Hyperspectral Image Analysis-Classification

Import Libraries

In [4]:
```
import plotly.express as px
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, conf
usion_matrix
import seaborn as sn
```

Download HSI Data

In [5]:
```
!wget http://www.ehu.eus/ccwintco/uploads/6/67/Indian_pines_corrected.m
at http://www.ehu.eus/ccwintco/uploads/c/c4/Indian_pines_gt.mat
```

```
--2021-05-24 16:00:02--  http://www.ehu.eus/ccwintco/uploads/6/67/India
n_pines_corrected.mat
Resolving www.ehu.eus (www.ehu.eus)... 158.227.0.65, 2001:720:1410::65
Connecting to www.ehu.eus (www.ehu.eus)|158.227.0.65|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5953527 (5.7M)
Saving to: ‘Indian_pines_corrected.mat’

Indian_pines_correc 100%[===================>]   5.68M   591KB/s    in
11s

2021-05-24 16:00:13 (540 KB/s) - ‘Indian_pines_corrected.mat’ saved [59
```

```
53527/5953527]

--2021-05-24 16:00:13--  http://www.ehu.eus/ccwintco/uploads/c/c4/India
n_pines_gt.mat
Connecting to www.ehu.eus (www.ehu.eus)|158.227.0.65|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1125 (1.1K)
Saving to: ‘Indian_pines_gt.mat’

Indian_pines_gt.mat 100%[===================>]   1.10K  --.-KB/s    in
0s

2021-05-24 16:00:13 (182 MB/s) - ‘Indian_pines_gt.mat’ saved [1125/112
5]

FINISHED --2021-05-24 16:00:13--
Total wall clock time: 12s
Downloaded: 2 files, 5.7M in 11s (540 KB/s)
```

In [6]:
```
!ls
```

```
Indian_pines_corrected.mat  Indian_pines_gt.mat  sample_data
```

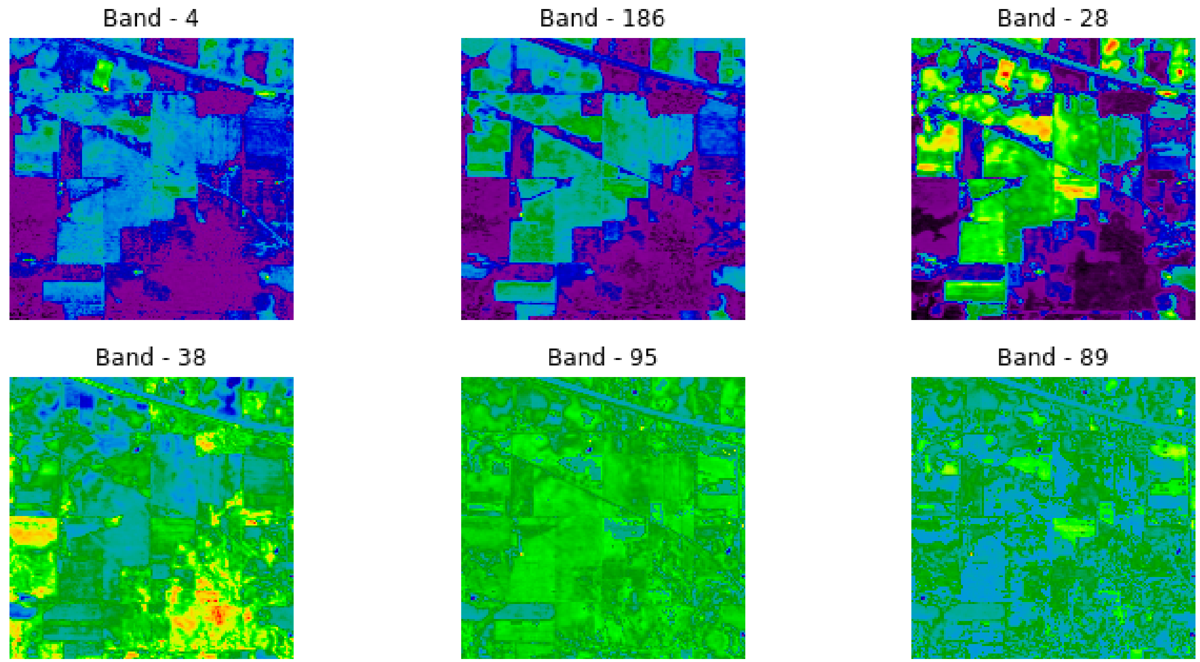
Read the Data

In [7]:
```
from scipy.io import loadmat

def read_HSI():
  X = loadmat('Indian_pines_corrected.mat')['indian_pines_corrected']
  y = loadmat('Indian_pines_gt.mat')['indian_pines_gt']
  print(f"X shape: {X.shape}\ny shape: {y.shape}")
  return X, y

X, y = read_HSI()
```

```
X shape: (145, 145, 200)
y shape: (145, 145)
```

Visualize Bands

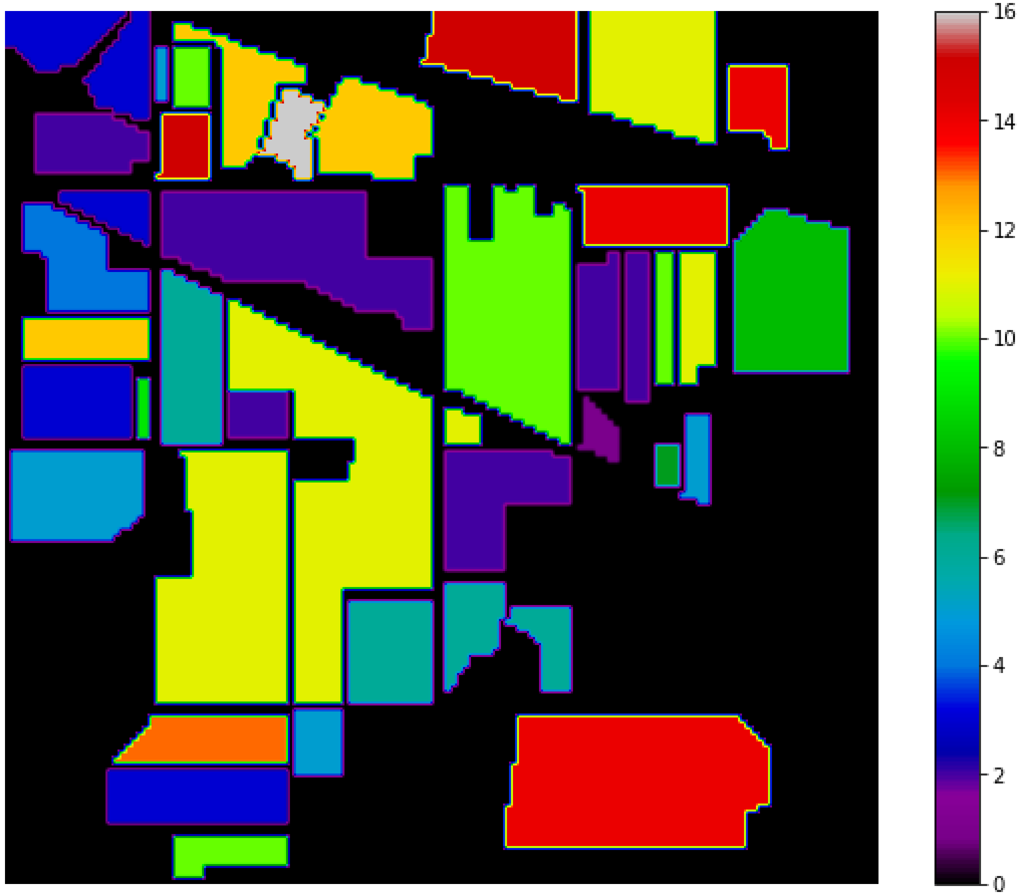
In [8]:
```
fig = plt.figure(figsize = (12, 6))

for i in range(1, 1+6):
    fig.add_subplot(2,3, i)
    q = np.random.randint(X.shape[2])
    plt.imshow(X[:,:,q], cmap='nipy_spectral')
    plt.axis('off')
    plt.title(f'Band - {q}')
plt.savefig('IP_Bands.png')
```



Visualize the Ground Truth

In [9]:
```
plt.figure(figsize=(10, 8))
```

```
plt.imshow(y, cmap='nipy_spectral')
plt.colorbar()
plt.axis('off')
plt.savefig('IP_GT.png')
plt.show()
```



Convert the dataset into csv

In [10]:
```
import pandas as pd
import numpy as np
```

```
def extract_pixels(X, y):
  q = X.reshape(-1, X.shape[2])
  df = pd.DataFrame(data = q)
  df = pd.concat([df, pd.DataFrame(data = y.ravel())], axis=1)
  df.columns= [f'band{i}' for i in range(1, 1+X.shape[2])]+['class']
  df.to_csv('Dataset.csv')
  return df

df = extract_pixels(X, y)
```

In [11]:
```
df.head()
```

Out[11]:

	band1	band2	band3	band4	band5	band6	band7	band8	band9	band10	band11	band12
0	3172	4142	4506	4279	4782	5048	5213	5106	5053	4750	4816	4769
1	2580	4266	4502	4426	4853	5249	5352	5353	5347	5065	5141	5100
2	3687	4266	4421	4498	5019	5293	5438	5427	5383	5132	5227	5172
3	2749	4258	4603	4493	4958	5234	5417	5355	5349	5096	5147	5078
4	2746	4018	4675	4417	4886	5117	5215	5096	5098	4834	4853	4857

5 rows × 201 columns

In [12]:
```
df.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
```

In [13]:
```
df.iloc[:, :-1].describe()
```

Out[13]:

	band1	band2	band3	band4	band5	band6

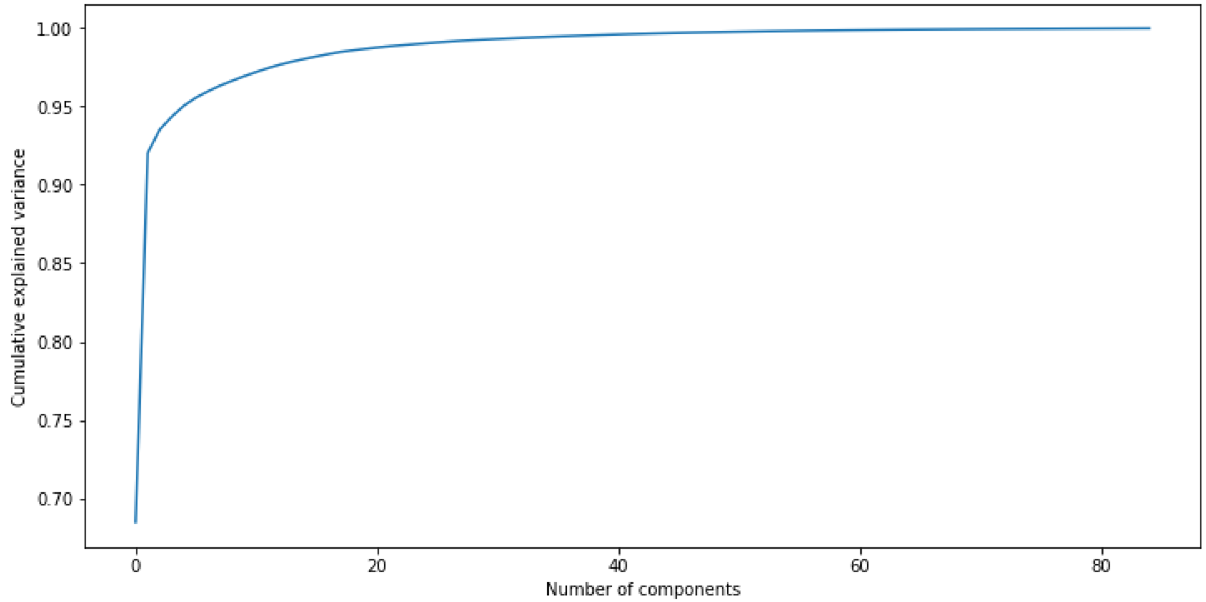
	band1	band2	band3	band4	band5	band6	
count	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	2102
mean	2957.363472	4091.321237	4277.502259	4169.956671	4516.678668	4790.595149	484
std	354.918708	230.390005	257.827640	280.761254	346.035984	414.382138	46
min	2560.000000	2709.000000	3649.000000	2810.000000	3840.000000	4056.000000	40(
25%	2602.000000	3889.000000	4066.000000	3954.000000	4214.000000	4425.000000	442
50%	2780.000000	4106.000000	4237.000000	4126.000000	4478.000000	4754.000000	48(
75%	3179.000000	4247.000000	4479.000000	4350.000000	4772.000000	5093.000000	51٤
max	4536.000000	5744.000000	6361.000000	6362.000000	7153.000000	7980.000000	82٤

8 rows × 200 columns

Principal Component Analysis(PCA)

In [14]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 85)

principalComponents = pca.fit_transform(df.iloc[:, :-1].values)

ev=pca.explained_variance_ratio_

plt.figure(figsize=(12, 6))
plt.plot(np.cumsum(ev))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')


plt.show()
```



Select 40 as the no.of components for PCA

In [15]:
```
pca = PCA(n_components = 40)
dt = pca.fit_transform(df.iloc[:, :-1].values)
q = pd.concat([pd.DataFrame(data = dt), pd.DataFrame(data = y.ravel
())], axis = 1)
q.columns = [f'PC-{i}' for i in range(1,41)]+['class']
```

In [16]:
```
q.head()
```

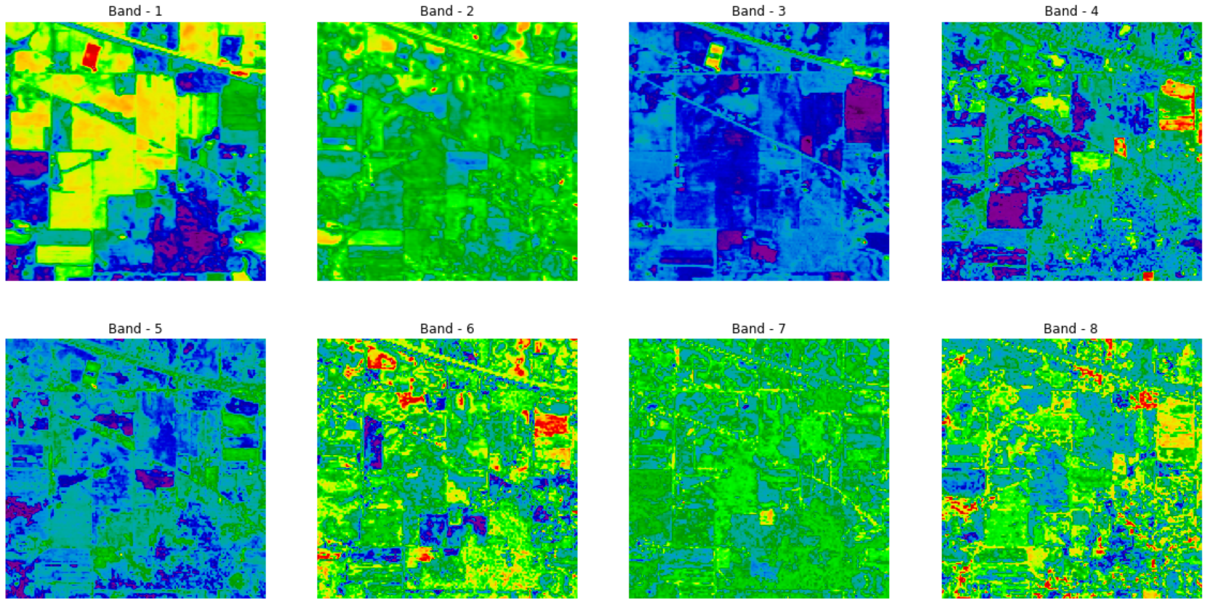
Out[16]:

	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC-7	
0	5014.905985	1456.863260	72.697049	71.204933	-435.686985	-68.840299	134.809886	-304
1	5601.383743	-2023.450087	350.134661	-528.465052	148.088297	-288.359029	202.956855	240
2	5796.135442	-3090.394852	490.539929	-760.214343	259.933300	-131.611182	172.927304	205
3	5586.204575	-2369.376085	356.274720	-502.687158	146.554957	-306.679338	251.071102	234

	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC-7	
4	5020.990792	339.603390	-23.007524	-92.556772	-368.495435	-438.266712	502.715438	-345

Display the bands after PCA

In [17]:
```
fig = plt.figure(figsize = (20, 10))

for i in range(1, 1+8):
    fig.add_subplot(2,4, i)
    plt.imshow(q.loc[:, f'PC-{i}'].values.reshape(145, 145), cmap='nipy
_spectral')
    plt.axis('off')
    plt.title(f'Band - {i}')

plt.savefig('IP_PCA_Bands.png')
```



In [18]:
```
# saving to .csv
```

```
q.to_csv('IP_40_PCA.csv', index=False)
```

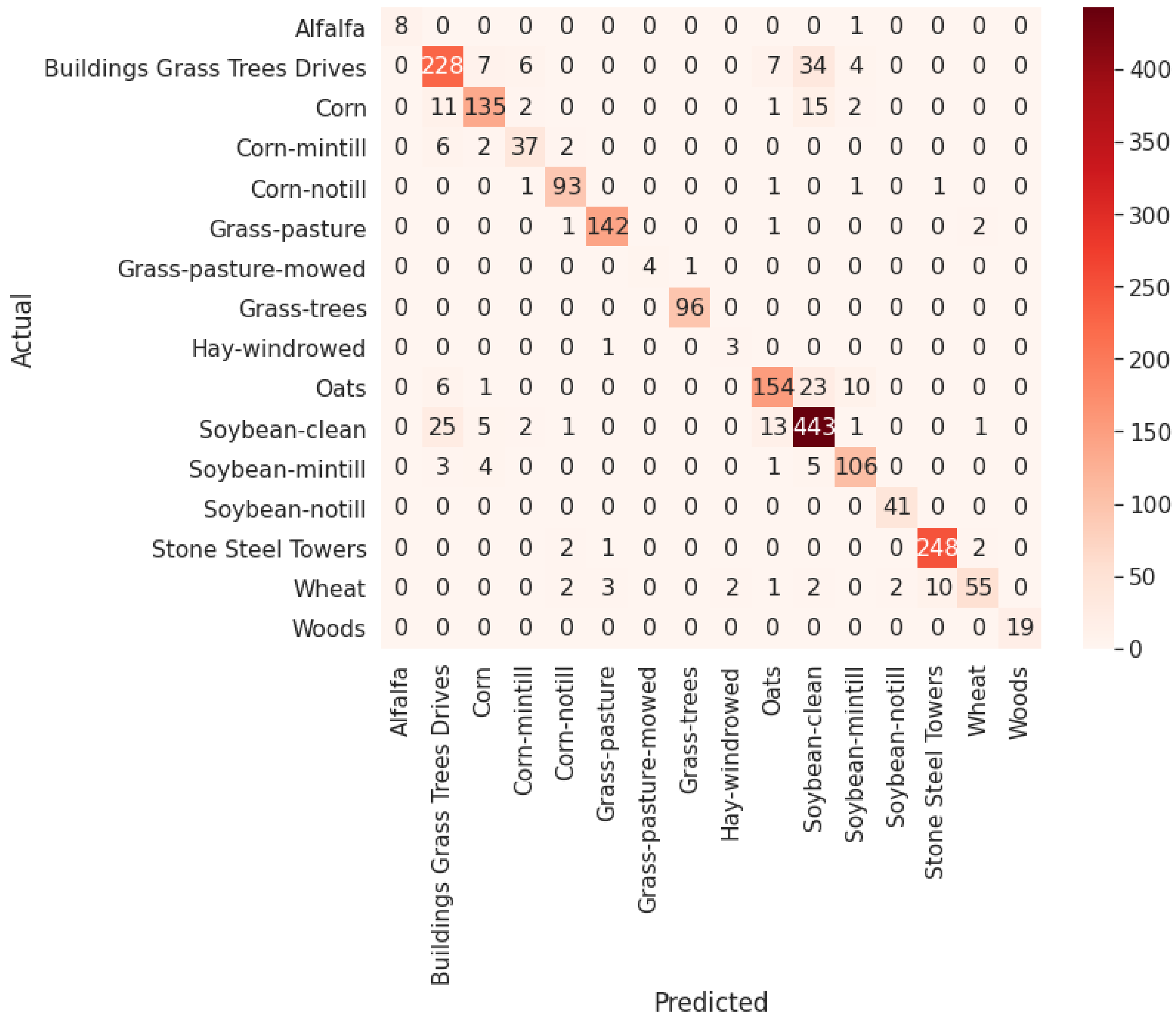
Support Vector Machine(SVM)

In [19]:

```
x = q[q['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','Oats','Soybean-notill','Soybean-
mintill',
'Soybean-clean', 'Wheat',       'Woods',        'Buildings Grass Trees
 Drives',       'Stone Steel Towers']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
0, random_state=11, stratify=y)

svm =  SVC(C = 100, kernel = 'rbf', cache_size = 10*1024)

svm.fit(X_train, y_train)

ypred = svm.predict(X_test)
```

In [20]:

```
data = confusion_matrix(y_test, ypred)
df_cm = pd.DataFrame(data, columns=np.unique(names), index = np.unique(
names))
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
plt.figure(figsize = (10,8))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, cmap="Reds", annot=True,annot_kws={"size": 16}, fmt=
'd')
plt.savefig('cmap.png', dpi=300)
```



In [21]:
```
print(classification_report(y_test, ypred, target_names = names))
```

```
              precision    recall  f1-score   support

     Alfalfa       1.00      0.89      0.94         9
```

```
              Corn-notill       0.82      0.80      0.81       286
             Corn-mintill       0.88      0.81      0.84       166
                     Corn       0.77      0.79      0.78        47
            Grass-pasture       0.92      0.96      0.94        97
              Grass-trees       0.97      0.97      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       0.86      0.79      0.83       194
          Soybean-mintill       0.85      0.90      0.87       491
            Soybean-clean       0.85      0.89      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                           0.88      2050
                macro avg       0.90      0.88      0.88      2050
             weighted avg       0.88      0.88      0.88      2050
```

Classification Map

In [22]:

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

In [23]:

```
clmap = np.array(l).reshape(145, 145).astype('float')
plt.figure(figsize=(10, 8))
plt.imshow(clmap, cmap='nipy_spectral')
plt.colorbar()
plt.axis('off')
plt.savefig('IP_cmap.png')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: Visible
DeprecationWarning:

Creating an ndarray from ragged nested sequences (which is a list-or-tu
ple of lists-or-tuples-or ndarrays with different lengths or shapes) is
deprecated. If you meant to do this, you must specify 'dtype=object' wh
en creating the ndarray
```