997009e-05 3.60339333e-05 3.59095309e-05
3.54373633e-05 3.49724191e-05 3.45244689e-05 3.37950292e-05
3.36291285e-05 3.27743905e-05 3.25644181e-05 3.21178209e-05
3.19244822e-05 3.10799692e-05 3.09331417e-05 2.94894004e-05
2.91756878e-05 2.83962756e-05 2.81371014e-05 2.79225030e-05
2.77906633e-05 2.67398224e-05 2.59938708e-05 2.56642096e-05
2.50919713e-05 2.43105404e-05 2.37864497e-05 2.30092165e-05
2.25317028e-05 2.23934244e-05 2.16883833e-05 2.15214417e-05
2.10911024e-05 2.08268307e-05 2.03827530e-05 2.00710575e-05
1.99803457e-05 1.99021985e-05 1.94025400e-05 1.89844207e-05
1.86494028e-05 1.85136678e-05 1.82165687e-05 1.81088746e-05
1.76603985e-05 1.66278782e-05 1.57089353e-05 1.54115036e-05
1.51549952e-05 1.50249370e-05 1.47080993e-05 1.45451539e-05
1.43044280e-05 1.38545739e-05 1.33429789e-05 1.31768694e-05
1.29182981e-05 1.27030474e-05 1.24393982e-05 1.24016971e-05
1.21828959e-05 1.14072147e-05 1.13012680e-05 1.06070042e-05
1.02620472e-05 9.56148163e-06 9.52945163e-06 9.10575389e-06
8.98275445e-06 8.75757198e-06 8.55595017e-06 8.35178205e-06
8.12010200e-06 8.09376601e-06 7.90381569e-06 7.82867719e-06
7.68271610e-06 7.34975476e-06 7.21421552e-06 6.98068054e-06
6.75081346e-06 6.37765242e-06 6.05542852e-06 5.97182830e-06
5.62616227e-06 5.56114360e-06 5.46099201e-06 5.19185703e-06
4.99765215e-06 4.96054571e-06 4.88619710e-06 4.86318602e-06
4.73721550e-06 4.70664055e-06 4.35574053e-06 4.22201736e-06
4.12472959e-06 3.91068849e-06 3.82041753e-06 3.55925122e-06
3.44424480e-06 3.42820329e-06 3.29320491e-06 3.20345407e-06
3.13927225e-06 3.09403825e-06 2.95921864e-06 2.87841745e-06
2.81101675e-06 2.77779306e-06 2.72110009e-06 2.68237081e-06
2.46354095e-06 2.40727609e-06 2.38866104e-06 2.36676041e-06
2.33457667e-06 2.31581395e-06 2.24288356e-06 2.23080154e-06
2.21334072e-06 2.04492487e-06 2.03198515e-06 1.96996026e-06
1.81520920e-06 1.72285600e-06 1.70774443e-06 1.67759129e-06
1.63622756e-06 1.62096015e-06 1.44984669e-06 1.42931173e-06
1.39579819e-06 1.32192223e-06 1.20697179e-06 1.16697114e-06
1.09450209e-06 1.06107529e-06 1.03072282e-06 1.01932649e-06
1.00064418e-06 9.86162852e-07 9.67166657e-07 8.87719094e-07
8.47555990e-07 8.19022109e-07 8.01809343e-07 7.83574276e-07
7.22076643e-07 6.80557728e-07 6.63138783e-07 6.23750331e-07
5.49326671e-07 5.45570011e-07 4.88195464e-07 4.81266100e-07
4.55592529e-07 4.43252853e-07 4.40010825e-07 4.34588966e-07
3.84571963e-07 3.78203948e-07 3.57045995e-07 3.50774036e-07
3.32338969e-07 3.23509248e-07 3.14142737e-07 2.89211636e-07
2.39829306e-07 1.91649826e-07 1.81168556e-07 1.77479172e-07
1.71308500e-07 1.60944330e-07 1.49818103e-07 1.24879987e-07
1.24495177e-07 1.20118913e-07 1.14264525e-07 9.87109238e-08
9.52006609e-08 9.30707437e-08 9.08810937e-08 8.58078832e-08
8.45186683e-08 7.19830377e-08 6.35612601e-08 5.81798498e-08
5.26980710e-08 5.22857315e-08 5.19382675e-08 5.08138344e-08
4.81242857e-08 3.90967428e-08 2.14710469e-08 2.04964821e-08
2.01989334e-08 1.99112628e-08 1.93523721e-08 1.84865061e-08
1.38879010e-08 1.34212430e-08 1.09661056e-08 1.03779840e-08
8.46369399e-09 7.36251570e-09 6.46083448e-09 6.22537632e-09
5.68411833e-09 5.41327070e-09 2.41992578e-09 1.29193418e-09
1.28101127e-09 7.61731750e-10 1.46031429e-11 1.20775158e-17
9.11479347e-18 5.38770336e-18 4.01801367e-18 3.50693911e-18
1.73484987e-18 1.62277543e-18 1.41394207e-18 8.90967884e-19
8.52027807e-19 8.11742232e-19 6.53444376e-19 6.24551761e-19
5.66618307e-19 4.31284742e-19 3.25028206e-19 1.93891378e-19
1.33264255e-19 1.16967774e-19 1.13027013e-19 9.27163796e-20
6.91719309e-20 6.16156844e-20 4.68090478e-20 3.74185363e-20
3.17903642e-20 2.88499211e-20 2.47984783e-20 2.15301954e-20
1.72610441e-20 1.01854104e-20 6.60601221e-21 4.90581318e-21
2.31071742e-21 1.01025777e-21 2.95304619e-22 1.65751445e-22
4.39797366e-23 5.40104778e-24 5.51050449e-35 2.36583845e-35
2.00702098e-35 9.40737338e-36 2.64348199e-36 5.09998019e-37
3.76804776e-37 2.33030146e-37 1.45492155e-37 1.41480845e-38
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
```

```
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
```

In [169…
```python
# Step 7: Transform the train and test data into the component basis vector to get the new linear
# combination features.

# Apply PCA transformation to the train and test datasets
x_train_pca = pca.transform(X_train)
x_test_pca = pca.transform(X_test)

# Select only the required components
x_train_pca = x_train_pca[:,:var_comp]
x_test_pca = x_test_pca[:,:var_comp]
```

In [170…
```python
# Step 8: Train SVM Classifier with RBF Kernel
svm = SVC(kernel='rbf')
svm.fit(x_train_pca, y_train)
```

Out[170…
```
▾ SVC
SVC()
```

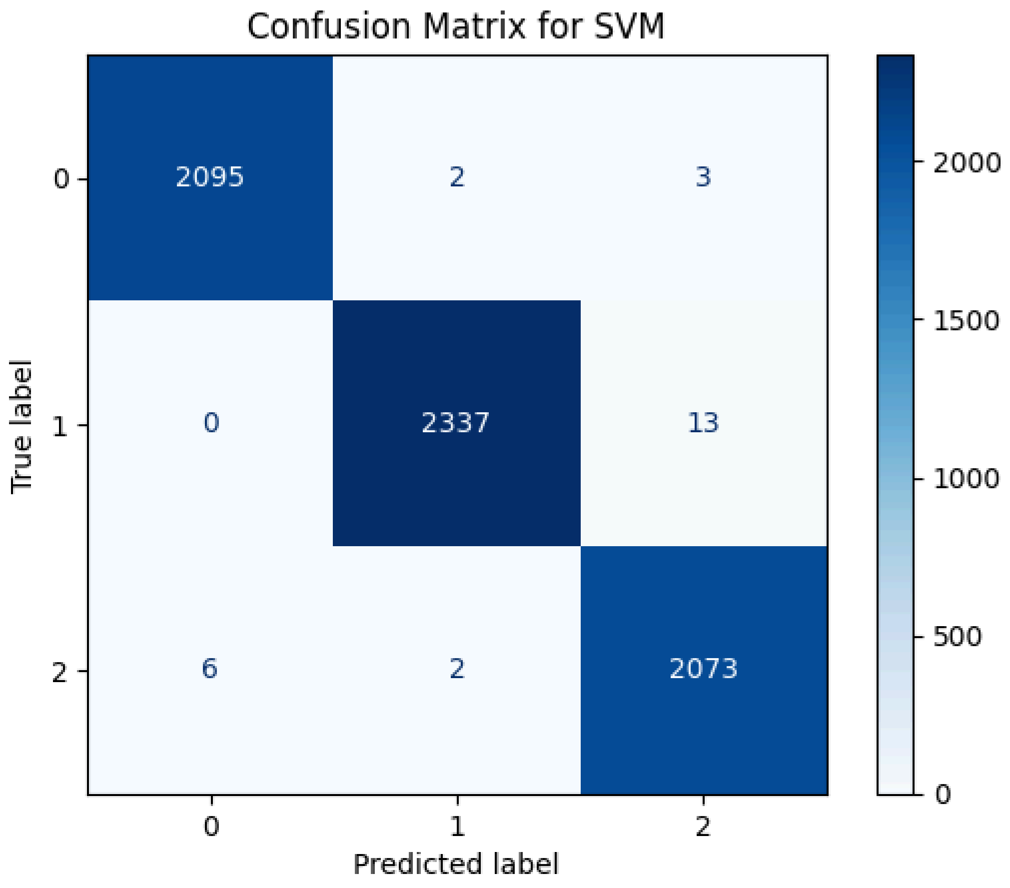
In [171…
```python
# Step 9: Predict and Evaluate
y_pred = svm.predict(x_test_pca)
print("Classification Report (SVM):")
print(classification_report(y_test, y_pred))

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix for SVM")
plt.show()
```

```
Classification Report (SVM):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2100
           1       1.00      0.99      1.00      2350
           2       0.99      1.00      0.99      2081

    accuracy                           1.00      6531
   macro avg       1.00      1.00      1.00      6531
weighted avg       1.00      1.00      1.00      6531
```



In [172…
```python
# Step 10: Pipeline (PCA + SVM) with custom PCAVarianceTreshold class to capture the components required.
class  PCAVarianceThreshold(BaseEstimator, TransformerMixin):
    def __init__(self, threshold=0.95):
        self.threshold = threshold

    def fit(self, X, y=None):
        # Fit PCA to determine the number of components
        self.pca_ = PCA(n_components=min(X.shape))
        self.pca_.fit(X)

        # Calculate the cumulative variance and select the number of components for the threshold
        cumsum = np.cumsum(self.pca_.explained_variance_ratio_)
        self.n_components_ = np.where(cumsum >= self.threshold)[0][0]

        return self

    def transform(self, X):
        # Use the selected number of components to transform the data
        return self.pca_.transform(X)[:, :self.n_components_]
```

```
    def fit_transform(self, X, y=None):
        return self.fit(X, y).transform(X)

pipeline = Pipeline([
    ('pca', PCAVarianceThreshold(var_thr)),  # PCA first
    ('svm', SVC(kernel='rbf'))  # Then SVM
])
pipeline.fit(X_train, y_train)

# Predict using pipeline (automatically applies PCA)
y_pred_pipe = pipeline.predict(X_test)

# Final Classification Report
print("Classification Report (Pipeline SVM):")
print(classification_report(y_test, y_pred_pipe))

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, y_pred_pipe)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix for Pipeline SVM")
plt.show()
```

```
Classification Report (Pipeline SVM):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2100
           1       1.00      0.99      1.00      2350
           2       0.99      1.00      0.99      2081

    accuracy                           1.00      6531
   macro avg       1.00      1.00      1.00      6531
weighted avg       1.00      1.00      1.00      6531
```