```
#pip install earthpy
```

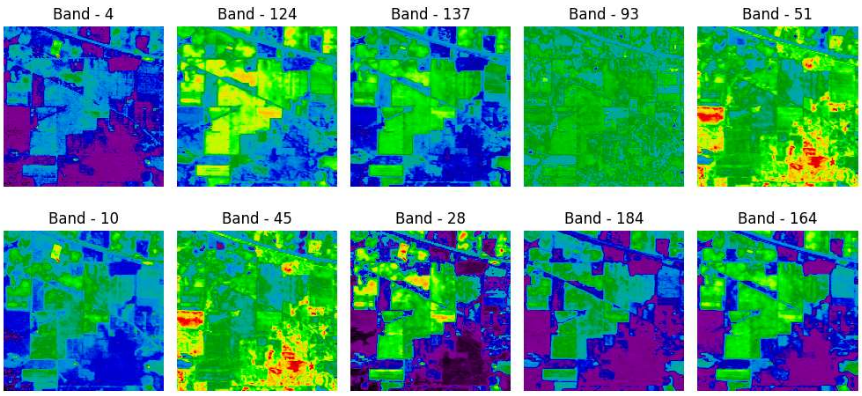
```
#importing libraries

import numpy as np
import pandas as pd
import earthpy.plot as ep
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.io import loadmat

from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix,  mean_absolute_error
```

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

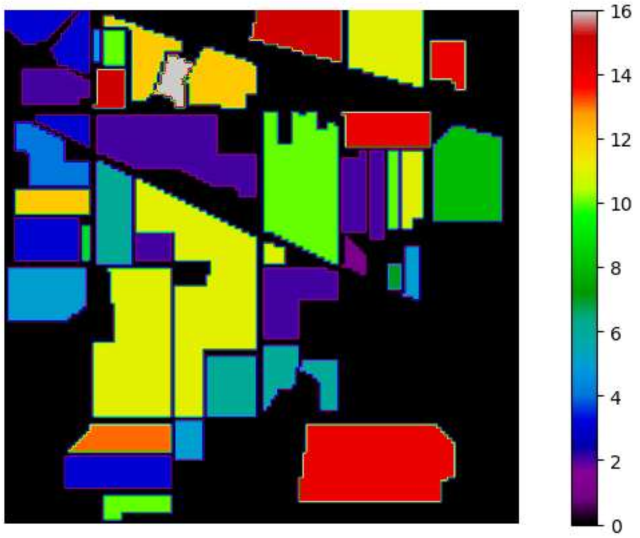
```
#shape of the dataset
print("pinesdata_shape", X.shape)
print("groundtruth_shape", y.shape)
```

```
pinesdata_shape (145, 145, 200)
groundtruth_shape (145, 145)
```

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

ax = fig.subplots(2, 5)

for i in range(2):
    for j in range(5):
        c = np.random.randint(200)
        ax[i][j].imshow(X[:, :, c], cmap = "nipy_spectral")
        ax[i][j].axis("off")
        ax[i][j].title.set_text(f"Band - {c}")
        c+=1

plt.tight_layout()

plt.show()
```



```
#visualizing of groundtruth
def plot_data(pines):
    fig = plt.figure(figsize = (10, 5))
    plt.imshow(y, cmap = "nipy_spectral")
    plt.colorbar()
    plt.axis("off")
    plt.show()

plot_data(y)
```




```
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("pinesdata.csv")
    return pines

pines = extract_pixels(X, y)
```

```
pines.head()
```

	band1	band2	band3	band4	band5	band6	band7	band8	band9	band10	...	band192
0	3172	4142	4506	4279	4782	5048	5213	5106	5053	4750	...	1094
1	2580	4266	4502	4426	4853	5249	5352	5353	5347	5065	...	1108
2	3687	4266	4421	4498	5019	5293	5438	5427	5383	5132	...	1111
3	2749	4258	4603	4493	4958	5234	5417	5355	5349	5096	...	1122
4	2746	4018	4675	4417	4886	5117	5215	5096	5098	4834	...	1110

```
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()
```

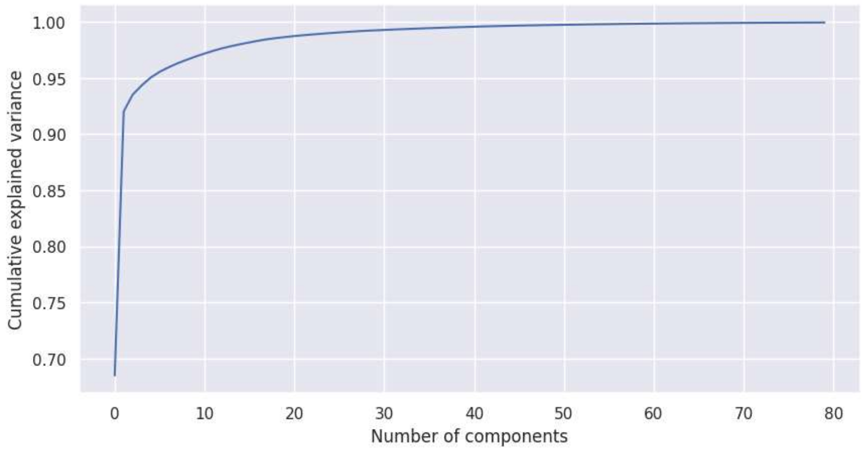
	band1	band2	band3	band4	band5	ban
count	21025.000000	21025.000000	21025.000000	21025.000000	21025.000000	21025.0000
mean	2957.363472	4091.321237	4277.502259	4169.956671	4516.678668	4790.5951
std	354.918708	230.390005	257.827640	280.761254	346.035984	414.3821
min	2560.000000	2709.000000	3649.000000	2810.000000	3840.000000	4056.0000
25%	2602.000000	3889.000000	4066.000000	3954.000000	4214.000000	4425.0000
50%	2780.000000	4106.000000	4237.000000	4126.000000	4478.000000	4754.0000
75%	3179.000000	4247.000000	4479.000000	4350.000000	4772.000000	5093.0000
max	4536.000000	5744.000000	6361.000000	6362.000000	7153.000000	7980.0000

8 rows × 201 columns

```
#Principal Component Analysis(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, 1.94079617e-04,
       1.81978321e-04, 1.70834583e-04, 1.55749872e-04, 1.41898396e-04,
       1.37335866e-04, 1.36430860e-04, 1.33485428e-04, 1.23374686e-04,
       1.21877863e-04, 1.20991220e-04, 1.14749905e-04, 1.13124569e-04,
       1.04952998e-04, 1.02963482e-04, 9.31345102e-05, 8.91231088e-05,
       8.49667881e-05, 8.41796594e-05, 7.60502686e-05, 7.02646554e-05,
       6.77536774e-05, 6.28481008e-05, 6.23356714e-05, 6.03227328e-05,
       5.56624387e-05, 5.29583422e-05, 5.09734149e-05, 4.56084178e-05,
       4.20277744e-05, 4.14197212e-05, 3.93165110e-05, 3.85844278e-05,
       3.77313382e-05, 3.66094028e-05, 3.57553235e-05, 3.48725971e-05])
```

```
sns.set()
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)
data_pines = pd.concat([pd.DataFrame(data = data), pd.DataFrame(data = y.ravel())], axis = 1)
data_pines.columns = [f"PC-{i}" for i in range(1,41)] + ["class"]
```

```
data_pines.head()
```

	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC·
0	5014.905666	1456.863532	72.697659	71.201091	-435.684640	-68.843373	134.80986
1	5601.383449	-2023.449776	350.135434	-528.457155	148.103479	-288.362816	202.95600
2	5796.135157	-3090.394530	490.540544	-760.205255	259.951278	-131.614590	172.9266
3	5586.204284	-2369.375772	356.275521	-502.679337	146.569639	-306.682856	251.0703
4	5020.990484	339.603668	-23.006921	-92.558409	-368.488736	-438.269677	502.7156

5 rows × 41 columns

```
#saving to .csv
data_pines.to_csv("Pines_PCA.csv", index = False)
x = data_pines[data_pines["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
#split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 11, stratify = y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((7174, 40), (3075, 40), (7174,), (3075,))
```

```
#Random Forest Classifier
model = RandomForestClassifier(n_estimators = 100, random_state = 42)
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(y_train, X_train_prediction)
print("Accuracy on Training Data: ", training_data_accuracy)
```

```
Accuracy on Training Data:  1.0
```

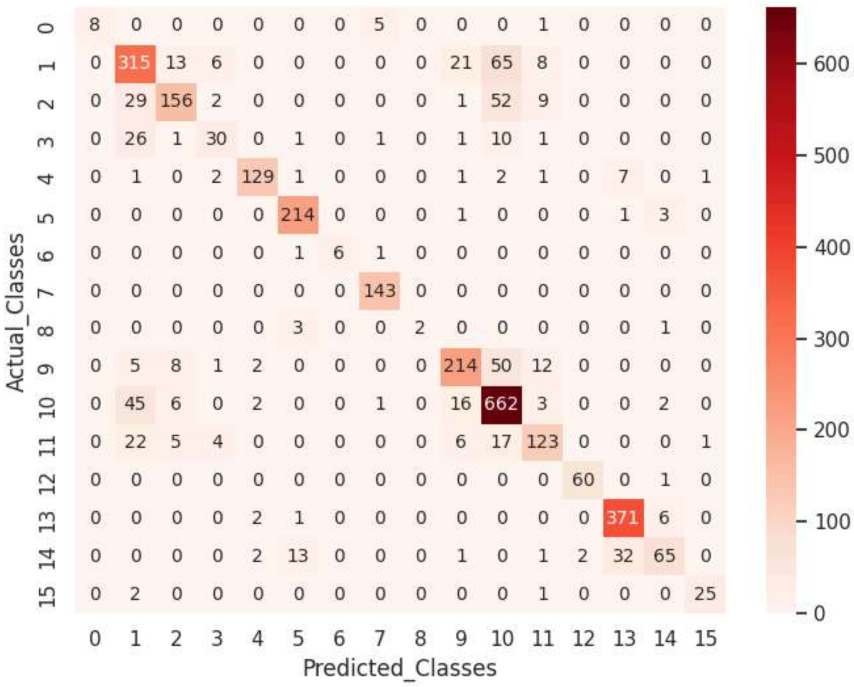
```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, y_test)
print("Accuracy on Test Data: ", test_data_accuracy)
```

```
Accuracy on Test Data:  0.8204878048780487
```

```
#classification report
print(classification_report(y_test, X_test_prediction, target_names = ["Brocoli_green_weeds_1", "Brocoli_green_weeds_2", "Fallow", "Fal]
```

```
                          precision    recall  f1-score   support

    Brocoli_green_weeds_1       1.00      0.57      0.73        14
    Brocoli_green_weeds_2       0.71      0.74      0.72       428
                   Fallow       0.83      0.63      0.71       249
        Fallow_rough_plow       0.67      0.42      0.52        71
            Fallow_smooth       0.94      0.89      0.91       145
                  Stubble       0.91      0.98      0.94       219
                   Celery       1.00      0.75      0.86         8
        Grapes_untrained        0.95      1.00      0.97       143
       Soil_vinyard_develop     1.00      0.33      0.50         6
  Corn_senesced_green_weeds     0.82      0.73      0.77       292
      Lettuce_romaine_4wk       0.77      0.90      0.83       737
      Lettuce_romaine_5wk       0.77      0.69      0.73       178
      Lettuce_romaine_6wk       0.97      0.98      0.98        61
      Lettuce_romaine_7wk       0.90      0.98      0.94       380
        Vinyard_untrained       0.83      0.56      0.67       116
   Vinyard_vertical_trellis     0.93      0.89      0.91        28

                 accuracy                           0.82      3075
                macro avg       0.87      0.75      0.79      3075
             weighted avg       0.82      0.82      0.82      3075
```

```
#confusion matrix
plt.figure(figsize = (8,6))

c_matrix = confusion_matrix(y_test, X_test_prediction)

sns.heatmap(c_matrix, annot = True, annot_kws = {"size": 10}, fmt = "d", cmap = "Reds")
plt.xlabel("Predicted_Classes")
plt.ylabel("Actual_Classes")
plt.show()
```




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

```
Root Mean Square Error:
 2.784626786746011
```

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

```
Mean Absolute Error (MAE) 21.802926829268294
```

```
l =[]

for i in range(data_pines.shape[0]):
    if data_pines.iloc[i, -1] == 0:
        l.append(0)
    else:
        l.append(model.predict(data_pines.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 = "nipy_spectral")
plt.colorbar()
plt.axis("off")
plt.title("Random Forest Classification Map")
plt.savefig("randomforest_classification_map.png")
plt.show()
```

```
<ipython-input-42-db5af8ac9e04>:2: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-t
  clmap = np.array(l).reshape(145, 145).astype("float")
```