Soil Layering by Cone Penetration Test Data

June 4, 2025

1 Soil Layering by Cone Penetration Test Data

This notebook aims to explore various machine learning approaches to automatically determine soil stratification based on CPT data. Author: Zhiyan Jiang (linkedIn.com/zhiyanjiang)

[31]:
```
import warnings
warnings.simplefilter("ignore") # default

#%matplotlib inline
```

[32]:
```
# import external libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import time
from scipy.signal import find_peaks
from scipy.signal import peak_prominences
import pickle
import io

# import internal libraries
from constants import PSF2TSF, PA2TSF, PSI2TSF
from importData import *
from plotCPT import *
from plot import *
```

1.1 Step 1: configurations

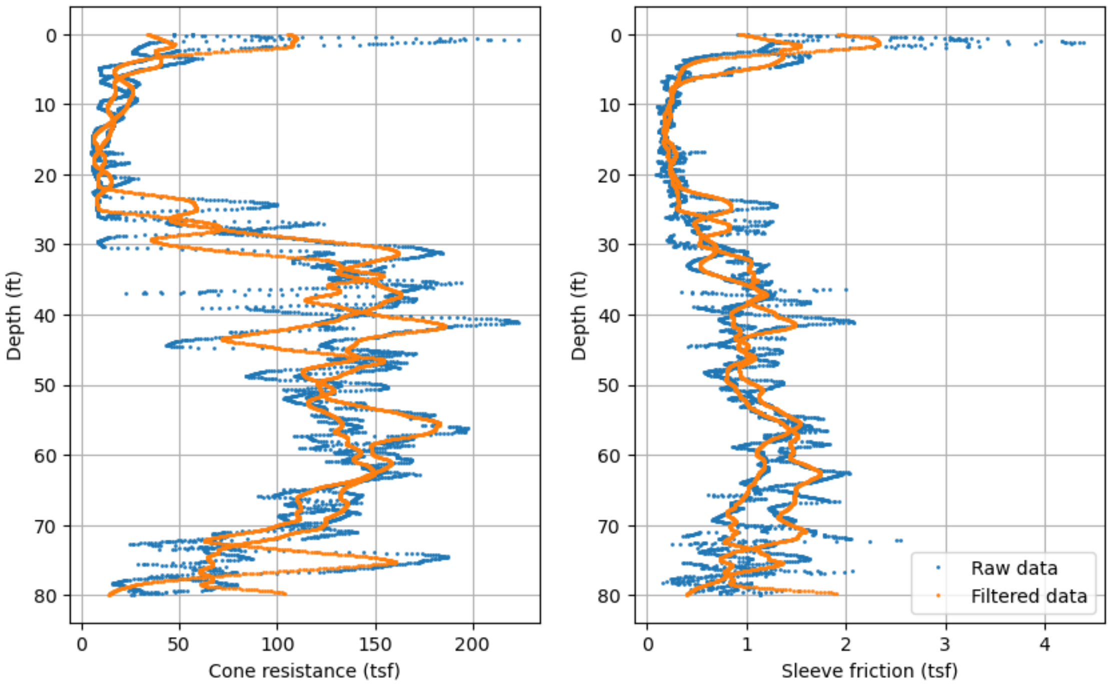
[33]:
```
cptFileName = "..\\Data\\NSF\\4_5.csv"
gwtFileName = "..\\Data\\GWT data.csv"
SBTnImgFileName = '.\\Images\\SBTn_background.jpg'

# Include multiple cptFileNames to the list, if needed
cptFileNames = [cptFileName] #, "..\\Data\\NSF\\981.csv"]

# Clustering parameters
randomState = None # 0 or # None, i.e. time.time()
numberClusters = 4
```

```
# CPT-specific parameters
# area ratio
an = 0.88

# Assume uniform unit weight
soilUnitWeight = 120 # unit shall be pcf
waterUnitWeight = 62.4 # unit shall be pcf

# Applying filtering, unit of feet
# Purpose: to remove consecutive peaks
windowLength = 3
# Repeating filtering. Default = 1
filterTimes = 1

# Remove peaks
removePeaksFlag = False
```

1.2 Step 2: Import data, averaging, and removing spikes.

[34]:
```
# Import data and apply filtering
plotImportFlag = True
rawData = importCPTs(cptFileNames,  windowLength, filterTimes,removePeaksFlag,␣
 ↪plotImportFlag)

# Import GWT data
gwtData = importGWT(gwtFileName)

# Backup rawData
rawDataCopy = rawData.copy()
```

```
Row number is: 4866, and Column number is: 4
```

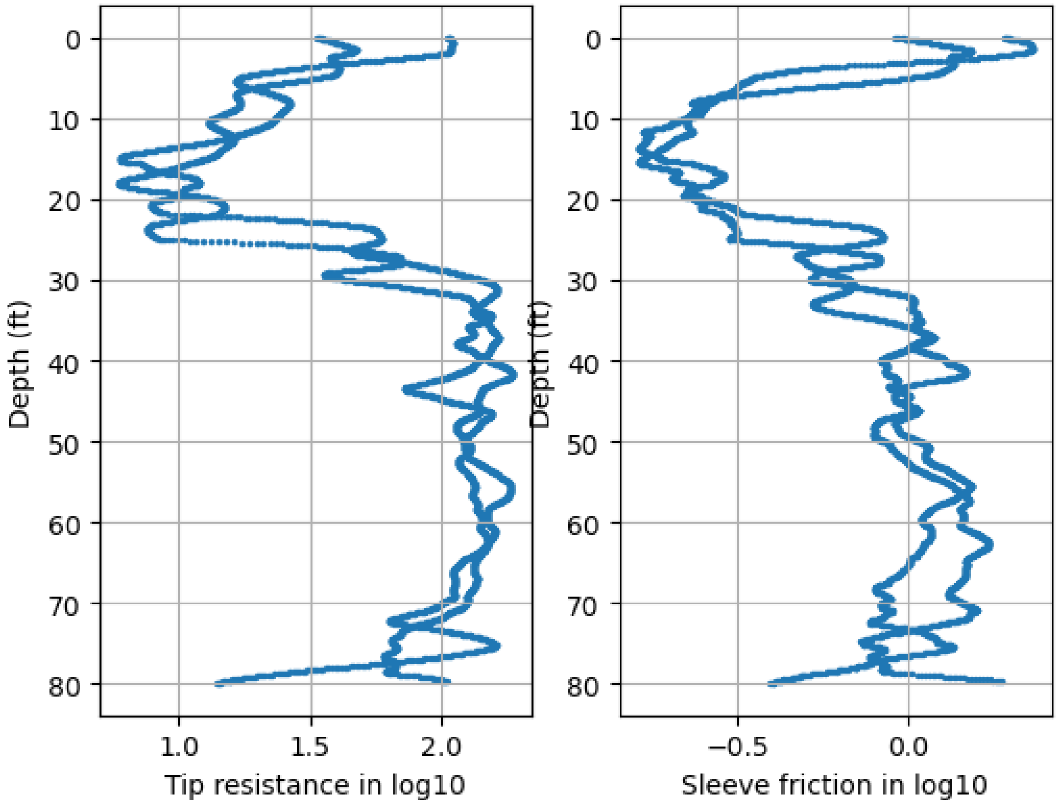


1.3 Step 3: Pre-processing

[35]:
```
# Need to sort data in terms of depth
from processCPT import *
# Correct cone resistance to get tip resistance
data = pd.DataFrame(columns = ["Depth (ft)", "Tip resistance (tsf)", "Sleeve␣
↪friction (tsf)"])

data["Depth (ft)"] = rawData.iloc[:, 0]
data["Tip resistance (tsf)"] = rawData.iloc[:, 1]

data["Tip resistance (tsf)"] = rawData.iloc[:, 1]
if rawData.shape[1] > 3: # rawData has pore pressure
    data["Tip resistance (tsf)"] += (1-an) * rawData.iloc[:, 3] * PSI2TSF


data["Sleeve friction (tsf)"] = rawData.iloc[:, 2]

depth = data["Depth (ft)"].to_frame()
```

[36]:
```
# Prepare data in log-scale
dataLog = data.copy()
dataLog.iloc[:,1:] = np.log10(dataLog.iloc[:,1:])
```

```
dataLog.columns =  ["Depth (ft)", "Tip resistance in log10", "Sleeve friction␣
 ↪in log10"]


dataLogAxes = plotAggregate(dataLog, labels = dataLog.columns, markerSize = 2,␣
 ↪axes  = None)
```



1.4 3.1 Sort

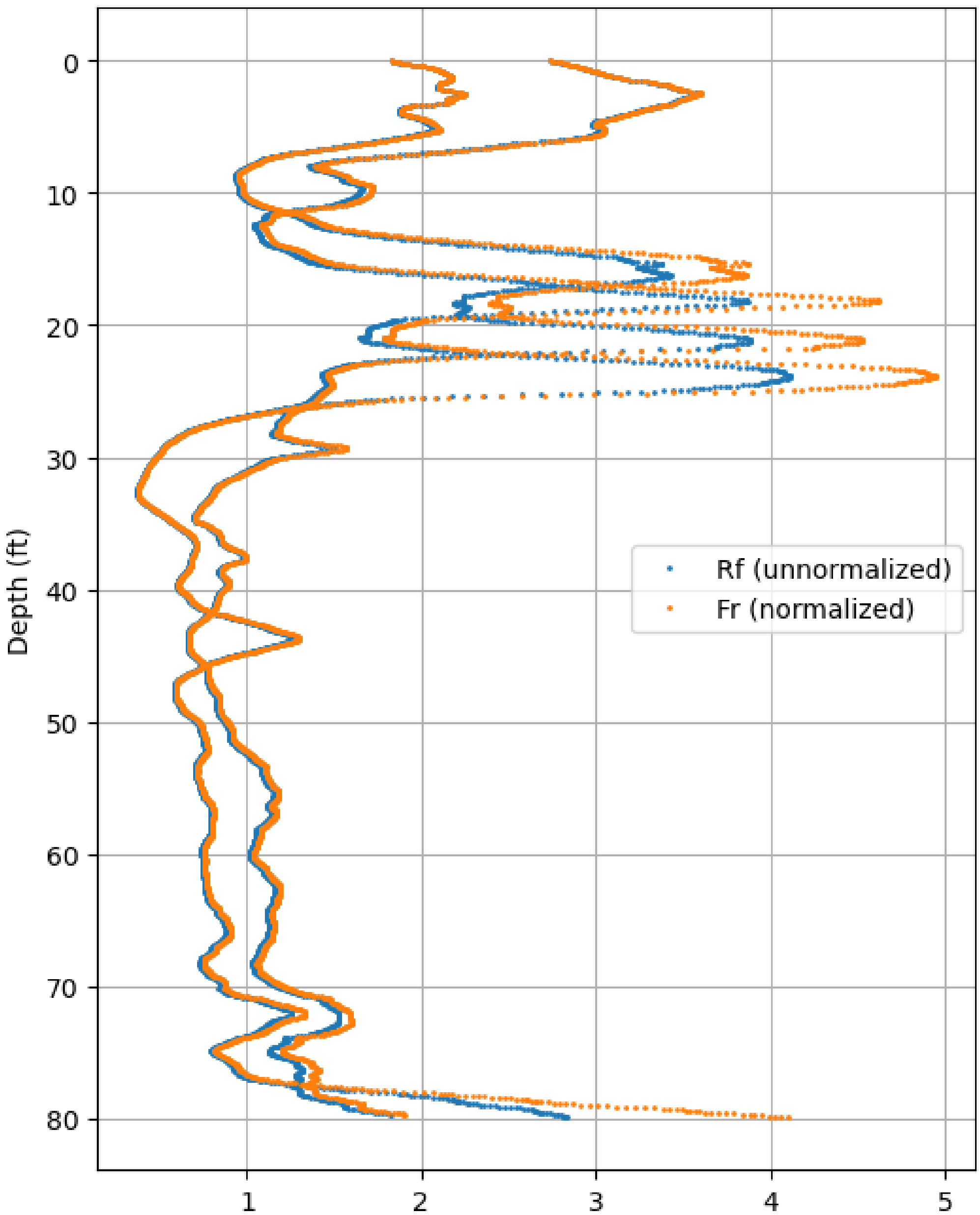
[37]:
```
# PLACE HOLDER
```

1.5 3.2 Calculate derived CPT parameters

The equation for normalized soil behavior type can be found in: Robertson, P.K. Guide to Cone Penetration Testing, 6th Ed., 2015

[38]:
```
# Calculate friciton ratio
Rf = calculateRf(data.iloc[:,1], data.iloc[:,2])


# calculate normalized friction ratio
```

```
Fr = calculateFr(data.iloc[:,0].to_frame(), data.iloc[:,1].to_frame(), data.
↪iloc[:,2].to_frame(), soilUnitWeight)

plotRfFr(depth, Rf, Fr)
```

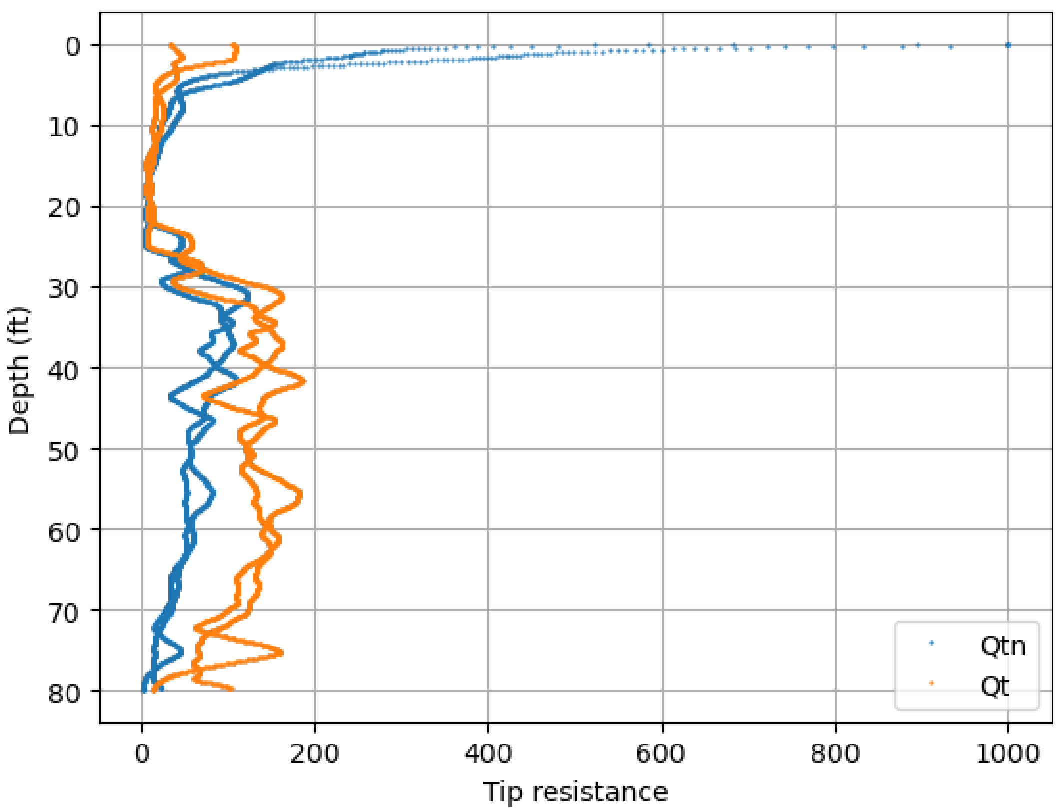


[39]:
```
# Calculate stresses
sigma_vo = calculateSigma_vo(data.iloc[:,0].to_frame(), soilUnitWeight)
```

```
hydroStaticPressure = calculateHydroStaticPressure(data.iloc[:, 0].to_frame(),␣
↪gwtData.iloc[0,0], waterUnitWeight)

sigma_vo_prime = (sigma_vo.iloc[:,0] - hydroStaticPressure.iloc[:,0]).to_frame()

# calculate Qtn
Qtn = iterateQtn(depth, data.iloc[:,1].to_frame(), Fr, gwtData.iloc[0,0],␣
↪soilUnitWeight, waterUnitWeight, PA2TSF)
plt.plot(Qtn, depth, linestyle = '', marker = '.', markersize = 1)
plt.gca().invert_yaxis()
plt.plot(data.iloc[:,1], data.iloc[:,0], linestyle = '', marker = '.',␣
↪markersize = 1)
plt.legend(["Qtn", "Qt"])
plt.grid(True)
plt.ylabel("Depth (ft)")
plt.xlabel("Tip resistance")
```

```
Qtn converged in 4 times
```

[39]:
```
Text(0.5, 0, 'Tip resistance')
```



[40]:
```
# Calculate and plot NORMALIZED Soil behavior type Index
Ic = calculateIc(Qtn, Fr)
```

1.6 Step 4: Clustering

1.7 4.1 Decision Tree

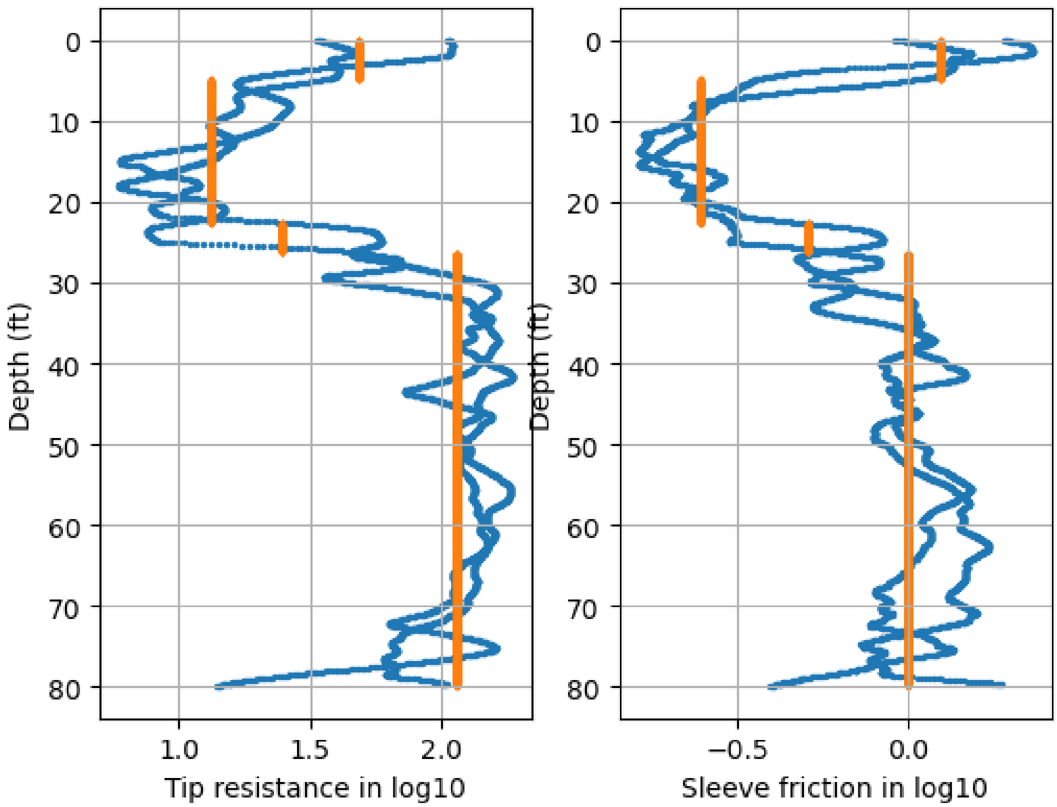
[41]:
```
from performDecisionTreeClustering import *
```

1.7.1 4.1.1 Decision Tree on log of [tip resistance, sleeve friction]

[42]:
```
# Perform decisiontree regression on [tip resistance, sleeve friction]
performDecisionTreeFlag = "regression"
decisionTreeInput = [dataLog, numberClusters, randomState]
decisionTreeObj, decisionTreeResult =␣
 ↪performDecisionTree(performDecisionTreeFlag, decisionTreeInput)

# Extract decisionTreeCriteria
decisionTreeCriteria = getDecisionTreeCriteria(decisionTreeObj)
print(f"The layer interface depths resulting from Decision tree method is below:
 ↪")
print(decisionTreeCriteria)
print()

# plot decision tree results
dataLogAxes = plotAggregate(dataLog, labels = dataLog.columns, markerSize = 2,␣
 ↪axes  = None)
plotDecisionTreeResult(decisionTreeResult, dataLog, axes = dataLogAxes)
```

```
The score by Decision tree regression: 0.7923484234428955
The layer interface depths resulting from Decision tree method is below:
   Decision Tree Criteria
0                4.839239
1               22.621390
2               26.295932

Below is the results by Decision tree regression
```

[42]:
```
array([<Axes: xlabel='Tip resistance in log10', ylabel='Depth (ft)'>,
       <Axes: xlabel='Sleeve friction in log10', ylabel='Depth (ft)'>],
      dtype=object)
```



1.7.2 4.1.2 Decision Tree on Ic

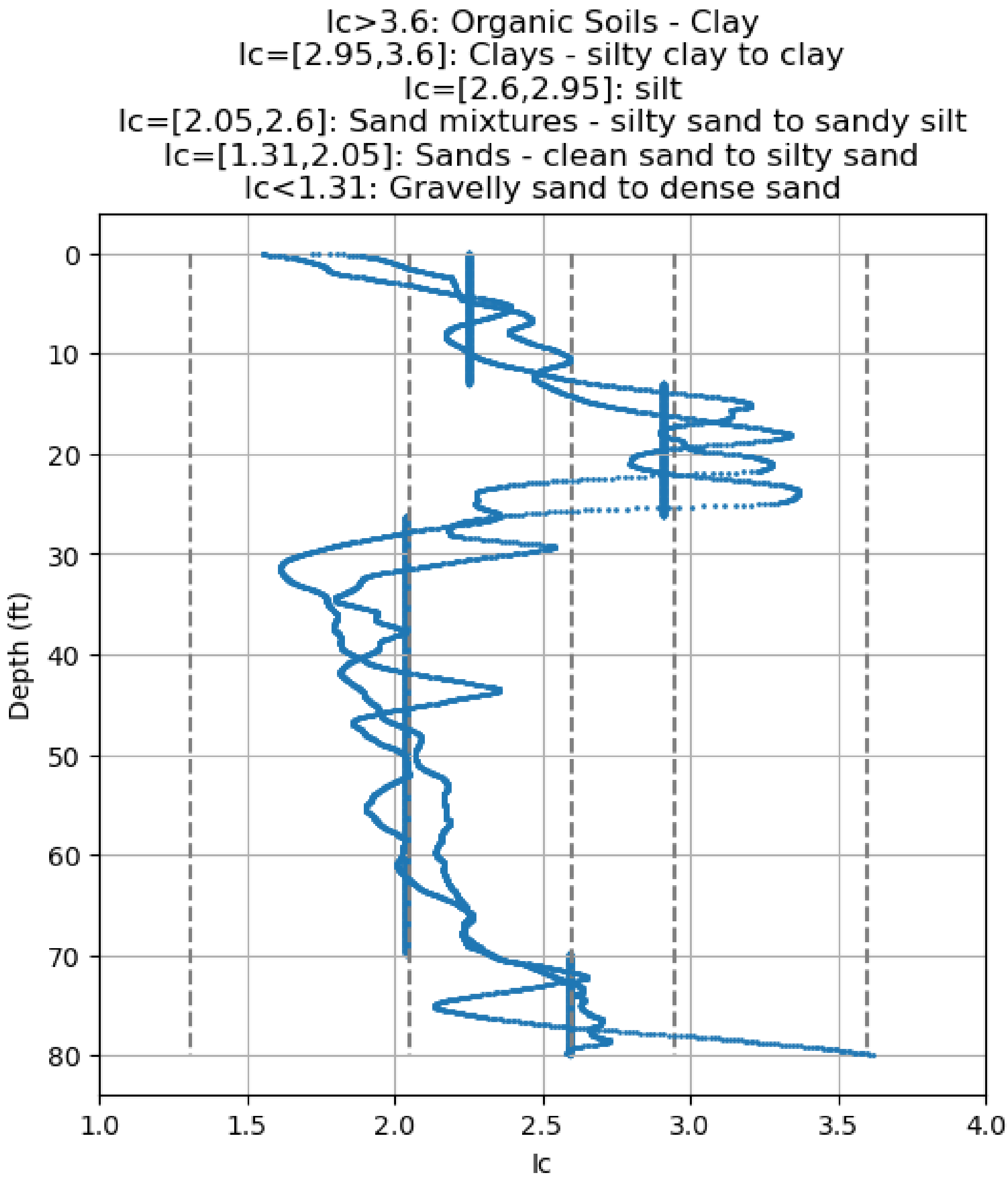
[43]:
```
# Perform decision tree on Ic
dataIc = pd.concat([data.iloc[:,0], Ic], axis = 1)
performDecisionTreeFlag = "regression"
decisionTreeInput = [dataIc, numberClusters, randomState]
decisionTreeObj, decisionTreeResult =␣
 ↪performDecisionTree(performDecisionTreeFlag, decisionTreeInput)

# Extract decisionTreeCriteria
decisionTreeCriteria = getDecisionTreeCriteria(decisionTreeObj)
print(f"The layer interface depths resulting from Decision tree method is below:
 ↪")
print(decisionTreeCriteria)
print()

# plot decision tree results
IcAxes = plotIc(data.iloc[:,0].to_frame(), Ic)
plotDecisionTreeResult(decisionTreeResult, dataIc, IcAxes)

# Plot Ic histogram
```

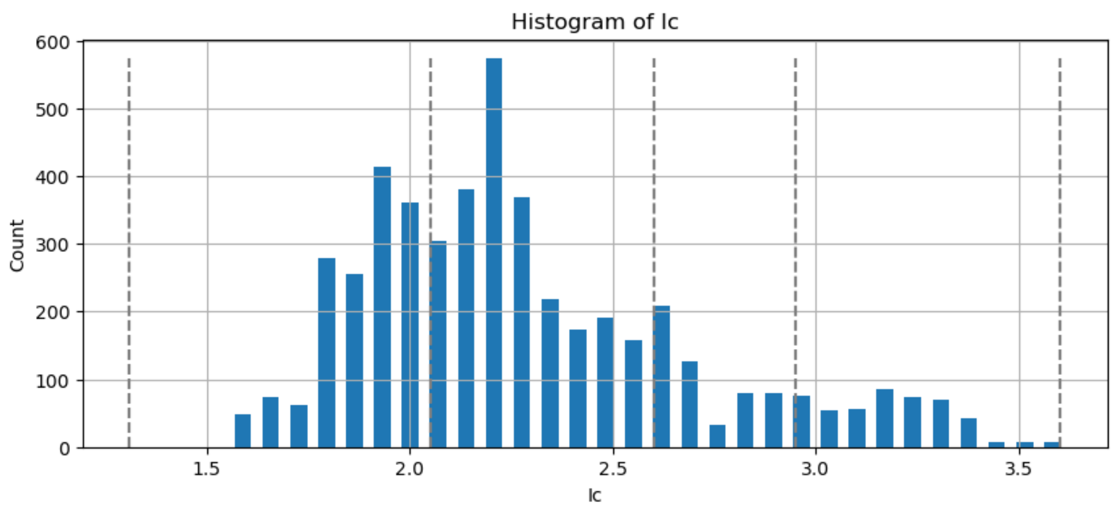
```
IcHistogramAxes = plotIcHistogram(Ic)
IcCounts, _ = np.histogram(Ic, bins = 30)

# Plot Icthreshold
plotIcThreadholds(np.max(IcCounts), IcHistogramAxes)

#IcAxes[0].figure
```

```
The score by Decision tree regression: 0.6668726596924246
The layer interface depths resulting from Decision tree method is below:
   Decision Tree Criteria
0               12.942913
1               26.197507
2               69.832680

Below is the results by Decision tree regression
```

[43]:

```
[<Axes: title={'center': 'Histogram of Ic'}, xlabel='Ic', ylabel='Count'>]
```


Ic>3.6: Organic Soils - Clay
Ic=[2.95,3.6]: Clays - silty clay to clay
Ic=[2.6,2.95]: silt
Ic=[2.05,2.6]: Sand mixtures - silty sand to sandy silt
Ic=[1.31,2.05]: Sands - clean sand to silty sand
Ic<1.31: Gravelly sand to dense sand
Depth (ft)
Ic
0
10
20
30
40
50
60
70
80
1.0
1.5
2.0
2.5
3.0
3.5
4.0



1.7.3 4.1.3 Decision tree on SBTn type

Infer soil behavior type based on SBTn type, then apply decision tree

[44]:
```
# Obtain sbtn
dataSBTn1D = calculateSBTn1D(dataIc)
```

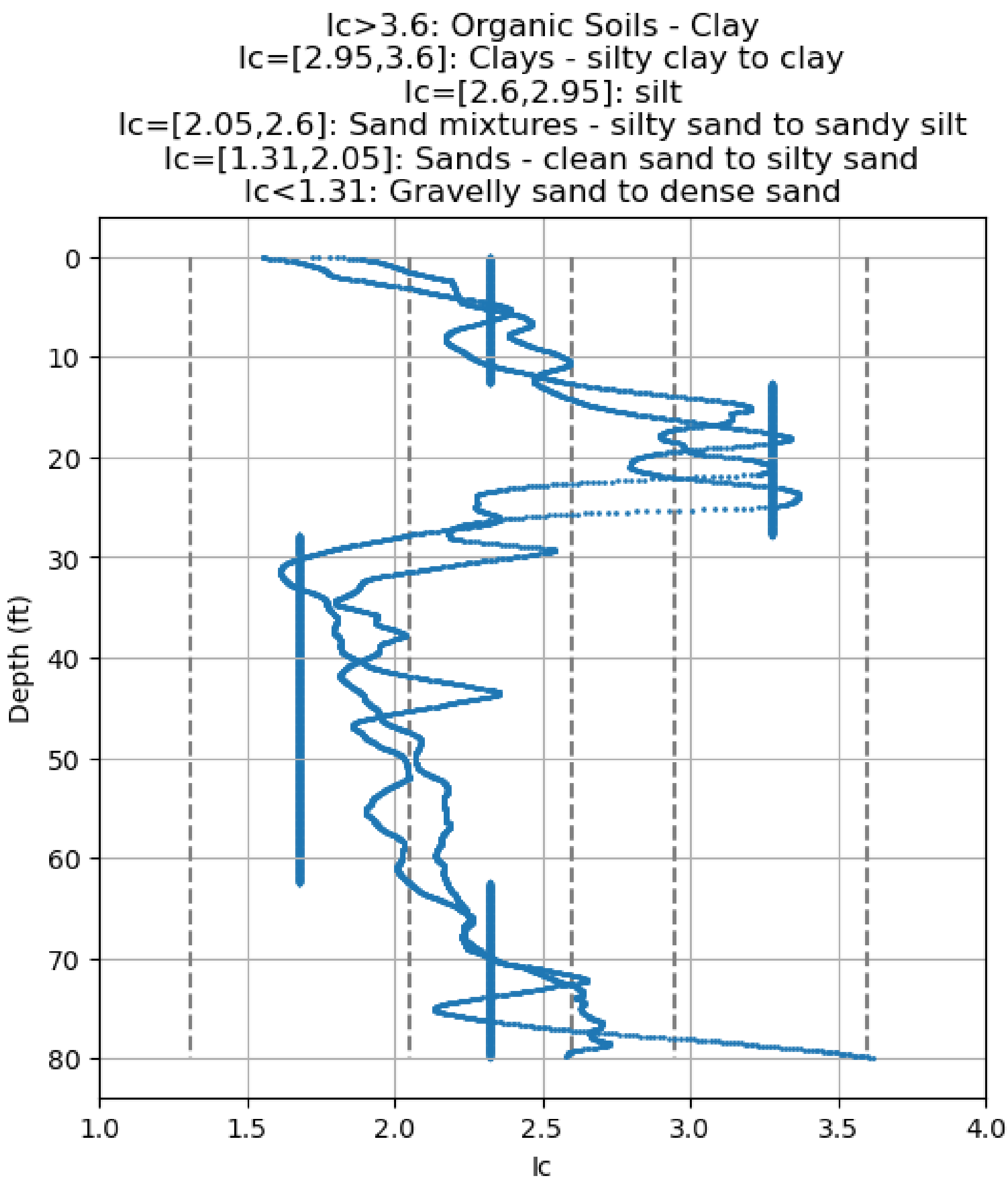
```
4866
```

[45]:
```
# Perform decisiontree regression on SBTn
performDecisionTreeFlag = "classification" # when using SBTn, must use␣
↪"classification"

decisionTreeInput = [dataSBTn1D, numberClusters, randomState]
decisionTreeObj, decisionTreeResult =␣
↪performDecisionTree(performDecisionTreeFlag, decisionTreeInput)

# Extract decisionTreeCriteria
decisionTreeCriteria = getDecisionTreeCriteria(decisionTreeObj)
print(f"The layer interface depths resulting from Decision tree method is below:
↪")
print(decisionTreeCriteria)
print()

# plot decision tree results
IcAxes = plotIc(data.iloc[:,0].to_frame(), Ic)
# Obtain SBTn corresponded middle Ic value
decisionTreeResultSBTn1DIc = calculateSBTn1DIc(decisionTreeResult)
plotDecisionTreeResult(decisionTreeResultSBTn1DIc, dataIc, IcAxes)
```

```
The score by Decision tree regression: 0.6623510069872586
```

```
The layer interface depths resulting from Decision tree method is below:
   Decision Tree Criteria
0               12.647637
1               27.772309
2               62.483595

Below is the results by Decision tree regression
```

```
[45]: [<Axes: title={'center': 'Ic>3.6: Organic Soils - Clay\nIc=[2.95,3.6]: Clays -
silty clay to clay\nIc=[2.6,2.95]: silt\nIc=[2.05,2.6]: Sand mixtures - silty
sand to sandy silt\nIc=[1.31,2.05]: Sands - clean sand to silty sand\nIc<1.31:
Gravelly sand to dense sand'}, xlabel='Ic', ylabel='Depth (ft)'>]
```



1.8 4.2 Random Forest

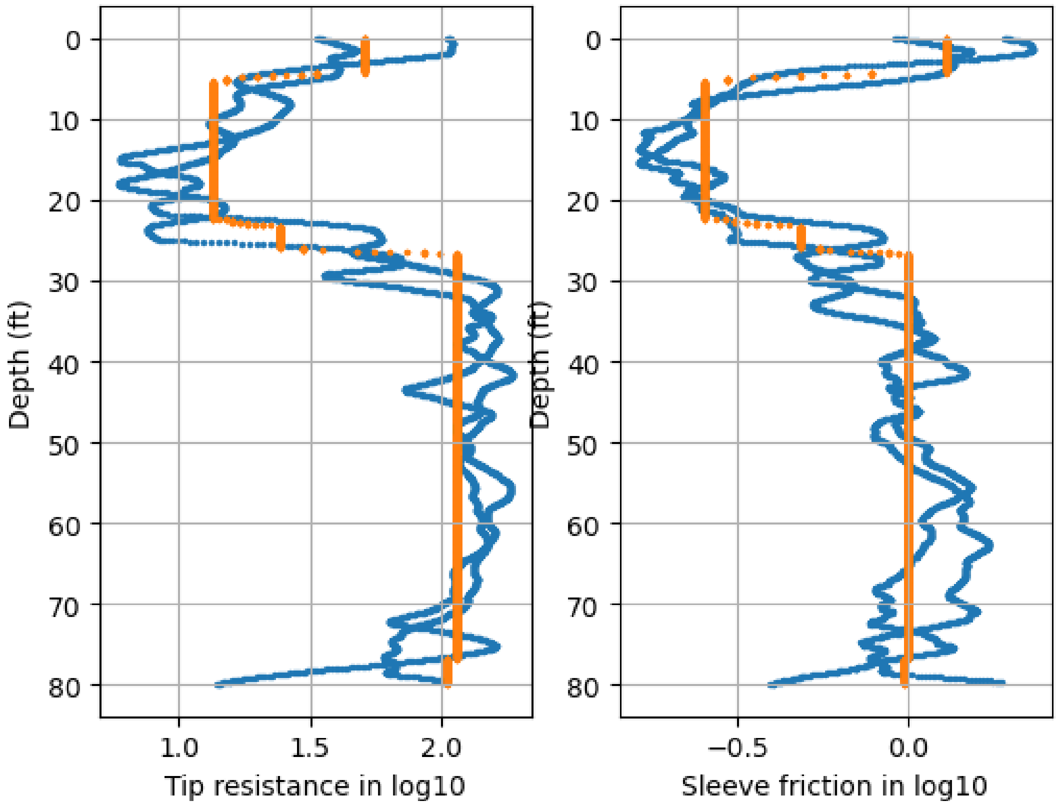
[46]:
```python
# Perform Random Forest
from performRandomForestClustering import *
```

1.8.1 4.2.1 Perform random forest regression on log [tip resistance, sleeve friction]

[47]:
```python
# Perform Random Forest on log of [Tip resistance, sleeve friction]

performRandomForestFlag = "regression"
numberTrees = 10
maxLeafNodes = numberClusters
randomForestInput = [dataLog, numberTrees, maxLeafNodes, randomState]

randomForestObj, randomForestResult = performRandomForest(performRandomForestFlag, randomForestInput)

# plot random forest result
dataLogAxes = plotAggregate(dataLog, labels = dataLog.columns, markerSize = 2, axes  = None)
plotRandomForestResult(randomForestResult, dataLog, dataLogAxes)
```

```
Use Random Forest regressor.
The score by Random Forest: 0.8054848914861151
Below is the results by Random forest regression
```

[47]:
```
array([<Axes: xlabel='Tip resistance in log10', ylabel='Depth (ft)'>,
       <Axes: xlabel='Sleeve friction in log10', ylabel='Depth (ft)'>],
      dtype=object)
```

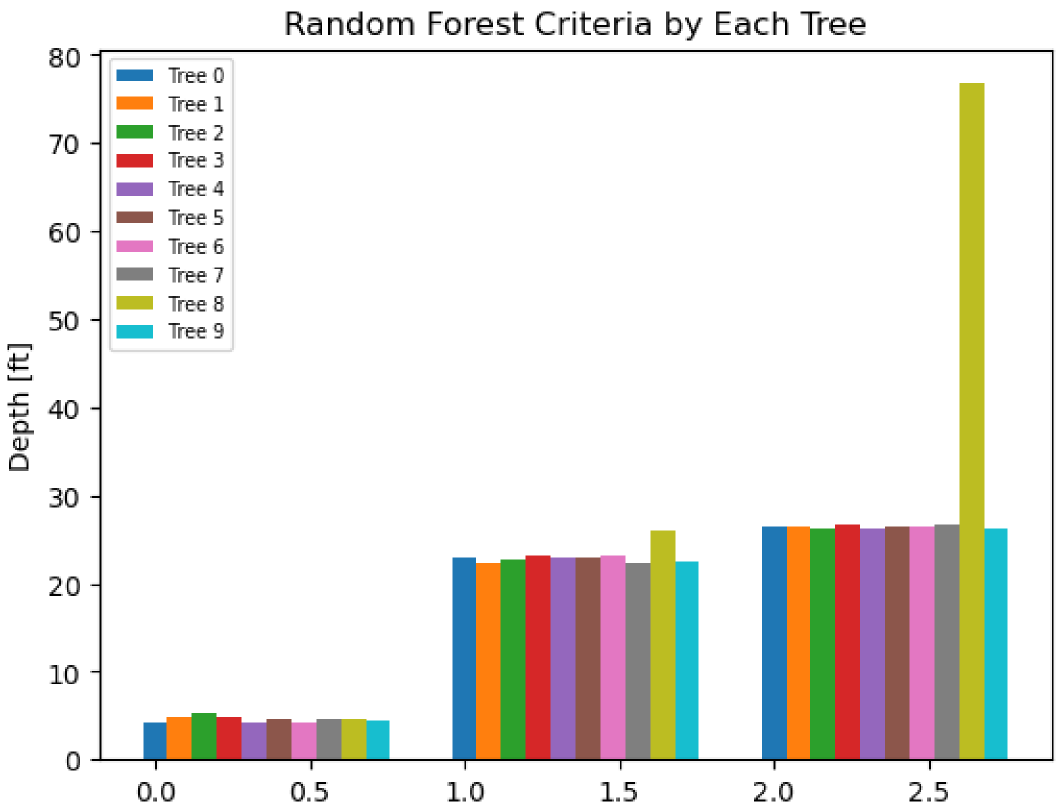


[48]:

```
# Extract layer interface depths from non-leaf nodes
randomForestCriteria = getRandomForestCriteria(randomForestObj)

# plot results by each tree using bar chart
plotRandomForestCriteria(randomForestCriteria)

# reduce randomForestCriteria as median
randomForestCriteriaReduced = randomForestCriteriaMedian(randomForestCriteria)
print(f"The reduced Random Forest criteria by median is:
 ↪\n{randomForestCriteriaReduced}")

# reduce randomForestCriteria as majority
randomForestCriteriaReduced = randomForestCriteriaMajority(randomForestCriteria)
print(f"The reduced Random Forest criteria by majority is:
 ↪\n{randomForestCriteriaReduced}")
```

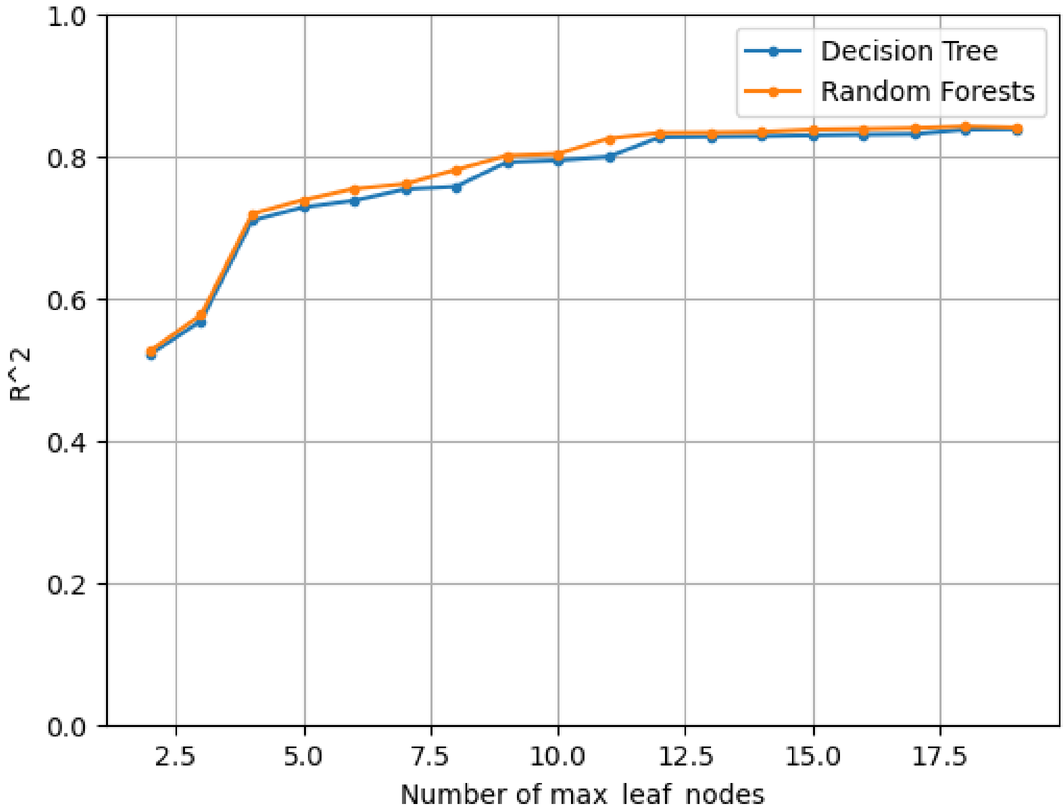
```
The reduced Random Forest criteria by median is:
0     4.552166
1    22.941273
2    26.410761
dtype: float64
```

```
The reduced Random Forest criteria by majority is:
        Random Forest Criteria 1D
Labels
1                        4.594816
0                       24.680550
2                       76.755249
```



Senstivity analysis on number of layers, i.e., number of leaf nodes

[49]:
```python
# test multiple max_leaf_nodes
from testMaxLeafNodes import *

testDecisionTreeFlag = "regression"
testRandomForestFlag =  "regression"
testObjFlags = [testDecisionTreeFlag, testRandomForestFlag]
leafNodesRange = [2, 20]

testMaxLeafNodes(leafNodesRange, testObjFlags, data)
```



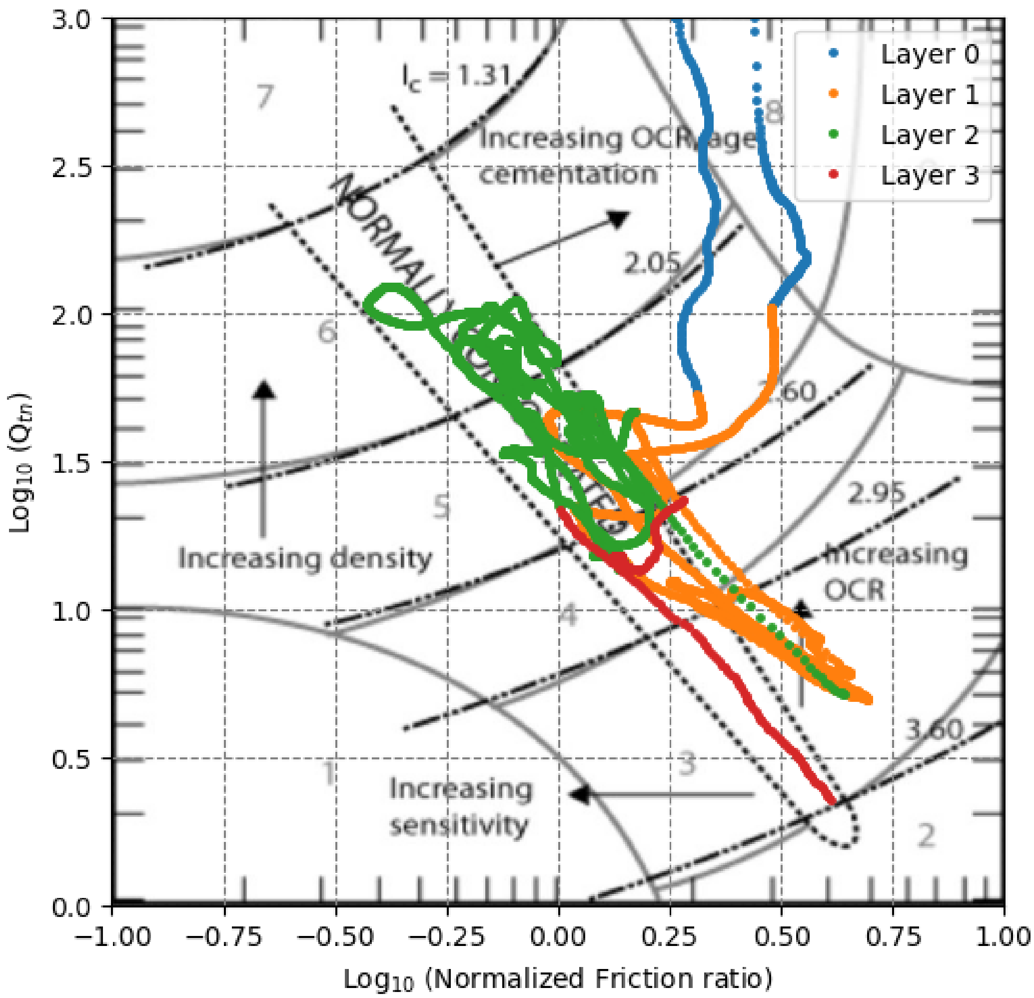
Notes: If only a limit number of CPT soundings is used, need to avoid overfitting.

Plot random forest results on Peter Robertson Soil Behavior Type Chart

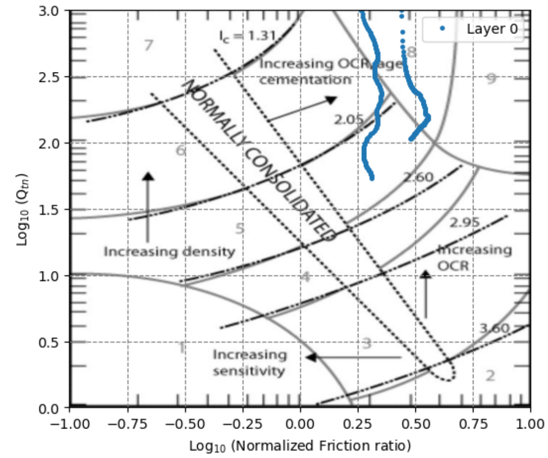
[50]:
```
from applyCriteria import *

# Get strata index of each data point
randomForestStrataIndex = getStrataIndex(randomForestCriteriaReduced, data)

# Plot strataIndex on soil behavior type chart

plotSBTnAllinOne(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
 ↪SBTnImgFileName)
plt.figure()
plotSBTnAllinAll(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
 ↪SBTnImgFileName)
```

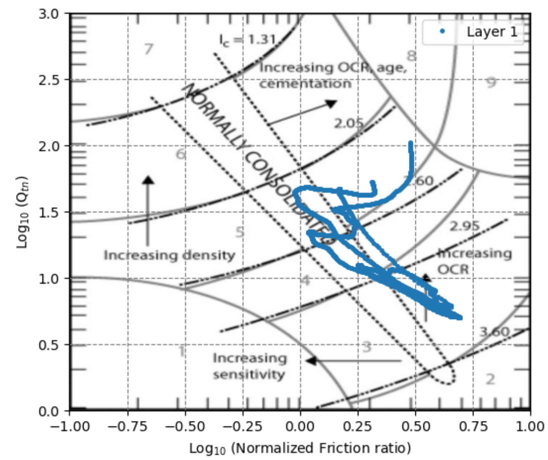
[50]:
```
array([<Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
```

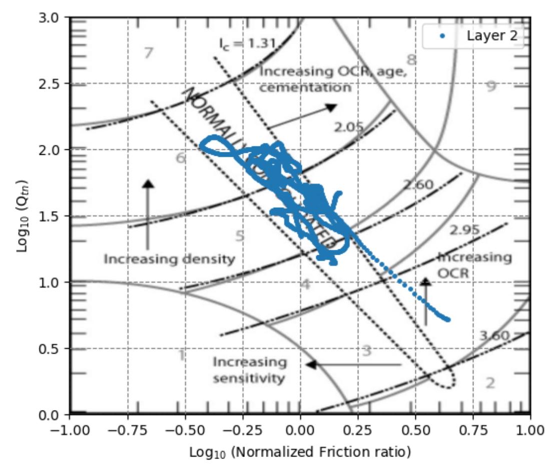
```
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>],
      dtype=object)
```

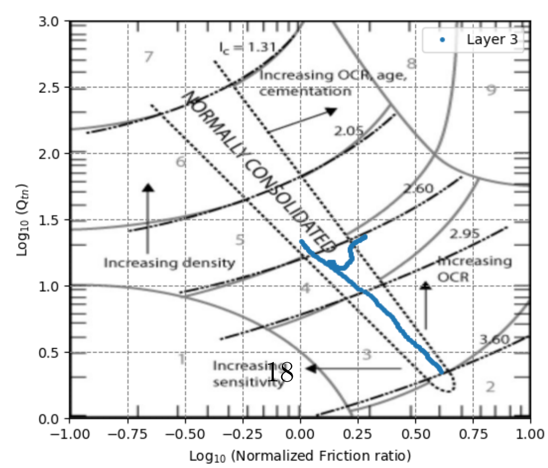



```
<Figure size 640x480 with 0 Axes>
```


Layer 0
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
$\mathrm{Log}_{10}\ (Q_{tn})$
Log_{10} (Normalized Friction ratio)


Layer 1
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
$\mathrm{Log}_{10}\ (Q_{tn})$
Log_{10} (Normalized Friction ratio)


Layer 2
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
$\mathrm{Log}_{10}\ (Q_{tn})$
Log_{10} (Normalized Friction ratio)

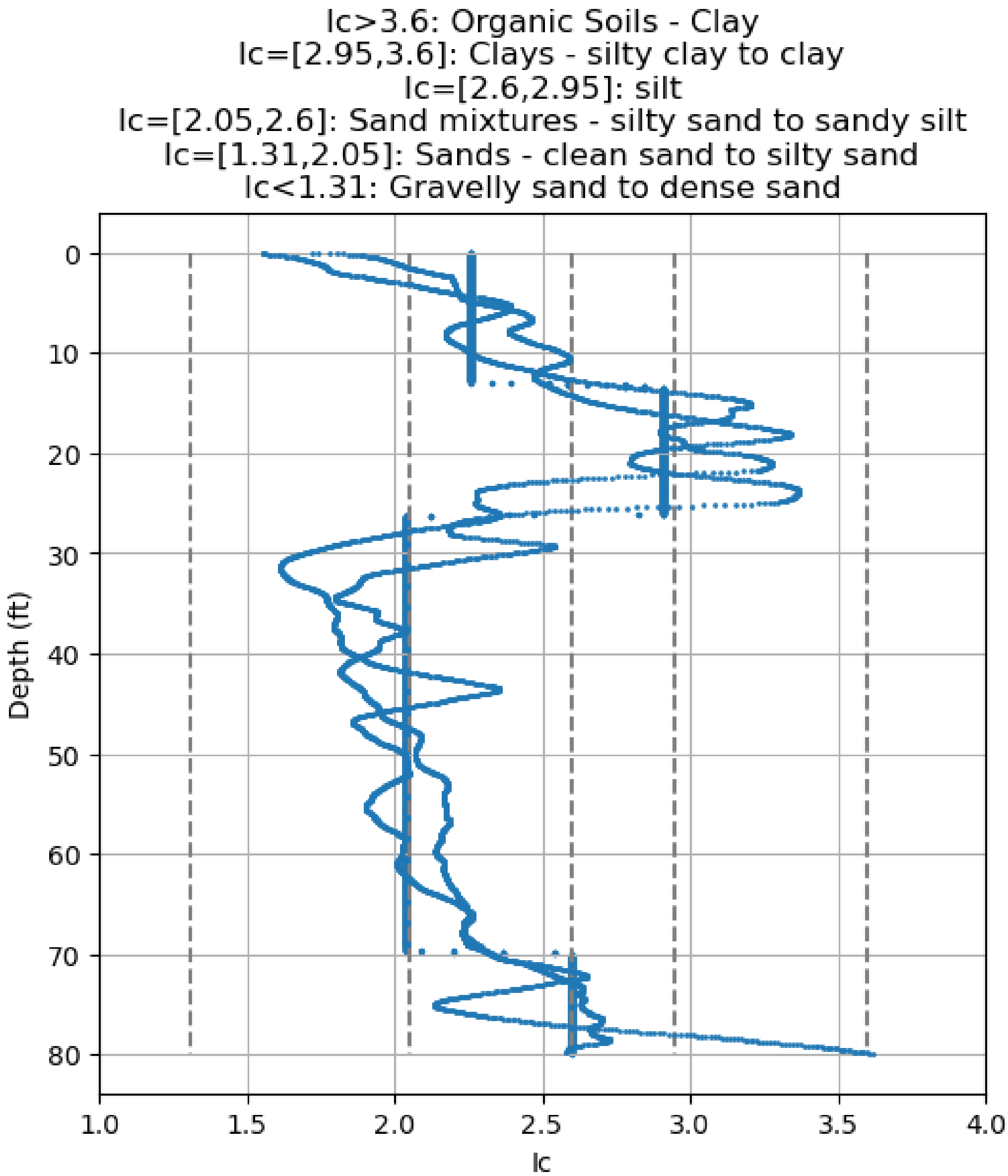

Layer 3
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
$\mathrm{Log}_{10}\ (Q_{tn})$
Log_{10} (Normalized Friction ratio)

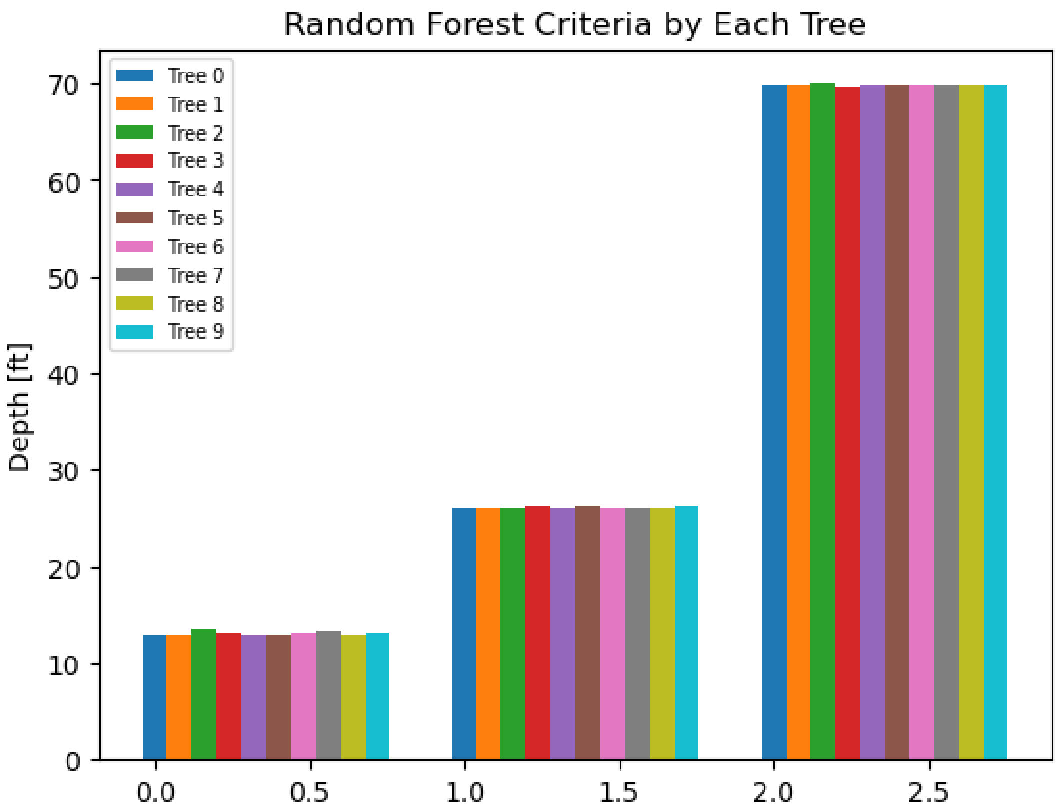
1.8.2 4.2.2 Perform random forest regression on Ic

[51]:
```
# Perform Random Forest on Ic
performRandomForestFlag = "regression"
numberTrees = 10
maxLeafNodes = numberClusters
randomForestInput = [dataIc, numberTrees, maxLeafNodes, randomState]

randomForestObj, randomForestResult =␣
↪performRandomForest(performRandomForestFlag, randomForestInput)

# plot random forest result
IcAxes = plotIc(data.iloc[:,0].to_frame(), Ic)
plotRandomForestResult(randomForestResult, dataIc, IcAxes)

# Extract layer interface depths from non-leaf nodes
randomForestCriteria = getRandomForestCriteria(randomForestObj)

# plot results by each tree using bar chart
plt.figure()
plotRandomForestCriteria(randomForestCriteria)

# reduce randomForestCriteria as median
randomForestCriteriaReduced = randomForestCriteriaMedian(randomForestCriteria)
print()
print(f"The reduced Random Forest criteria by median is:␣
↪\n{randomForestCriteriaReduced}")

# reduce randomForestCriteria as majority
print()
randomForestCriteriaReduced = randomForestCriteriaMajority(randomForestCriteria)
print(f"The reduced Random Forest criteria by majority is:␣
↪\n{randomForestCriteriaReduced}")
```

```
Use Random Forest regressor.
The score by Random Forest: 0.6727148693902694
Below is the results by Random forest regression

The reduced Random Forest criteria by median is:
0    13.106956
1    26.148294
2    69.816273
dtype: float64

The reduced Random Forest criteria by majority is:
```

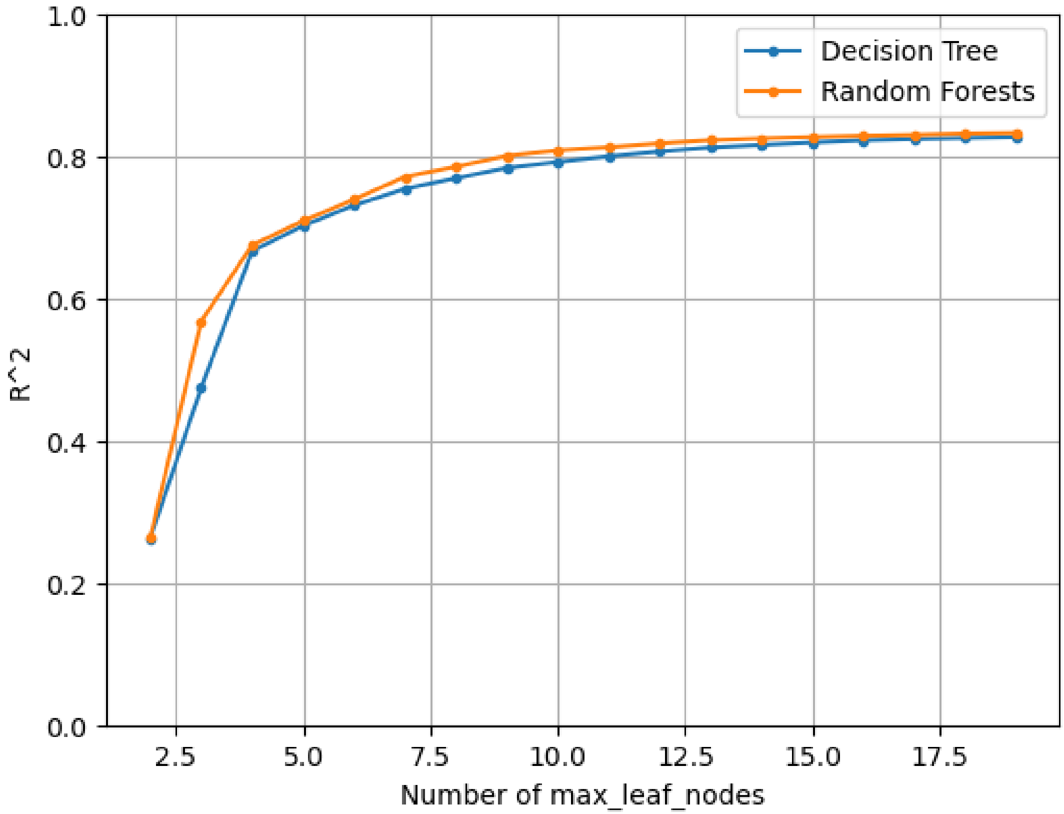
```
        Random Forest Criteria 1D
Labels
0                        13.134842
2                        26.163058
1                        69.826115
```





Senstivity analysis on number of layers, i.e., number of leaf nodes

[52]:
```
# test multiple max_leaf_nodes
from testMaxLeafNodes import *

testDecisionTreeFlag = "regression"
testRandomForestFlag = "regression"
testObjFlags = [testDecisionTreeFlag, testRandomForestFlag]
leafNodesRange = [2, 20]

testMaxLeafNodes(leafNodesRange, testObjFlags, dataIc)
```

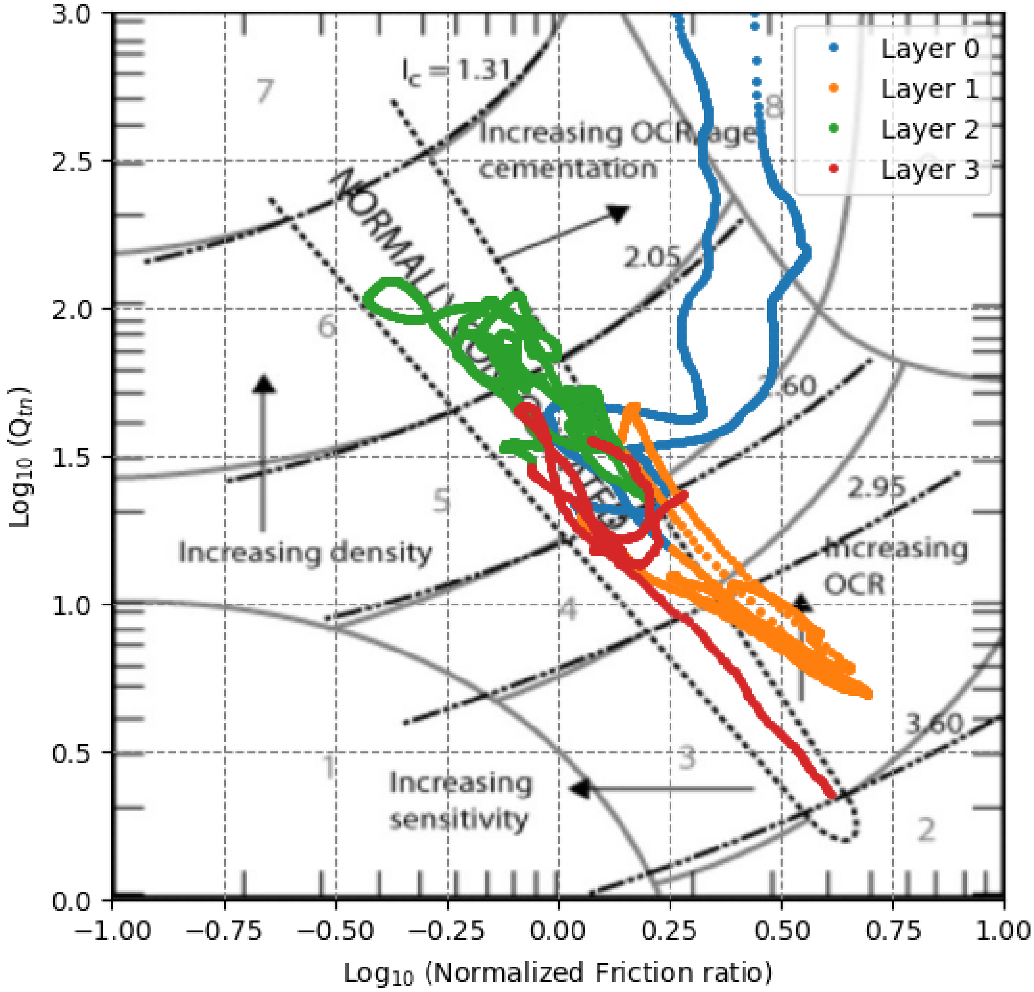


Plot random forest results on Peter Robertson Soil Behavior Type Chart

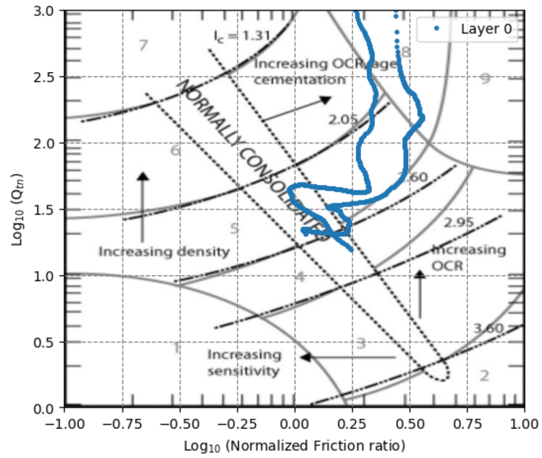
[53]:
```
from applyCriteria import *

# Get strata index of each data point
randomForestStrataIndex = getStrataIndex(randomForestCriteriaReduced, dataIc)

plotSBTnAllinOne(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
 ↪SBTnImgFileName)
plt.figure()
plotSBTnAllinAll(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
 ↪SBTnImgFileName)
```

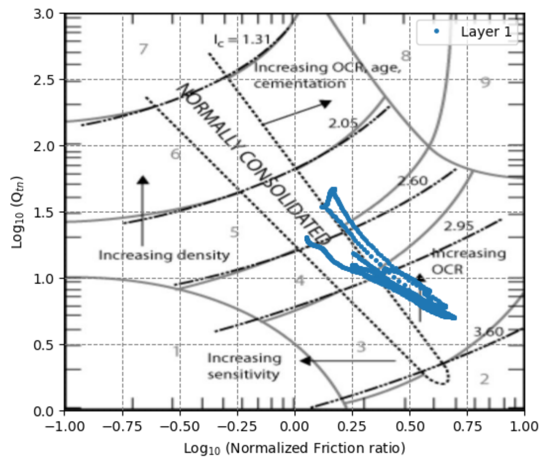
[53]:
```
array([<Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>],
```

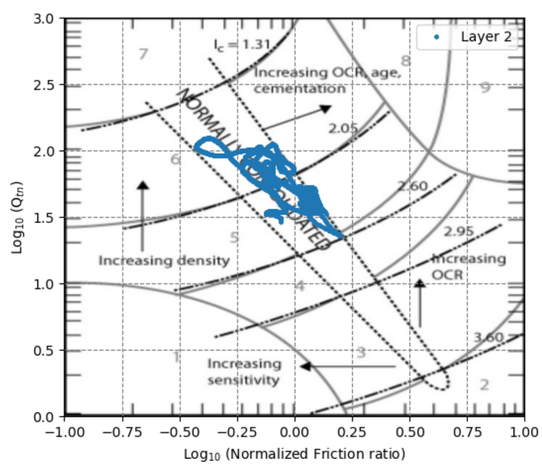
```
dtype=object)
```

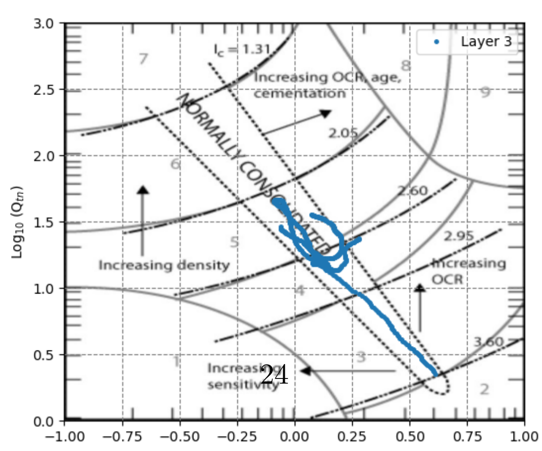



```
<Figure size 640x480 with 0 Axes>
```


Layer 0
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity


Layer 1
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity


Layer 2
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity

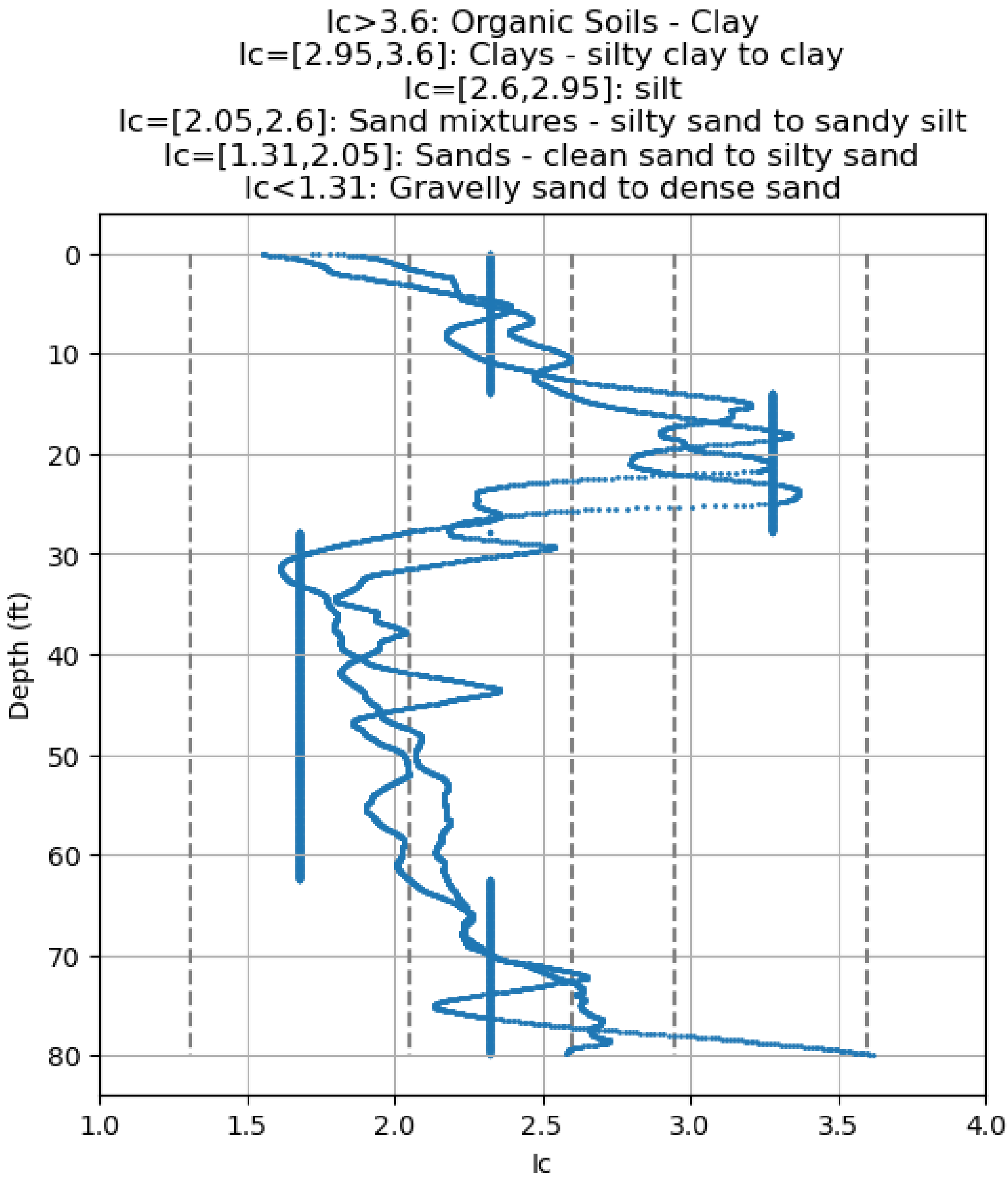

Layer 3
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR

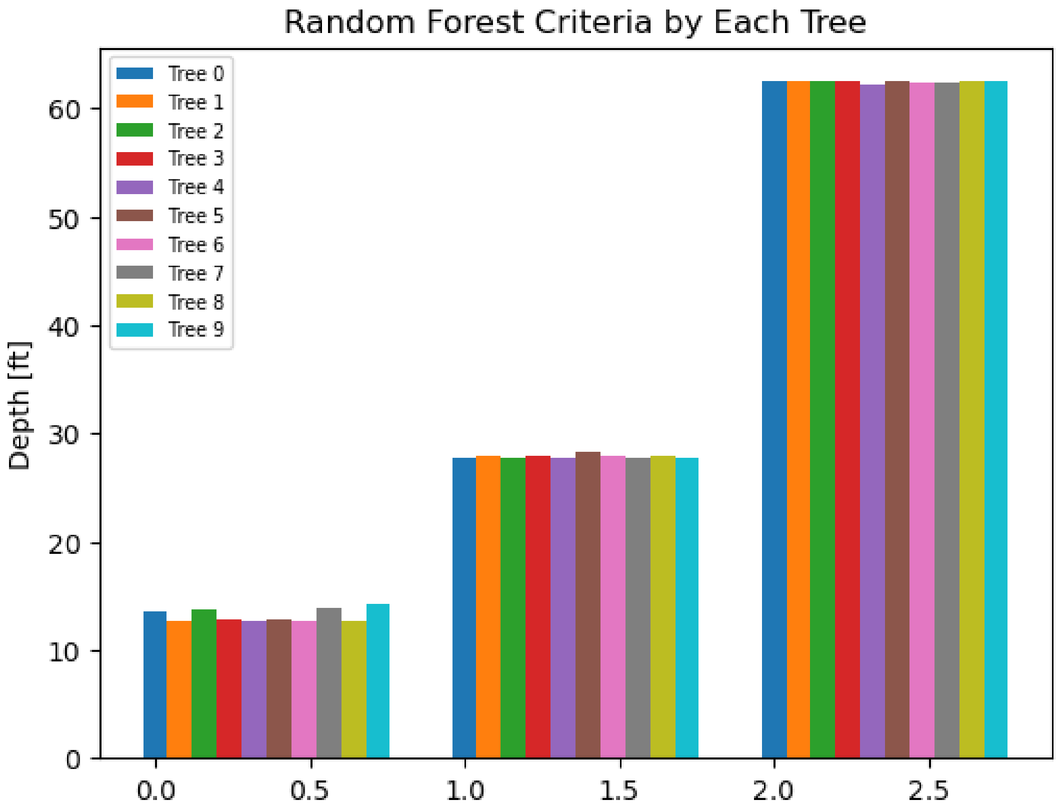
1.8.3 4.2.3 Perform random forest regression on Ic-correlated SBTn

[54]:
```
# Perform random forest regressino on SBTn
performRandomForestFlag = "classification" # when using SBTn, must use␣
↪"classification"
numberTrees = 10
maxLeafNodes = numberClusters
randomForestInput = [dataSBTn1D, numberTrees, maxLeafNodes, randomState]

randomForestObj, randomForestResult =␣
↪performRandomForest(performRandomForestFlag, randomForestInput)

# plot random forest result
IcAxes = plotIc(data.iloc[:,0].to_frame(), Ic)
randomForestResultSBTn1DIc = calculateSBTn1DIc(randomForestResult)
plotRandomForestResult(randomForestResultSBTn1DIc, dataSBTn1D, IcAxes)

# Extract layer interface depths from non-leaf nodes
randomForestCriteria = getRandomForestCriteria(randomForestObj)

# plot results by each tree using bar chart
plt.figure()
plotRandomForestCriteria(randomForestCriteria)

# reduce randomForestCriteria as median
randomForestCriteriaReduced = randomForestCriteriaMedian(randomForestCriteria)
print()
print(f"The reduced Random Forest criteria by median is:␣
↪\n{randomForestCriteriaReduced}")

# reduce randomForestCriteria as majority
print()
randomForestCriteriaReduced = randomForestCriteriaMajority(randomForestCriteria)
print(f"The reduced Random Forest criteria by majority is:␣
↪\n{randomForestCriteriaReduced}")
```

```
Use Random Forest classifier.
The score by Random Forest: 0.6703658035347307
Below is the results by Random forest regression

The reduced Random Forest criteria by median is:
0    12.795276
1    27.829724
2    62.450787
dtype: float64
```

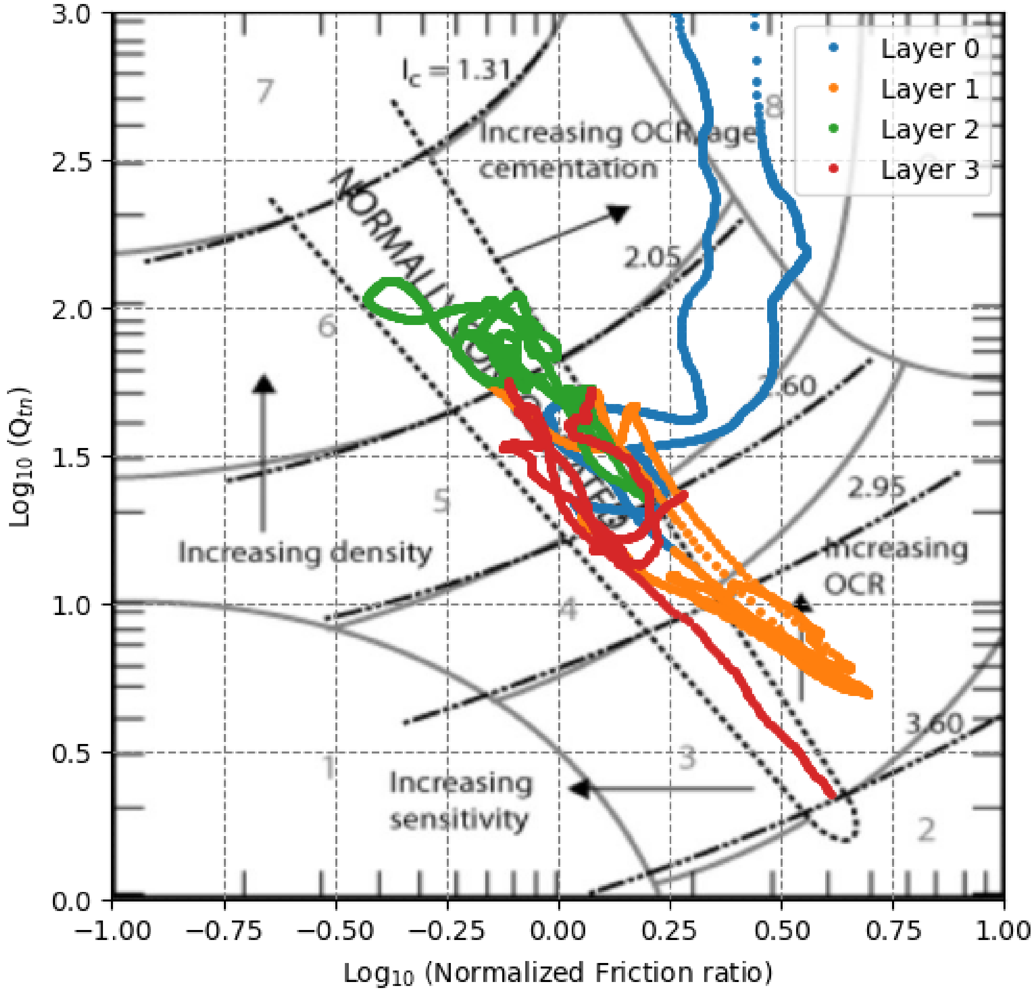
```
The reduced Random Forest criteria by majority is:
        Random Forest Criteria 1D
Labels
0                        13.169291
2                        27.870735
1                        62.406495
```



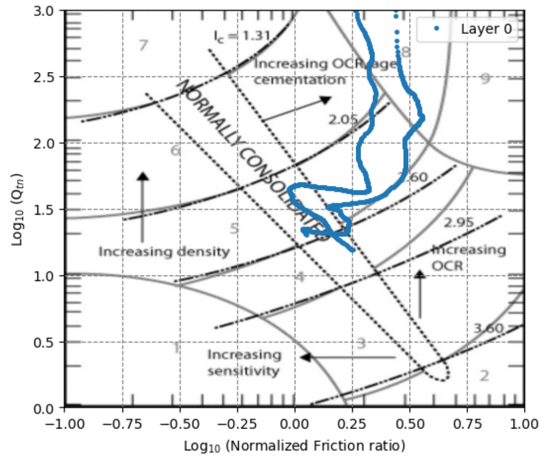


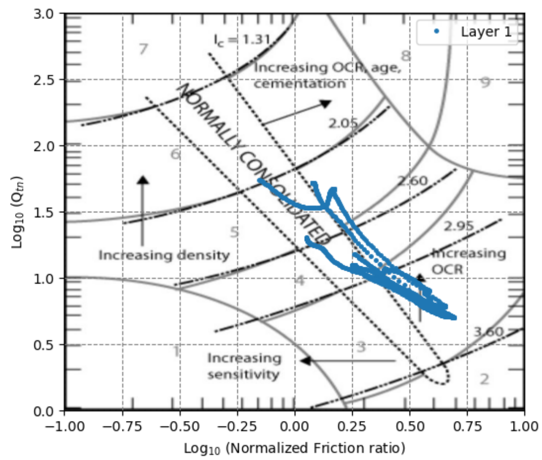
[55]:
```
# Get strata index of each data point
randomForestStrataIndex = getStrataIndex(randomForestCriteriaReduced, dataIc)

plotSBTnAllinOne(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
 ↪SBTnImgFileName)
plt.figure()
plotSBTnAllinAll(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
 ↪SBTnImgFileName)
```

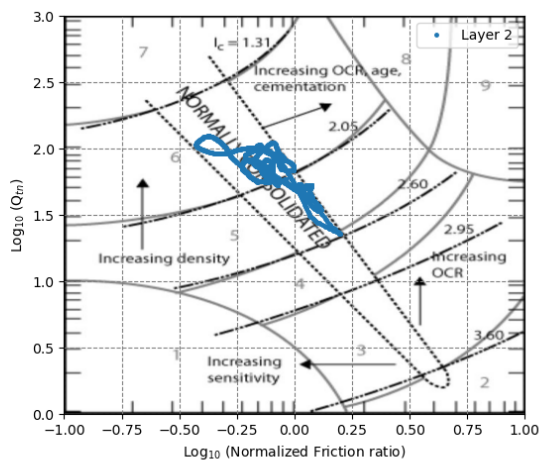
[55]:
```
array([<Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>],
      dtype=object)
```

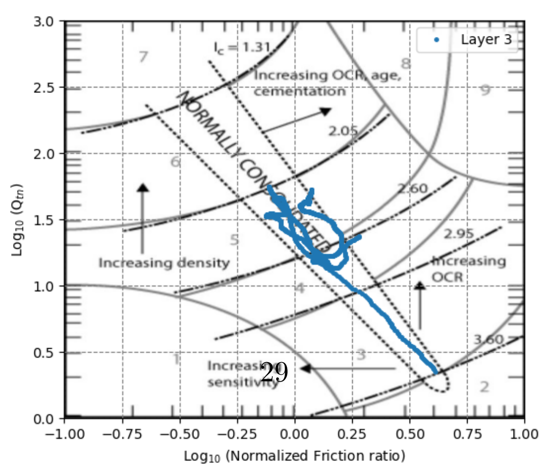



```
<Figure size 640x480 with 0 Axes>
```


Layer 0
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
1 2 3 4 5 6 7 8 9
Log10 (Qtn)
Log10 (Normalized Friction ratio)


Layer 1
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
1 2 3 4 5 6 7 8 9
Log10 (Qtn)
Log10 (Normalized Friction ratio)


Layer 2
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
1 2 3 4 5 6 7 8 9
Log10 (Qtn)
Log10 (Normalized Friction ratio)


Layer 3
$I_c = 1.31$
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity
1 2 3 4 5 6 7 8 9
Log10 (Qtn)
Log10 (Normalized Friction ratio)

1.9 4.3 Agglomerative Clustering

[56]:
```
# PLACE HOLDER
```

1.10 4.4 Clustering based on 2D SBTn chart

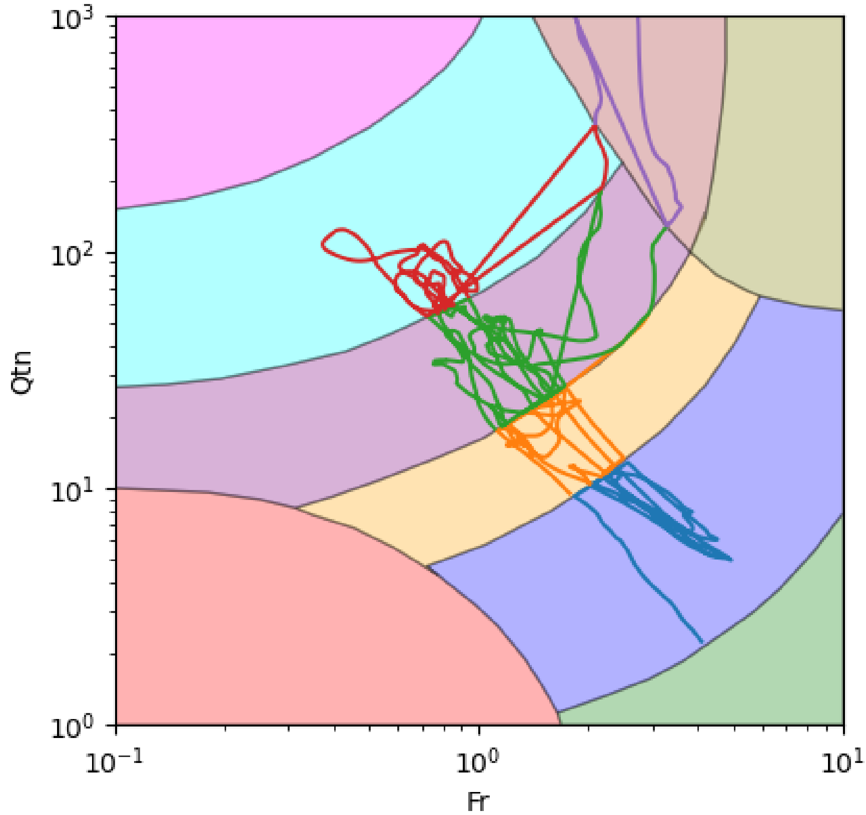
[ ]:
```
import shapely
from shapely.geometry import Point, Polygon
print("External package Shapely is loaded. Version:")
print(shapely.__version__)
```

```
External package Shapely has been loaded. Version:
2.0.7
```

[58]:
```
# Load digitized shape file
SBTnShapeFile = "..\\Digitize SBTn chart\\SBTn zone shapes.csv"

SBTnShapeData = importSBTnChart(SBTnShapeFile)

SBTnShapeCoords = digitizeSBTnChart(SBTnShapeData)
```

[59]:
```
# Determine SBTn 2D zone index for each point
SBTn2D = calculateSBTn2D(Fr, Qtn,  SBTnShapeCoords)

# Verify SBTn2D
verifySBTn2D(Fr, Qtn, SBTn2D, SBTnShapeCoords)

dataSBTn2D = pd.concat([depth, SBTn2D], axis = 1)
```

```
To verify SBTn2D, Points in each zone shall have the same color.
```



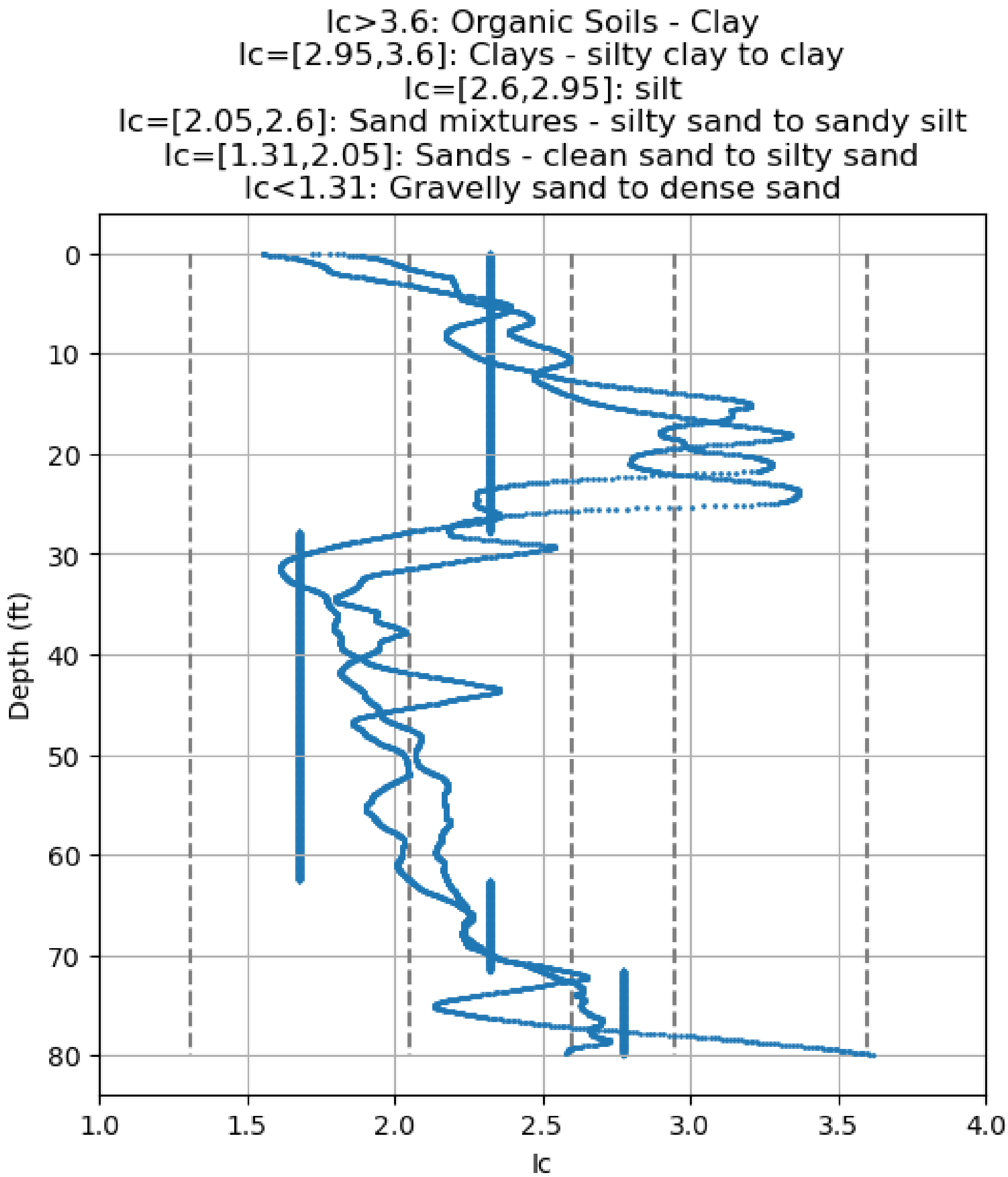

```
[60]: # Perform random forest on 2D SBTn
performRandomForestFlag = "classification" # when using 2D SBTn, must use␣
  ↪"classification"
numberTrees = 10
maxLeafNodes = numberClusters
randomForestInput = [dataSBTn2D, numberTrees, maxLeafNodes, randomState]

randomForestObj, randomForestResult =␣
  ↪performRandomForest(performRandomForestFlag, randomForestInput)

# plot random forest result
IcAxes = plotIc(data.iloc[:,0].to_frame(), Ic)
randomForestResultSBTn1DIc = calculateSBTn1DIc(randomForestResult)
plotRandomForestResult(randomForestResultSBTn1DIc, dataSBTn2D, IcAxes)

# Extract layer interface depths from non-leaf nodes
randomForestCriteria = getRandomForestCriteria(randomForestObj)

# plot results by each tree using bar chart
plt.figure()
```

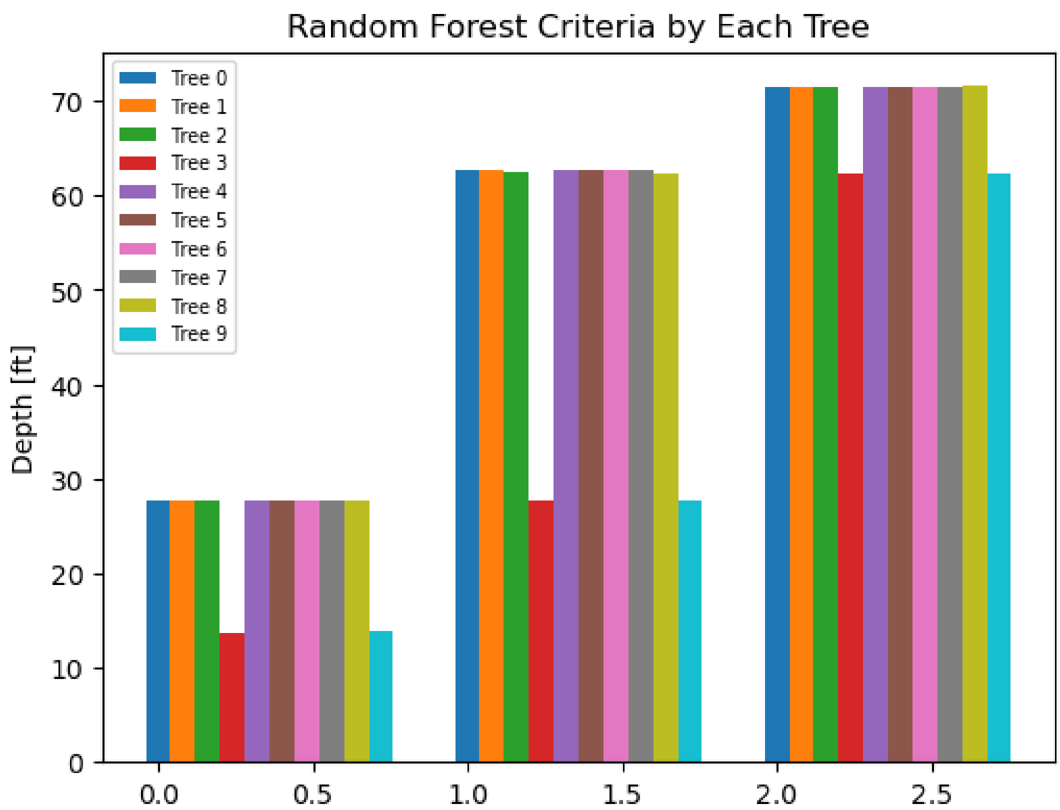
```python
plotRandomForestCriteria(randomForestCriteria)

# reduce randomForestCriteria as median
randomForestCriteriaReduced = randomForestCriteriaMedian(randomForestCriteria)
print()
print(f"The reduced Random Forest criteria by median is:␣
↪\n{randomForestCriteriaReduced}")

# reduce randomForestCriteria as majority
print()
randomForestCriteriaReduced = randomForestCriteriaMajority(randomForestCriteria)
print(f"The reduced Random Forest criteria by majority is:␣
↪\n{randomForestCriteriaReduced}")

# Get strata index of each data point
randomForestStrataIndex = getStrataIndex(randomForestCriteriaReduced,␣
↪dataSBTn2D)

plotSBTnAllinOne(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
↪SBTnImgFileName)
plt.figure()
plotSBTnAllinAll(Fr, Qtn, numberClusters, randomForestStrataIndex,␣
↪SBTnImgFileName)
```

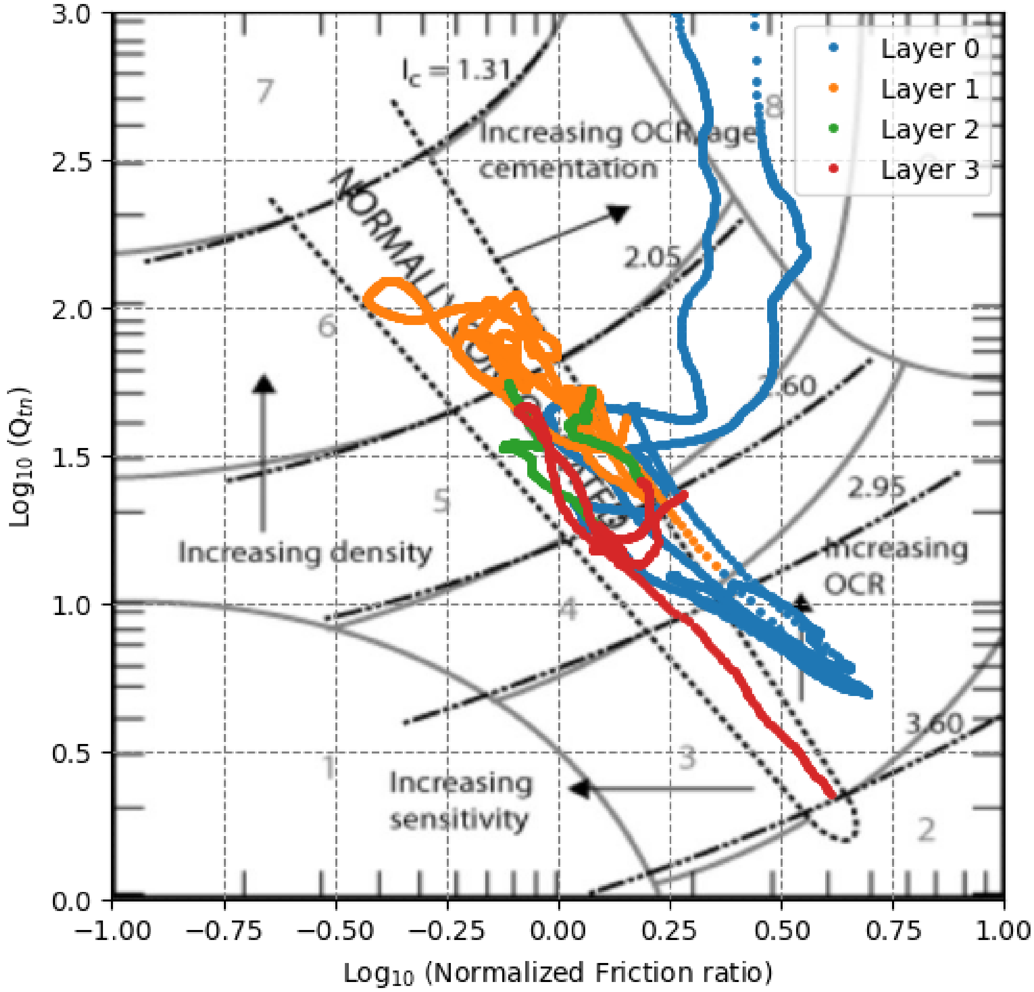
```
Use Random Forest classifier.
The score by Random Forest: 0.6196054254007398
Below is the results by Random forest regression

The reduced Random Forest criteria by median is:
0    27.772309
1    62.582022
2    71.432087
dtype: float64

The reduced Random Forest criteria by majority is:
        Random Forest Criteria 1D
Labels
0                       25.426509
1                       62.491798
2                       71.460793
```

[60]:
```
array([<Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
       <Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>,
```

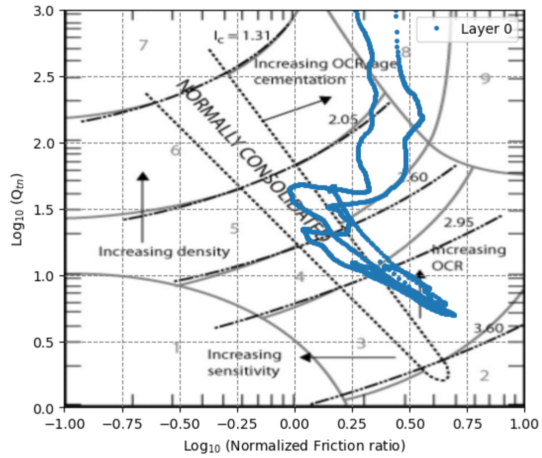
```
<Axes: xlabel='Log$_{10}$ (Normalized Friction ratio)',
ylabel='Log$_{10}$ (Q$_{tn}$)'>],
      dtype=object)
```

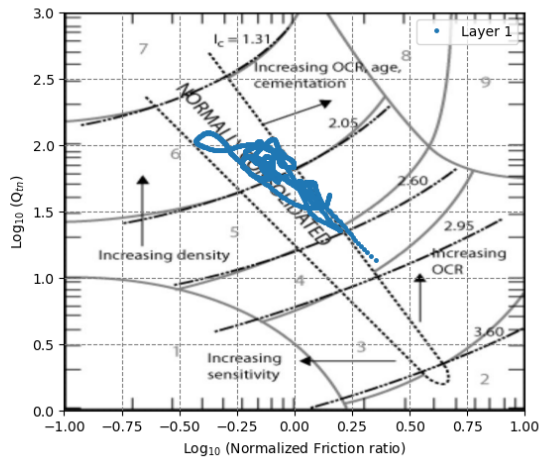


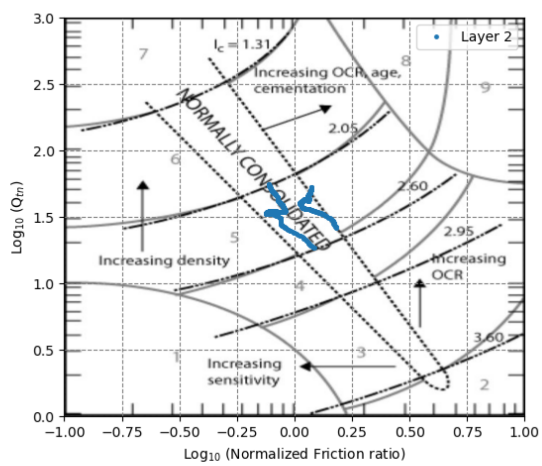

Random Forest Criteria by Each Tree
Tree 0
Tree 1
Tree 2
Tree 3
Tree 4
Tree 5
Tree 6
Tree 7
Tree 8
Tree 9
Depth [ft]
0
10
20
30
40
50
60
70
0.0
0.5
1.0
1.5
2.0
2.5

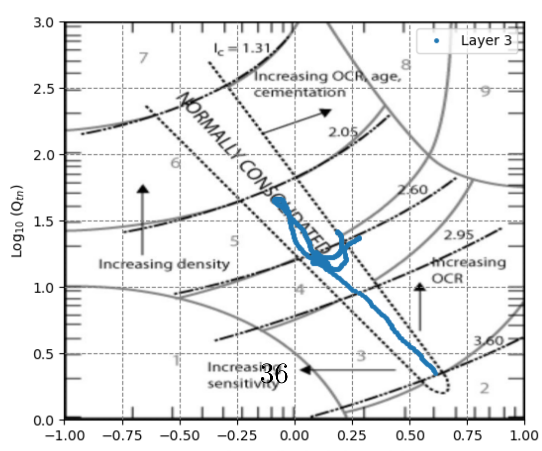


<Figure size 640x480 with 0 Axes>


Layer 0
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity


Layer 1
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity


Layer 2
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
NORMALLY CONSOLIDATED
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR
Increasing sensitivity


Layer 3
Log10 (Qtn)
Log10 (Normalized Friction ratio)
Ic = 1.31
Increasing OCR, age, cementation
2.05
2.60
2.95
3.60
Increasing density
Increasing OCR

2 END