1. Import all the required libraries

In [3]:
```
# calling database program from geoAnalytics package. Use the term db as sh
from geoAnalytics import database as db
# calling oneclassClassifiers program from geoAnalytics to get top-k sample
from geoAnalytics.oneclassClassifiers import *
import pandas as pd
# import gdal library from osgeo package
from osgeo import gdal
```

2. Read CSV File which contains points selected using QGIS Plugin

In [9]:
```
df1 = pd.read_csv('trainFile.csv',header=None)

# removing note,x,y column from dataframe
train_df = df1.iloc[:,3:-1]
cols = []
for i in range(1,train_df.shape[1]+1):
    cols.append('b'+str(i))
train_df.columns = cols
train_df.head()
```

Out[9]:

	b1	b2	b3	b4	b5	b6	b7	b8	b9
0	0.04100	0.07654	0.08522	0.08680	0.08886	0.08942	0.09378	0.11334	0.13946
1	0.05128	0.09314	0.10362	0.10430	0.10836	0.10788	0.11530	0.13682	0.16400
2	0.04030	0.07332	0.07970	0.08116	0.08454	0.08272	0.08664	0.10588	0.13092
3	0.04336	0.08134	0.09140	0.09310	0.09926	0.09644	0.10204	0.12248	0.14970
4	0.04548	0.08482	0.09202	0.09440	0.09888	0.09618	0.10230	0.12312	0.14908

In [10]:
```
train_df.shape
```

Out[10]:
```
(38, 9)
```

3. Connect to database

In [12]:
```
db.connect(dbName='kaguya',hostIP='163.143.165.136',user='raashika', passwo
```

```
('PostgreSQL 13.7 (Ubuntu 13.7-1.pgdg22.04+1) on x86_64-pc-linux-gnu, compi
led by gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0, 64-bit',)
You are now connected
```

4. Creating Repository

In [18]:
```
db.createRepository(repositoryName='MI_data1',totalBands=9)
```

```
Repository created
Repository connection closed.
```

5. Inserting data(File with which test data points are selected)

In [19]:
```
db.insertRaster(repositoryName='MI_data1',fileName="/hadoopData/kaguya/MI_M
```

6.Get Dataframe from the repository

In [20]:
```
test_df = db.getDataframe(repositoryName='MI_data1')
test_df.head()
```

```
Getting dataframe from database
/home/jupyterHub/anaconda3/envs/jupyterHub/lib/python3.10/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable (engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
Dataframe created
Repository connection closed.
```

Out[20]:

	x	y	b1	b2	b3	b4	b5	b6	b7	b8	b
0	0.0000	0.0	0.03692	0.06520	0.07148	0.07210	0.07406	0.07498	0.07918	0.09754	0.1219
1	14.8063	0.0	0.03674	0.06522	0.07140	0.07302	0.07468	0.07476	0.07890	0.09738	0.1214
2	29.6126	0.0	0.03652	0.06582	0.07164	0.07248	0.07506	0.07454	0.07862	0.09722	0.1209
3	44.4190	0.0	0.03628	0.06604	0.07178	0.07304	0.07530	0.07432	0.07832	0.09706	0.1206
4	59.2253	0.0	0.03666	0.06570	0.07132	0.07308	0.07540	0.07410	0.07828	0.09690	0.1205

In [21]:
```
test_df = test_df.iloc[:100,:]
test_df.shape
```

Out[21]:
```
(4194304, 11)
```

In [22]:
```
test_df1 = test_df.drop(['x','y'],axis=1)
```

In [23]:
```
test_df1.head()
```

Out[23]:

	b1	b2	b3	b4	b5	b6	b7	b8	b9
0	0.03692	0.06520	0.07148	0.07210	0.07406	0.07498	0.07918	0.09754	0.12190
1	0.03674	0.06522	0.07140	0.07302	0.07468	0.07476	0.07890	0.09738	0.12142
2	0.03652	0.06582	0.07164	0.07248	0.07506	0.07454	0.07862	0.09722	0.12096
3	0.03628	0.06604	0.07178	0.07304	0.07530	0.07432	0.07832	0.09706	0.12066
4	0.03666	0.06570	0.07132	0.07308	0.07540	0.07410	0.07828	0.09690	0.12052

7. Getting top-k samples

Applying rasterFuzzyDistance Algorithm from oneclassClassifiers to get top-k samples

In [34]:
```
FinalSamples, TopKSamples = rasterFuzzyTSC(train_df,test_df1, 10)
```

```
Total Execution time of proposedAlgo 1788.1689734458923
Memory of proposedAlgo in KB: 2306736.0
```

8.display top-k samples and save output

In [35]:
```
# print('final samples: \n')
```

```
# print(FinalSamples)

print('Top K samples : \n')
print(TopKSamples.head())
saveFinalTestSamples(FinalSamples,"oneNNEDFinalSamples.csv")
saveFinalTestSamples(TopKSamples, "oneNNEDTopKSamples.csv")
```

```
Top K samples :

              b1       b2       b3       b4       b5       b6       b7  \
1375222  0.04436  0.07962  0.08856  0.08996  0.09414  0.09264  0.09754
1164269  0.04384  0.07974  0.08834  0.09000  0.09428  0.09260  0.09784
2116315  0.04494  0.07976  0.08842  0.08996  0.09452  0.09250  0.09754
1606624  0.04364  0.07972  0.08822  0.09004  0.09448  0.09280  0.09770
2077460  0.04416  0.07964  0.08812  0.08978  0.09420  0.09266  0.09784

              b8       b9        RD
1375222  0.11812  0.14454  0.002501
1164269  0.11832  0.14514  0.002758
2116315  0.11826  0.14426  0.002825
1606624  0.11866  0.14468  0.002864
2077460  0.11822  0.14482  0.002899
```

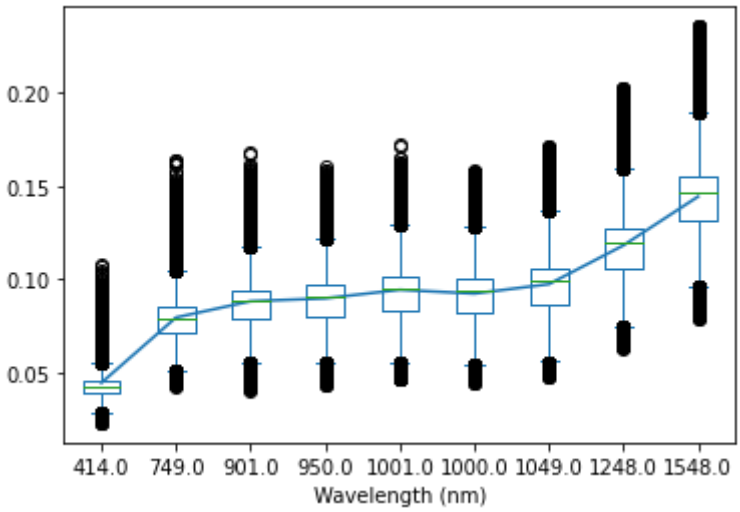
In [36]:
```
test_df1.drop(['RD'],axis=1,inplace=True)
```

In [38]:
```
import matplotlib.pyplot as plt
from osgeo import gdal

path  = '/hadoopData/kaguya/MI_MAP_03_N00E000S01E001SC.lbl'

filename=gdal.Open(path)

metadata=filename.GetMetadata()

wavelengthValues  = metadata.get('CENTER_FILTER_WAVELENGTH')[1:-1].replace(

wavelengthValues = [float(wavelength) for wavelength in wavelengthValues]

bandsCount = [*range(1,len(wavelengthValues)+1, 1)]
```

9 Plot Top K pixel

In [39]:
```
def plotPixel(index):
    # fig, ax = plt.subplots()
    test_df1.plot(kind = 'box')
    bandData = list(TopKSamples.iloc[index,:-1])
    plt.plot(bandsCount, bandData)
    plt.xticks(bandsCount, wavelengthValues)
    plt.xlabel("Wavelength (nm)")
```

In [40]:
```
plotPixel(2)
```


0.20
0.15
0.10
0.05
414.0
749.0
901.0
950.0
1001.0
1000.0
1049.0
1248.0
1548.0
Wavelength (nm)

In [ ]: