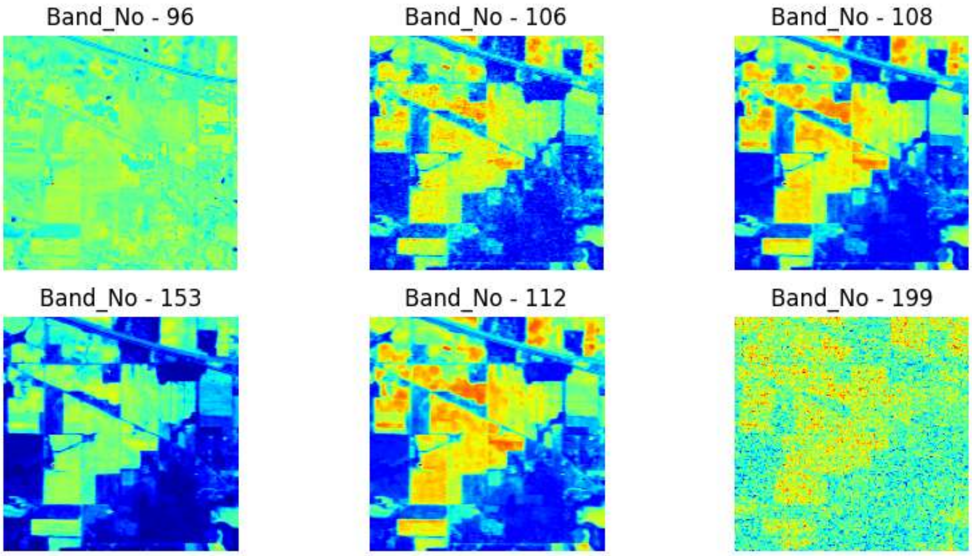
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.io import loadmat

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, mean_absolute_error
```

```
X = loadmat("/Indian_pines_corrected.mat")["indian_pines_corrected"]
y = loadmat("/Indian_pines_gt.mat")["indian_pines_gt"]
```

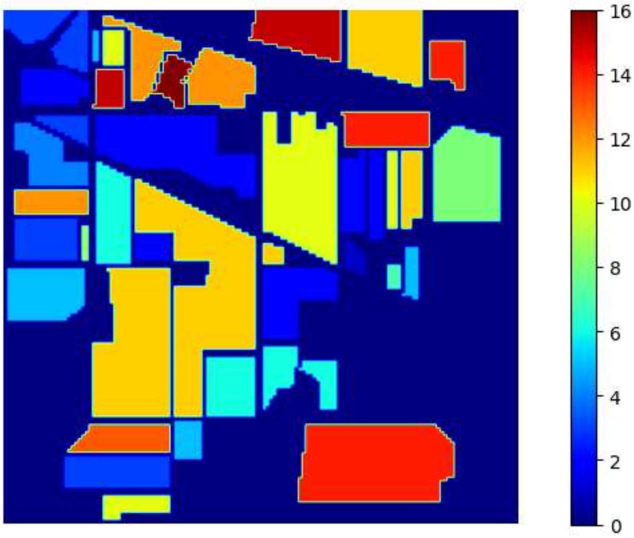
```
#shape of the dataset
print(f"Indian_Pines: {X.shape} \nGround_Truth: {y.shape}")
```

```
Indian_Pines: (145, 145, 200)
Ground_Truth: (145, 145)
```

```
#visualizing the bands
fig = plt.figure(figsize = (10, 5))

for i in range(1, 7):
    fig.add_subplot(2,3, i)
    q = np.random.randint(X.shape[2])
    plt.imshow(X[:,:,q], cmap = "jet")
    plt.axis("off")
    plt.title(f"Band_No - {q}")
```




```
#visualizing of groundtruth
plt.figure(figsize = (10, 5))
plt.imshow(y, cmap = "jet")
plt.axis("off")
plt.colorbar()
plt.show()
```




```
#extracting the pixels
def extract_pixels(X, y):

  data = X.reshape(-1, X.shape[2])
  pines = pd.DataFrame(data = data)
  pines = pd.concat([pines, pd.DataFrame(data = y.ravel())], axis = 1)
  pines.columns= [f"band{i}" for i in range(1, 1+X.shape[2])] + ["class"]
  pines.to_csv("Dataset.csv")
  return pines

pines = extract_pixels(X, y)
```

```
pines.head()
```

	band1	band2	band3	band4	band5	band6	band7	band8	band9	band10	...	band192	band193	band194	band195	band196	band197	b
0	3172	4142	4506	4279	4782	5048	5213	5106	5053	4750	...	1094	1090	1112	1090	1062	1069	
1	2580	4266	4502	4426	4853	5249	5352	5353	5347	5065	...	1108	1104	1117	1091	1079	1085	
2	3687	4266	4421	4498	5019	5293	5438	5427	5383	5132	...	1111	1114	1114	1100	1065	1092	
3	2749	4258	4603	4493	4958	5234	5417	5355	5349	5096	...	1122	1108	1109	1109	1071	1088	
4	2746	4018	4675	4417	4886	5117	5215	5096	5098	4834	...	1110	1107	1112	1094	1072	1087	

```
pines.shape
```

```
(21025, 201)
```

```
pines.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21025 entries, 0 to 21024
Columns: 201 entries, band1 to class
dtypes: uint16(200), uint8(1)
memory usage: 8.0 MB
```

```
pines.isnull().sum()
```

```
band1      0
band2      0
band3      0
band4      0
band5      0
          ..
band197    0
band198    0
band199    0
band200    0
class      0
Length: 201, dtype: int64
```

```
pines.duplicated().sum()
```

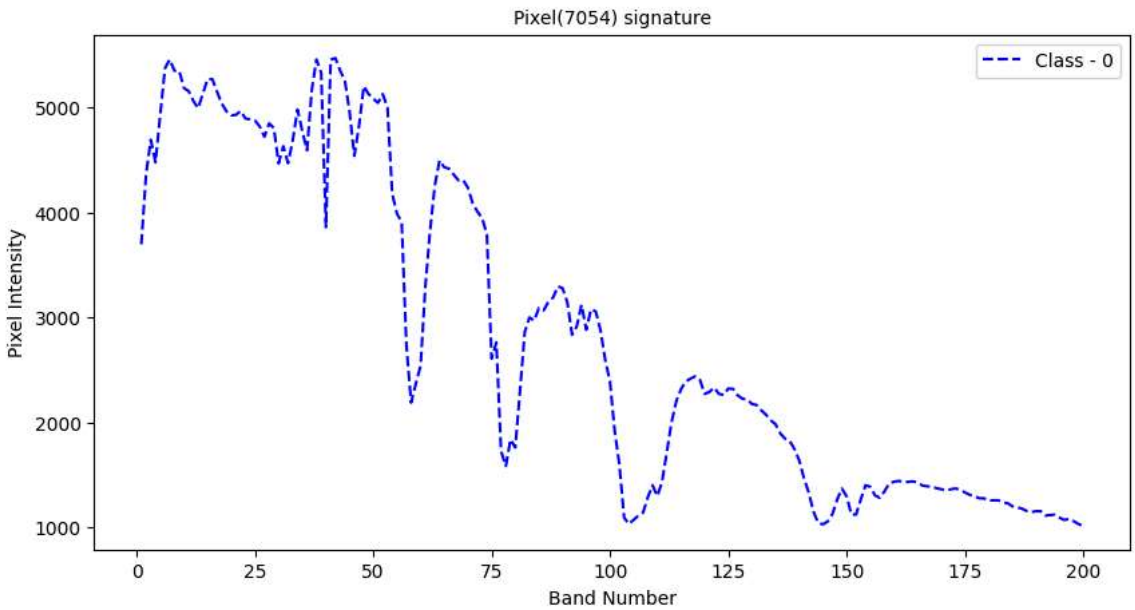
```
0
```

```
pines.describe()
```

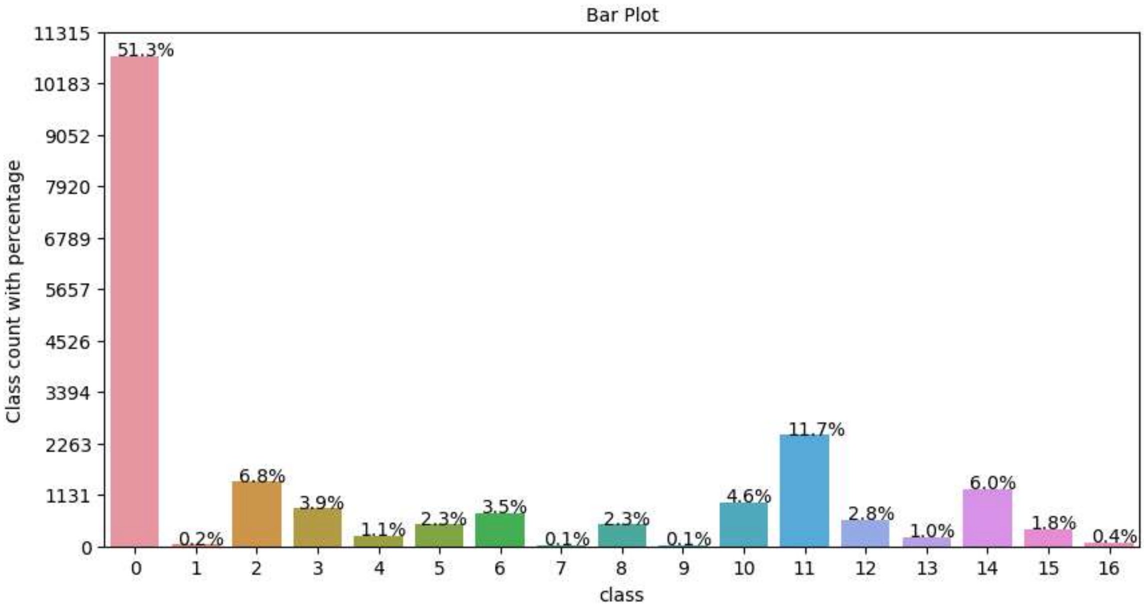
	band1	band2	band3	band4	band5	band6	band7	band8	band9	
count	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21(
mean	2957.363472	4091.321237	4277.502259	4169.956671	4516.678668	4790.595149	4848.317574	4714.732509	4668.904828	4
std	354.918708	230.390005	257.827640	280.761254	346.035984	414.382138	469.247667	491.728349	533.232855	[illegible]
min	2560.000000	2709.000000	3649.000000	2810.000000	3840.000000	4056.000000	4004.000000	3865.000000	3775.000000	2
25%	2602.000000	3889.000000	4066.000000	3954.000000	4214.000000	4425.000000	4421.000000	4263.000000	4173.000000	3
50%	2780.000000	4106.000000	4237.000000	4126.000000	4478.000000	4754.000000	4808.000000	4666.000000	4632.000000	4
75%	3179.000000	4247.000000	4479.000000	4350.000000	4772.000000	5093.000000	5198.000000	5100.000000	5084.000000	4
max	4536.000000	5744.000000	6361.000000	6362.000000	7153.000000	7980.000000	8284.000000	8128.000000	8194.000000	7

8 rows × 201 columns

```
#visualizing spectral signatures

def plot_signature(pines):
    plt.figure(figsize = (10, 5))
    pixel_no = np.random.randint(pines.shape[0])
    plt.plot(range(1, 201), pines.iloc[pixel_no, :-1].values.tolist(), "b--", label = f"Class - {pines.iloc[pixel_no, -1]}")
    plt.legend()
    plt.title(f"Pixel({pixel_no}) signature", fontsize = 10)
    plt.xlabel("Band Number", fontsize = 10)
    plt.ylabel("Pixel Intensity", fontsize = 10)
    plt.show()

plot_signature(pines)
```

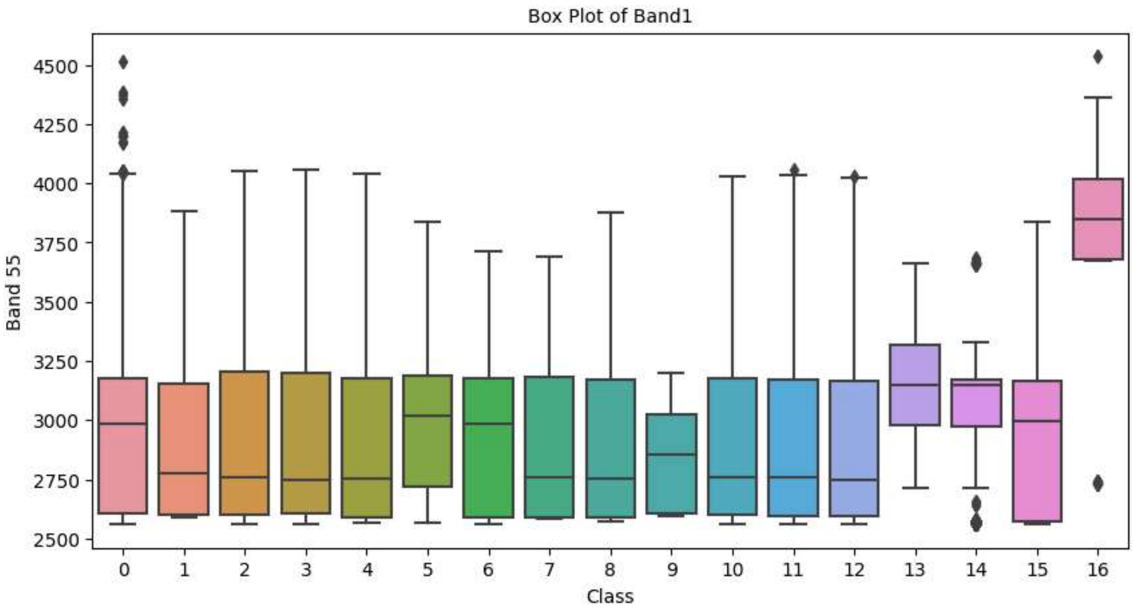



```
import matplotlib.ticker as ticker

plt.figure(figsize = (10, 5))

ax = sns.countplot(x = "class", data = pines[["class"]])

for p in ax.patches:
        ax.annotate('{:.1f}%'.format(100*p.get_height()/pines.shape[0]), (p.get_x()+0.1, p.get_height()+5))

        ax.yaxis.set_major_locator(ticker.LinearLocator(11))


plt.xlabel("class", fontsize = 10)
plt.ylabel("Class count with percentage", fontsize = 10)
plt.title("Bar Plot", fontsize = 10)
plt.show()
```




```
#box plot bands

n = int(input("Enter the band Number(1-200)"))

plt.figure(figsize = (10, 5))
sns.boxplot(x = pines["class"], y = pines["band1"]);
plt.title("Box Plot of Band1", fontsize = 10)
plt.xlabel("Class", fontsize = 10)
plt.ylabel(f"Band {n}", fontsize = 10)
plt.show()
```

```
Enter the band Number(1-200)55
```

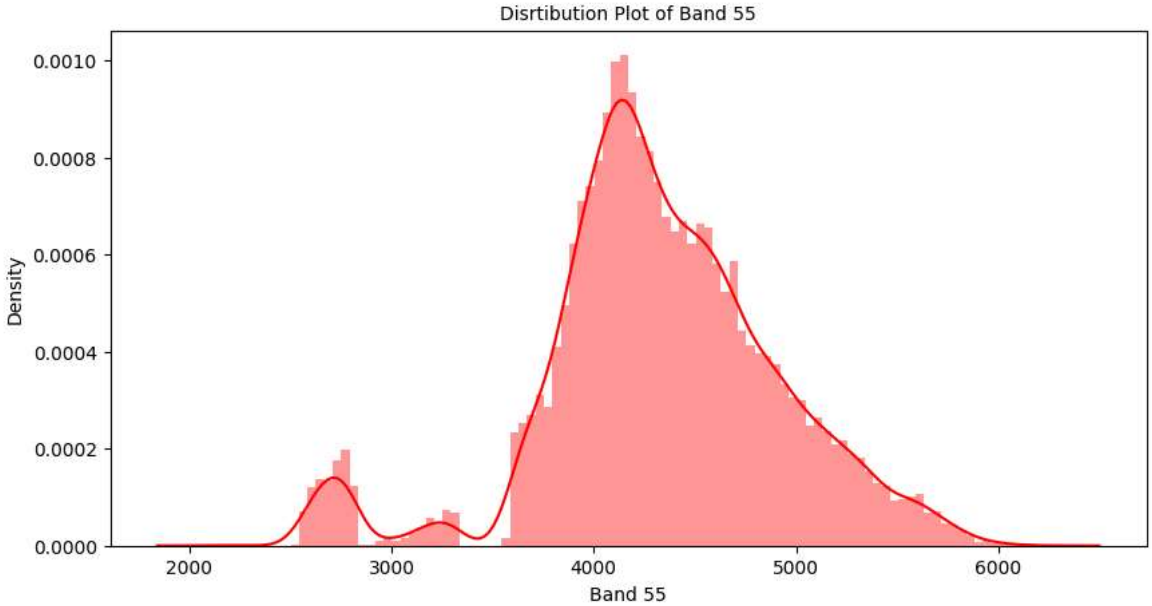



```
#band details

print(f"Details of Band - {n}: \n\n{pines['band'+str(n)].describe()}")
```

```
Details of Band - 55:

count    21025.00000
mean      4333.39396
std        591.08510
min       2084.00000
25%       4021.00000
50%       4297.00000
75%       4685.00000
max       6256.00000
Name: band55, dtype: float64
```

```
#distribution plot of band55

plt.figure(figsize = (10, 5))
sns.distplot(pines["band"+str(n)], color = "red", bins = 100, hist_kws = {"alpha": 0.4});
plt.xlabel("Band " +str(n), fontsize = 10)
plt.title("Disrtibution Plot of Band " +str(n), fontsize = 10)
plt.show()
```

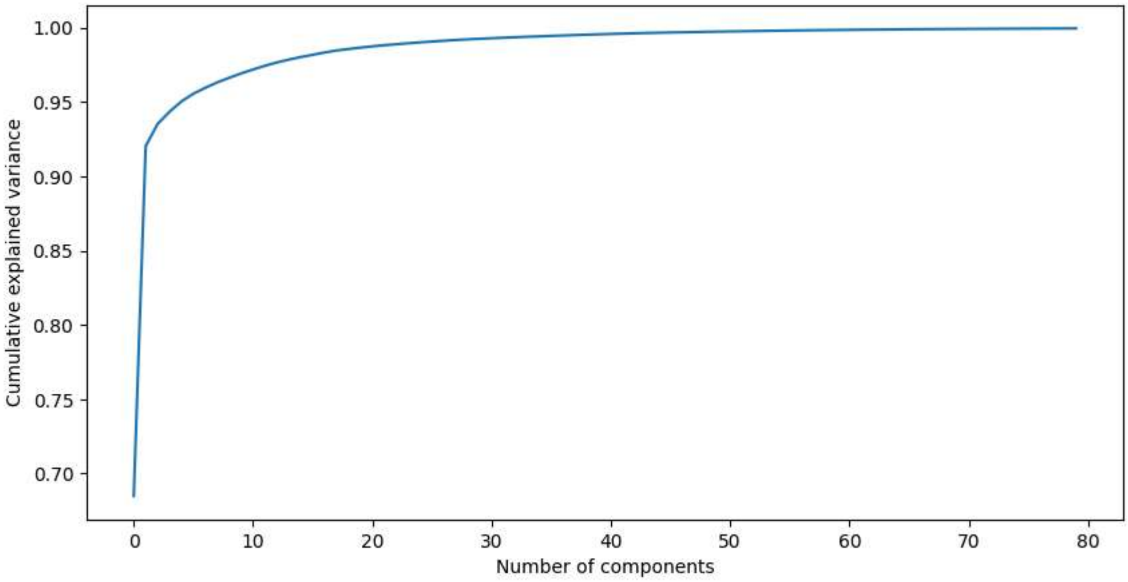
```
<ipython-input-17-c99157d3cb19>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(pines["band"+str(n)], color = "red", bins = 100, hist_kws = {"alpha": 0.4});
```




```
#Principal Component Analysis(PCA)

from sklearn.decomposition import PCA

pca = PCA(n_components = 80)
principalComponents = pca.fit_transform(pines)
ev = pca.explained_variance_ratio_
ev
```

```
array([6.84937528e-01, 2.35313543e-01, 1.49635396e-02, 8.21543227e-03,
       6.95012750e-03, 5.17010701e-03, 3.99681154e-03, 3.62359908e-03,
       3.07127269e-03, 2.93211761e-03, 2.67352834e-03, 2.49229944e-03,
       2.24688212e-03, 1.89388676e-03, 1.69434305e-03, 1.56043702e-03,
       1.53162388e-03, 1.35012957e-03, 1.00138965e-03, 9.24874694e-04,
       8.47884121e-04, 7.64385411e-04, 6.64597007e-04, 6.45680426e-04,
       6.16360583e-04, 5.61408927e-04, 5.43160665e-04, 5.15585128e-04,
       4.21073623e-04, 3.65029748e-04, 3.62711009e-04, 3.53239515e-04,
       3.24037211e-04, 3.13691891e-04, 3.03385418e-04, 2.87733751e-04,
       2.79164296e-04, 2.72731345e-04, 2.62985400e-04, 2.50311312e-04,
       2.46112535e-04, 2.32228734e-04, 2.11368775e-04, 1.94079618e-04,
       1.81978323e-04, 1.70834583e-04, 1.55749869e-04, 1.41898394e-04,
       1.37335867e-04, 1.36430858e-04, 1.33485423e-04, 1.23374680e-04,
       1.21877860e-04, 1.20991207e-04, 1.14749909e-04, 1.13124570e-04,
       1.04952992e-04, 1.02963502e-04, 9.31343947e-05, 8.91228226e-05,
       8.49670157e-05, 8.41797452e-05, 7.60502138e-05, 7.02624120e-05,
       6.77546758e-05, 6.28505367e-05, 6.23339259e-05, 6.03196039e-05,
       5.56434115e-05, 5.29588809e-05, 5.09735498e-05, 4.56128071e-05,
       4.20708046e-05, 4.14508367e-05, 3.92575510e-05, 3.85389786e-05,
       3.74666798e-05, 3.64689847e-05, 3.57606693e-05, 3.46685538e-05])
```

```
plt.figure(figsize=(10, 5))
plt.plot(np.cumsum(ev))
plt.xlabel("Number of components")
plt.ylabel("Cumulative explained variance")
plt.show()
```

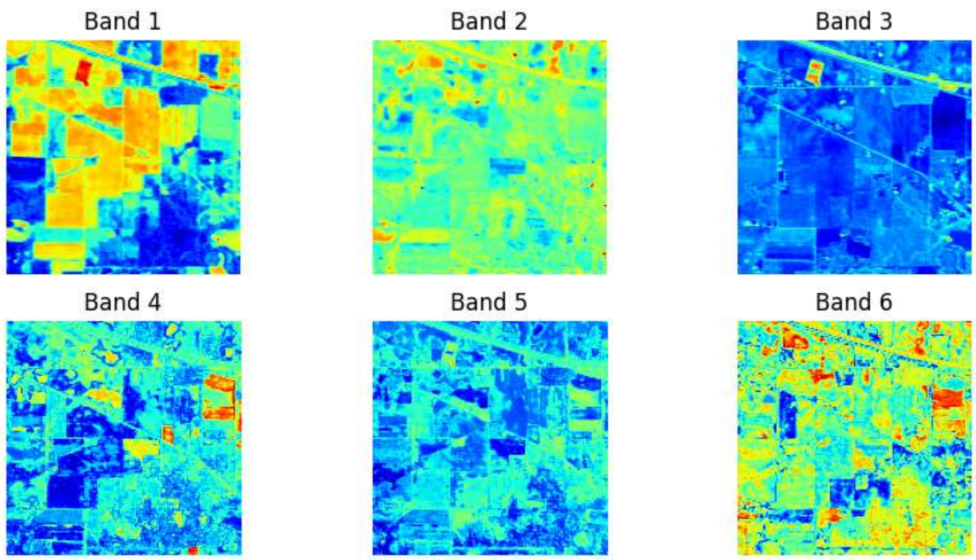



```
#select 40 components for PCA

pca = PCA(n_components = 40)
data = pca.fit_transform(pines)
p_data = pd.concat([pd.DataFrame(data = data), pd.DataFrame(data = y.ravel())], axis = 1)
p_data.columns = [f"PC-{i}" for i in range(1,41)] + ["class"]
p_data.head()
```

	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC-7	PC-8	PC-9	PC-10	...	
0	5014.905666	1456.863532	72.697658	71.201107	-435.684655	-68.843392	134.809841	-304.368847	256.430470	-66.626401	...	1
1	5601.383449	-2023.449776	350.135434	-528.457143	148.103477	-288.362818	202.955984	240.853090	-474.858947	93.492081	...	9
2	5796.135157	-3090.394530	490.540544	-760.205251	259.951266	-131.614633	172.926575	205.913482	572.491544	-191.616243	...	3
3	5586.204284	-2369.375772	356.275521	-502.679332	146.569636	-306.682882	251.070325	234.970823	-314.024411	54.961267	...	12
4	5020.990484	339.603668	-23.006921	-92.558409	-368.488742	-438.269715	502.715682	-345.532591	-188.355243	-67.505651	...	-8

5 rows × 41 columns

```
#plotting the bands after pca

fig = plt.figure(figsize = (10, 5))

for i in range(1, 7):
    fig.add_subplot(2,3, i)
    plt.imshow(p_data.loc[:, f"PC-{i}"].values.reshape(145, 145), cmap = "jet")
    plt.axis("off")
    plt.title(f"Band {i}")
```



```
# saving to .csv
p_data.to_csv("Pines_PCA.csv", index=False)


x = p_data[p_data["class"] != 0]

X = x.iloc[:, :-1].values

y = x.loc[:, "class"].values

names = ["Alfalfa", "Corn-notill", "Corn-mintill", "Corn", "Grass-pasture", "Grass-trees", "Grass-pasture-mowed", "Hay-windrowed", "Oat:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 11, stratify = y)
svm =  SVC(C = 100, kernel = "rbf", cache_size = 10*1024)
svm.fit(X_train, y_train)
```

```
        SVC
SVC(C=100, cache_size=10240)
```

```
X_train_prediction = svm.predict(X_train)
training_data_accuracy = accuracy_score(y_train, X_train_prediction)


print("Accuracy score of the training data is: ", training_data_accuracy)
```

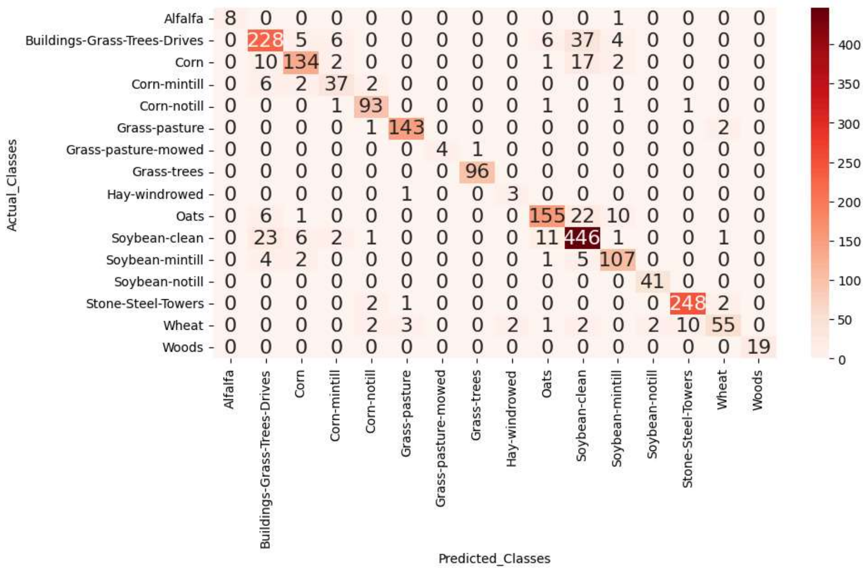
```
Accuracy score of the training data is:  0.9224295645810465
```

```
y_prediction = svm.predict(X_test)
test_data_accuracy = accuracy_score(y_prediction, y_test)
print("Accuracy score of the test data is: ", test_data_accuracy)
```

```
Accuracy score of the test data is:  0.8863414634146342
```

```
c_matrix = confusion_matrix(y_test, y_prediction)

data_cmatrix = pd.DataFrame(c_matrix, columns = np.unique(names), index = np.unique(names))

data_cmatrix.index.name = "Actual_Classes"
data_cmatrix.columns.name = "Predicted_Classes"
plt.figure(figsize = (10,5))
#sns.set(font_scale = 1.4) #for label size
sns.heatmap(data_cmatrix, cmap = "Reds", annot = True, annot_kws = {"size": 16}, fmt = "d")
```

```
<Axes: xlabel='Predicted_Classes', ylabel='Actual_Classes'>
```

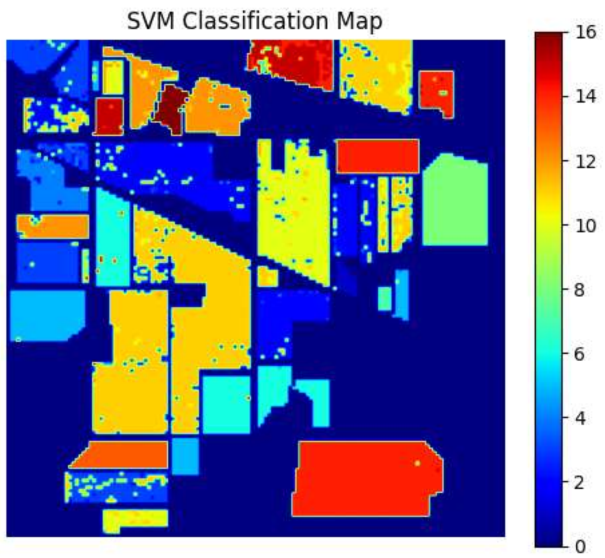



```
print("Classification Report", classification_report(y_test, y_prediction, target_names = names))
```

```
Classification Report                          precision    recall  f1-score   support

                     Alfalfa       1.00      0.89      0.94         9
                 Corn-notill       0.82      0.80      0.81       286
                Corn-mintill       0.89      0.81      0.85       166
                        Corn       0.77      0.79      0.78        47
               Grass-pasture       0.92      0.96      0.94        97
                 Grass-trees       0.97      0.98      0.97       146
         Grass-pasture-mowed       1.00      0.80      0.89         5
               Hay-windrowed       0.99      1.00      0.99        96
                        Oats       0.60      0.75      0.67         4
              Soybean-notill       0.88      0.80      0.84       194
             Soybean-mintill       0.84      0.91      0.87       491
               Soybean-clean       0.85      0.90      0.87       119
                       Wheat       0.95      1.00      0.98        41
                       Woods       0.96      0.98      0.97       253
Buildings-Grass-Trees-Drives       0.92      0.71      0.80        77
          Stone-Steel-Towers       1.00      1.00      1.00        19

                    accuracy                           0.89      2050
                   macro avg       0.90      0.88      0.89      2050
                weighted avg       0.89      0.89      0.89      2050
```

```
l =[]

for i in range(p_data.shape[0]):
    if p_data.iloc[i, -1] == 0:
        l.append(0)
    else:
        l.append(svm.predict(p_data.iloc[i, :-1].values.reshape(1, -1)))


plt.figure(figsize = (6, 5))
clmap = np.array(l).reshape(145, 145).astype("float")
plt.imshow(clmap, cmap = "jet")
plt.colorbar()
plt.axis("off")
plt.title("SVM Classification Map")
plt.savefig("svm_classification_map.png")
plt.show()
```

```
<ipython-input-30-7a65eac540c6>:2: VisibleDeprecationWarning: Creating an ndarray fro
  clmap = np.array(l).reshape(145, 145).astype("float")
```




```
import math
MSE = np.square(np.subtract(y_test,y_prediction)).mean()
RMSE = math.sqrt(MSE)
print("Root Mean Square Error:\n", RMSE)
```

```
Root Mean Square Error:
 2.196671539775243
```

```
print("Mean Absolute Error (MAE)", mean_absolute_error(y_test,y_prediction))
```

```
Mean Absolute Error (MAE) 13.562926829268292
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.