```
# SOIL BEARING CAPACITY PROJECT

# three classes

#Soil_class (1-5) = Field data
#reclassified: 1 -> 1; 2,3,4 -> 2; 5 -> 3

!git clone https://github.com/williamlidberg/Penetration
!pip install geopandas
```

```
Cloning into 'Penetration'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 43 (delta 15), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (43/43), done.
Collecting geopandas
  Downloading geopandas-0.10.2-py2.py3-none-any.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 23.6 MB/s
Requirement already satisfied: pandas>=0.25.0 in /usr/local/lib/python3.7/dist-packag
Collecting pyproj>=2.2.0
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 39.4 MB/s
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages
Collecting fiona>=1.8
  Downloading Fiona-1.8.20-cp37-cp37m-manylinux1_x86_64.whl (15.4 MB)
     |████████████████████████████████| 15.4 MB 47.6 MB/s
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Collecting cligj>=0.5
  Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Installing collected packages: munch, cligj, click-plugins, pyproj, fiona, geopandas
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.20 geopandas-0.10.2
```

```
import geopandas as gpd
import pandas as pd


data = gpd.read_file("/content/Penetration/Penetration_transects_william.shp")


df = pd.DataFrame(data)
df.head(1)
```

	Name	Northing	Easting	Elevation	Waypoint	soil_moist	Average_so	F5c
0	1610	6.331031e+06	485406.80102	271.45832	1610	100.0	20.0	26.

```
#Soil_class (1-5) = Field data
#reclassified: 1 -> 1; 2,3,4 -> 2; 5 -> 3


# reclassifying

df['Soil_class_NEW'] = 1
for i in range(len(df)):
  if df['Soil_class'][i] > 1 and df['Soil_class'][i] < 5:
    df['Soil_class_NEW'][i] = 2
  if df['Soil_class'][i] > 4:
    df['Soil_class_NEW'][i] = 3



df.head(5)
```

```
     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarni
     A value is trying to be set on a copy of a slice from a DataFrame

     See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
# confusion matrix
from sklearn.metrics import confusion_matrix
map = df['Classified']
field = df['Soil_class_NEW']
df_confusion = confusion_matrix(field, map) #(true, predicted)
df_confusion

#The main diagonal gives the correct predictions → soils that were predicted to be class 1
```

```
     array([[  8,   7,   0],
            [275, 535,  19],
            [ 17, 193,   9]])
```

```
from sklearn.metrics import cohen_kappa_score
cohen_kappa_score(field, map)

# no agreement at all between soil moisture levels
```

```
     -0.06503837089380449
```

```
# Kruskal-Wallis test
```

```
from scipy import stats
```

```
classes = df[['Classified', 'Kpa15']]

dry = classes[classes['Classified'] == 1]
mesic = classes[classes['Classified'] == 2]
wet = classes[classes['Classified'] == 3]

drypen = dry['Kpa15']
# values of Kpa15 in sites that are classified as dry

mesicpen = mesic['Kpa15']
wetpen = wet['Kpa15']

stats.kruskal(drypen, mesicpen, wetpen)

# < 0.05 -> significant difference
#soil bearing capacity -> significant difference between classes, we don’t care about the
```

```
     KruskalResult(statistic=145.9121302030011, pvalue=2.0681570036861054e-32)
```

```
# dunn test
```

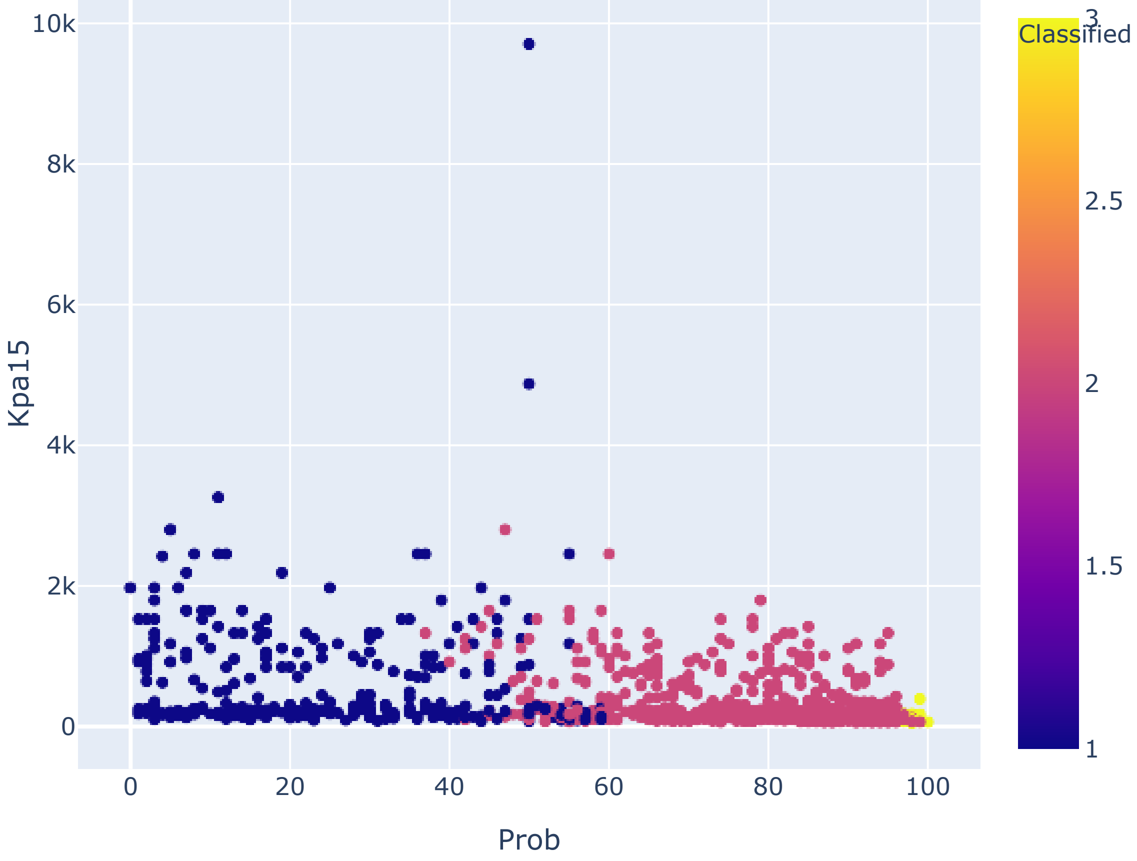
```
!pip install scikit-posthocs
import scikit_posthocs as sp
```

```
Collecting scikit-posthocs
  Downloading scikit-posthocs-0.6.7.tar.gz (43 kB)
     |████████████████████████████████| 43 kB 1.4 MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: scikit-posthocs
  Building wheel for scikit-posthocs (PEP 517) ... done
  Created wheel for scikit-posthocs: filename=scikit_posthocs-0.6.7-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/b8/21/e6/f39794d4a6ee3d3cc5146ca80b5cc
Successfully built scikit-posthocs
Installing collected packages: scikit-posthocs
Successfully installed scikit-posthocs-0.6.7
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnin
  import pandas.util.testing as tm
```

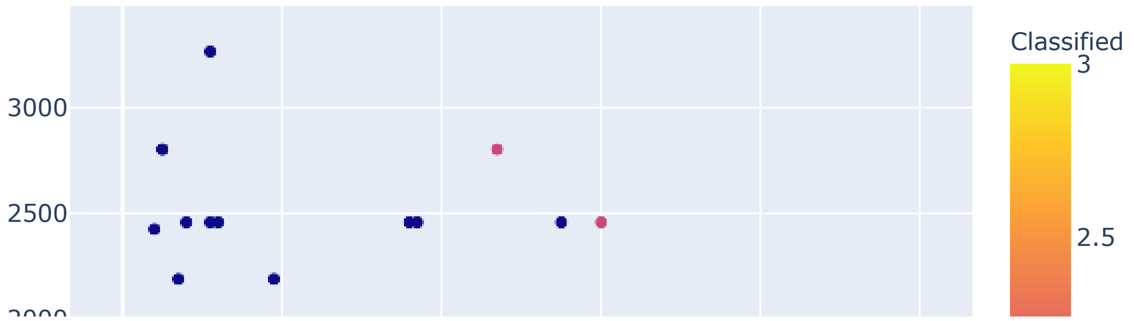
```
dunn_data = [drypen, mesicpen, wetpen]
sp.posthoc_dunn(dunn_data, p_adjust = 'bonferroni')

# the smaller the number, the more significant the difference.
# class 3 has significantly lower soil bearing capacity
# map can be used to predict soil bearing capacity in the field
```

	1	2	3
1	1.000000e+00	1.925358e-27	6.867570e-14
2	1.925358e-27	1.000000e+00	2.481536e-04
3	6.867570e-14	2.481536e-04	1.000000e+00

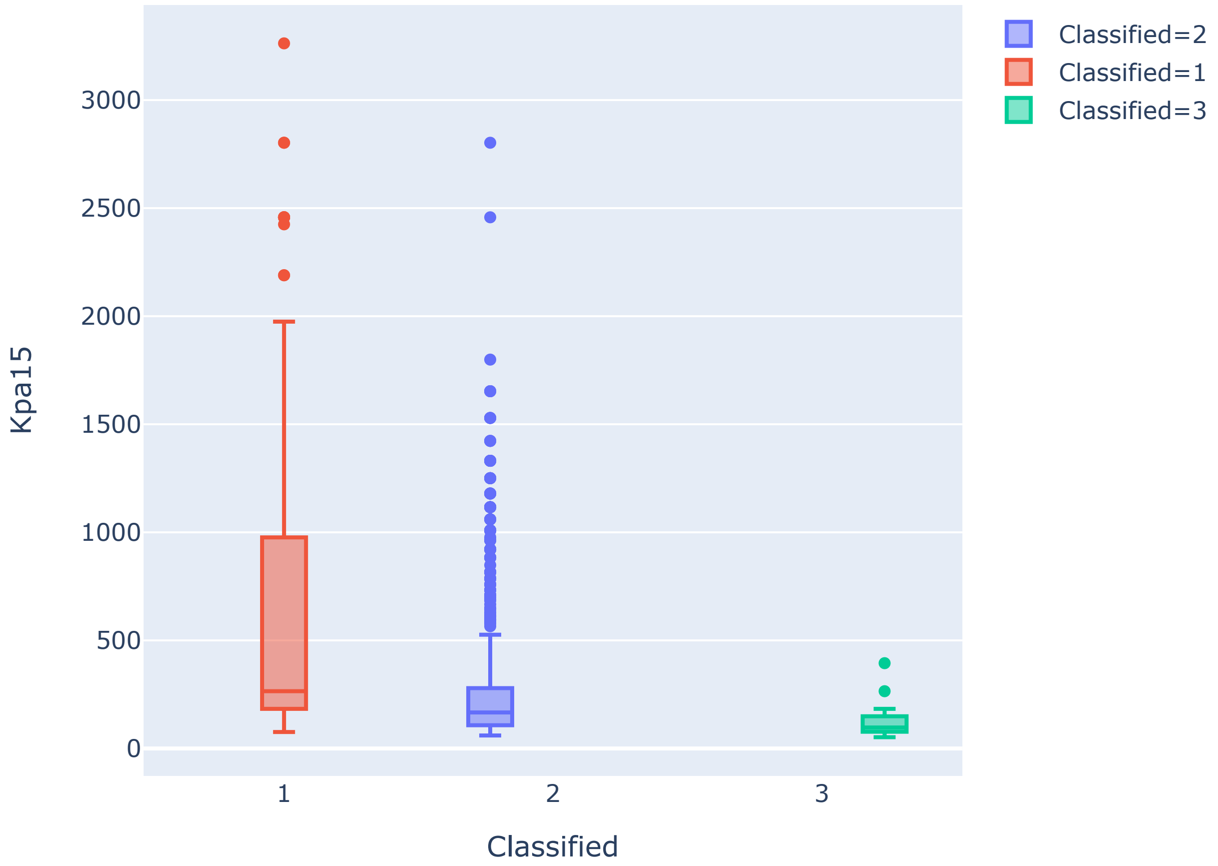
```
### PLOTS ###
# Kpa15 [cone penetration after 15 bonks] and Prob [probability from the SLU moisture map,
import matplotlib.pyplot as plt
import plotly.express as px
fig1 = px.scatter(df, x='Prob', y='Kpa15', color = 'Classified')
fig1.show()
```



```
# removing the two outliers with penetration index > 4000
subset = df.loc[df['Kpa15'] < 4000]
fig2 = px.scatter(subset, x = 'Prob', y = 'Kpa15', color = 'Classified')
fig2.show()

# cone penetration index = resistance to penetration into the terrain
# -> as expected, Kpa15 for higher probabilities of wet soil are lower -> lower bearing ca
```



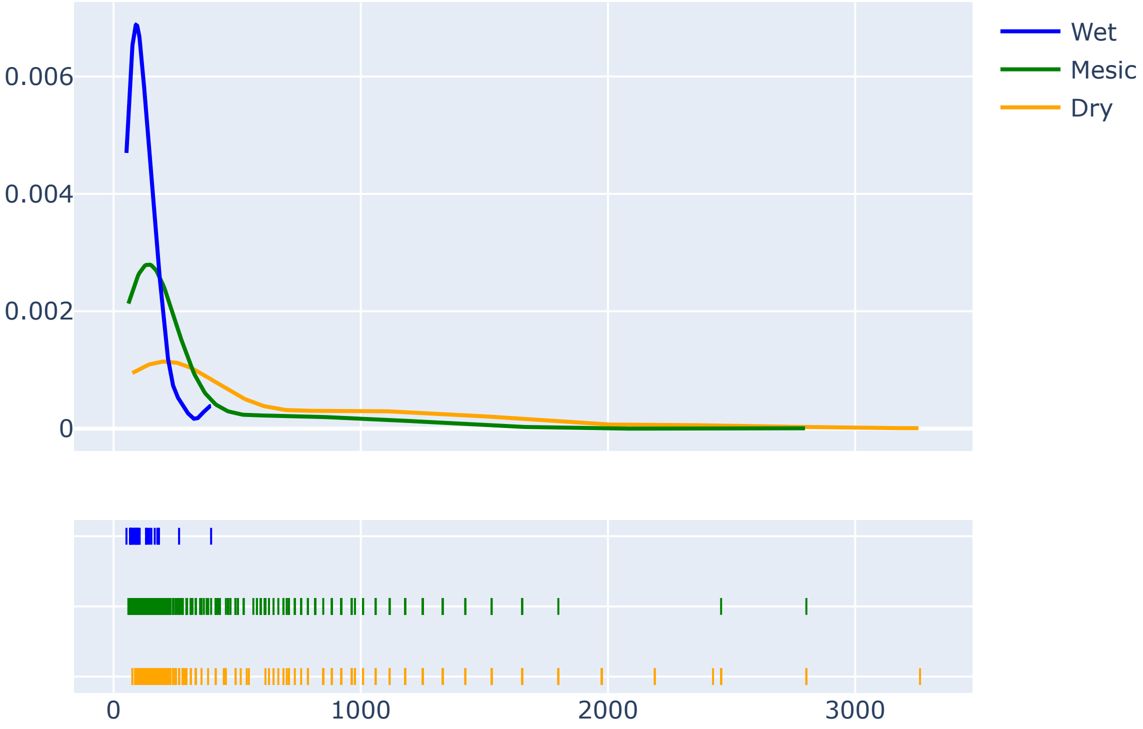
```
# same as a boxplot
fig3 = px.box(subset, x = 'Classified', y = 'Kpa15', color = 'Classified')
fig3.show()
```



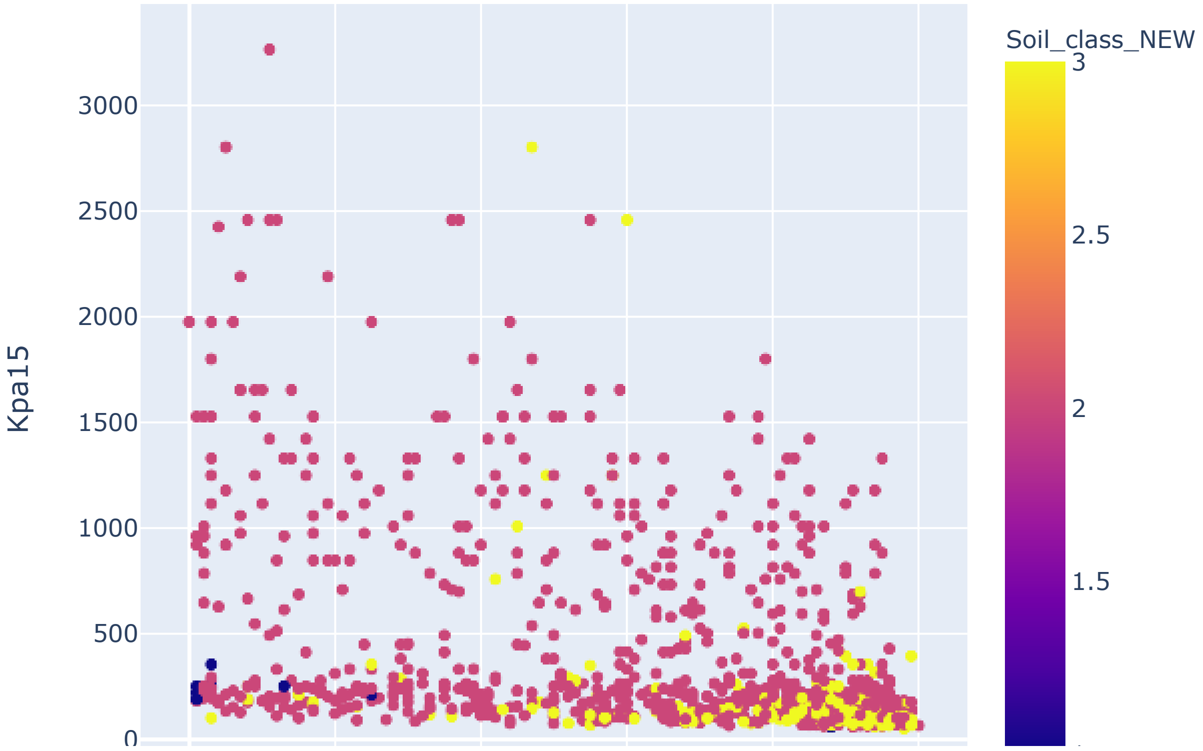
```
# distributions within each soil class on the soil moisture map
import plotly.figure_factory as ff
import numpy as np
```

```
dry = subset[subset['Classified']==1]
mesic = subset[subset['Classified']==2]
wet = subset[subset['Classified']==3]
```

```
#dry.head()
drypen = dry['Kpa15']
mesicpen = mesic['Kpa15']
wetpen = wet['Kpa15']

group_labels = ['Dry', 'Mesic', 'Wet']
hist_data = [drypen, mesicpen, wetpen]
colors = ['orange', 'green', 'blue']
# Create distplot with custom bin_size
fighist = ff.create_distplot(hist_data, group_labels, bin_size=.2,show_hist=False,colors=c
fighist.show()
```

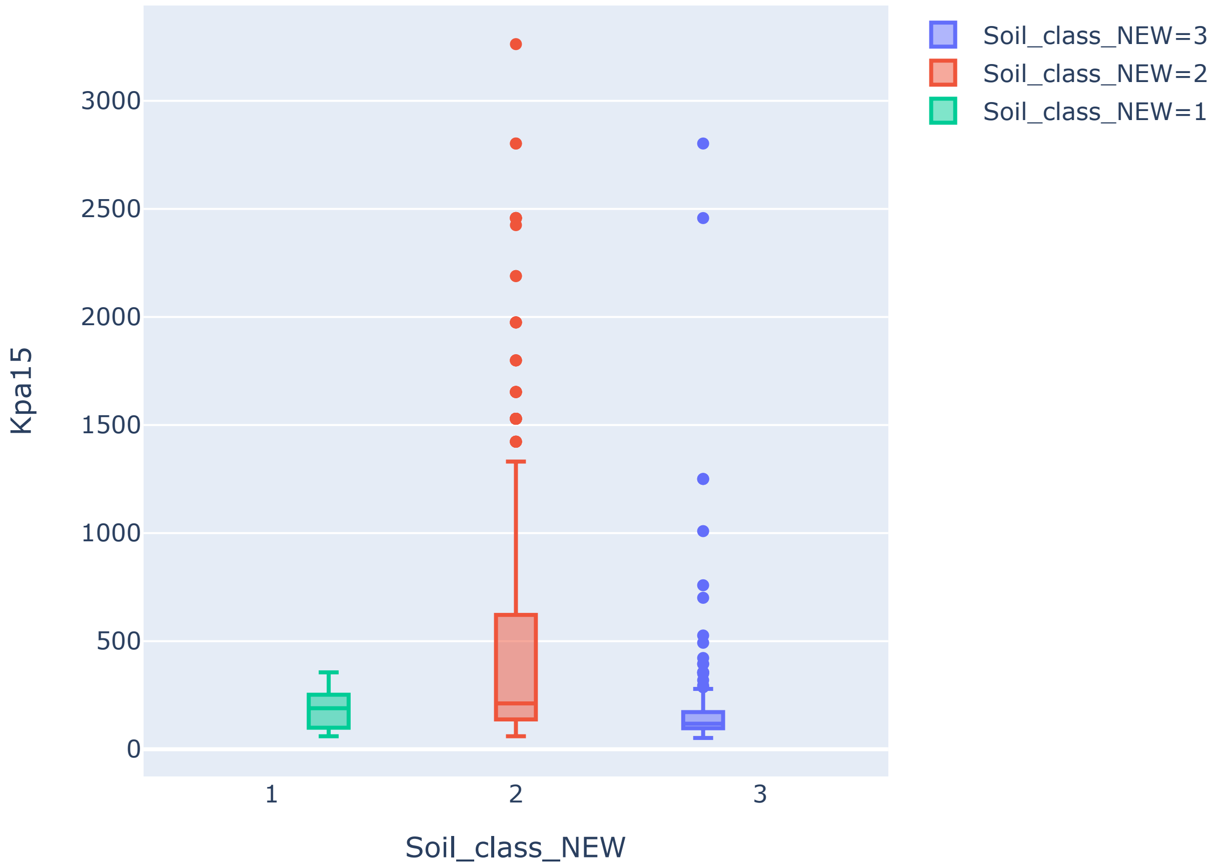


```
### the same plots for Kpa15 and the estimated soil moisture (soil_class; based on vegetat
# using the subset again to exclude the outliers
fig4 = px.scatter(subset, x='Prob', y='Kpa15', color = 'Soil_class_NEW')
fig4.show()
```

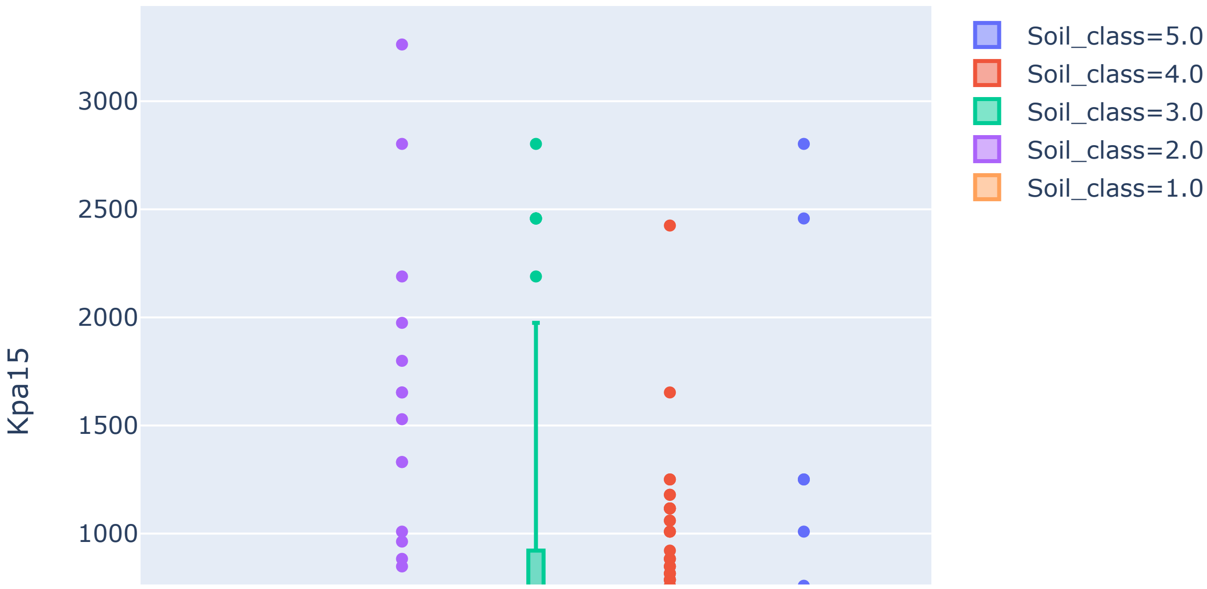


```
# with the original data before reclassification

fig5 = px.scatter(subset, x='Prob', y='Kpa15', color = 'Soil_class')
fig5.show()
```

```
# same as a boxplot after reclassification
fig6 = px.box(subset, x = 'Soil_class_NEW', y = 'Kpa15', color = 'Soil_class_NEW')
fig6.show()
```

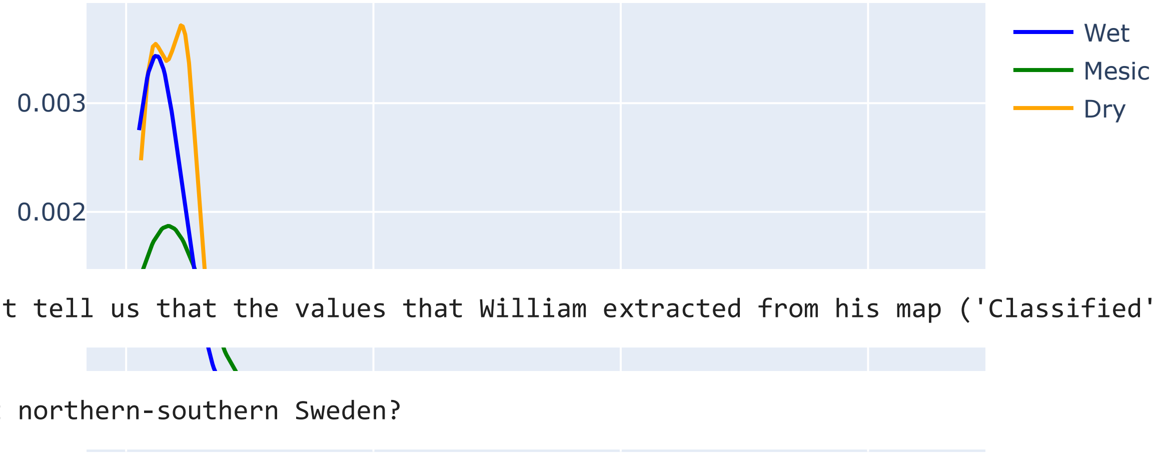


```
# same as a boxplot before reclassification
fig7 = px.box(subset, x = 'Soil_class', y = 'Kpa15', color = 'Soil_class')
fig7.show()

# the wet soil class (5) has some very high values for the cone penetration index
# human error?
# makes it difficult/impossible to reclassify into three classes
```

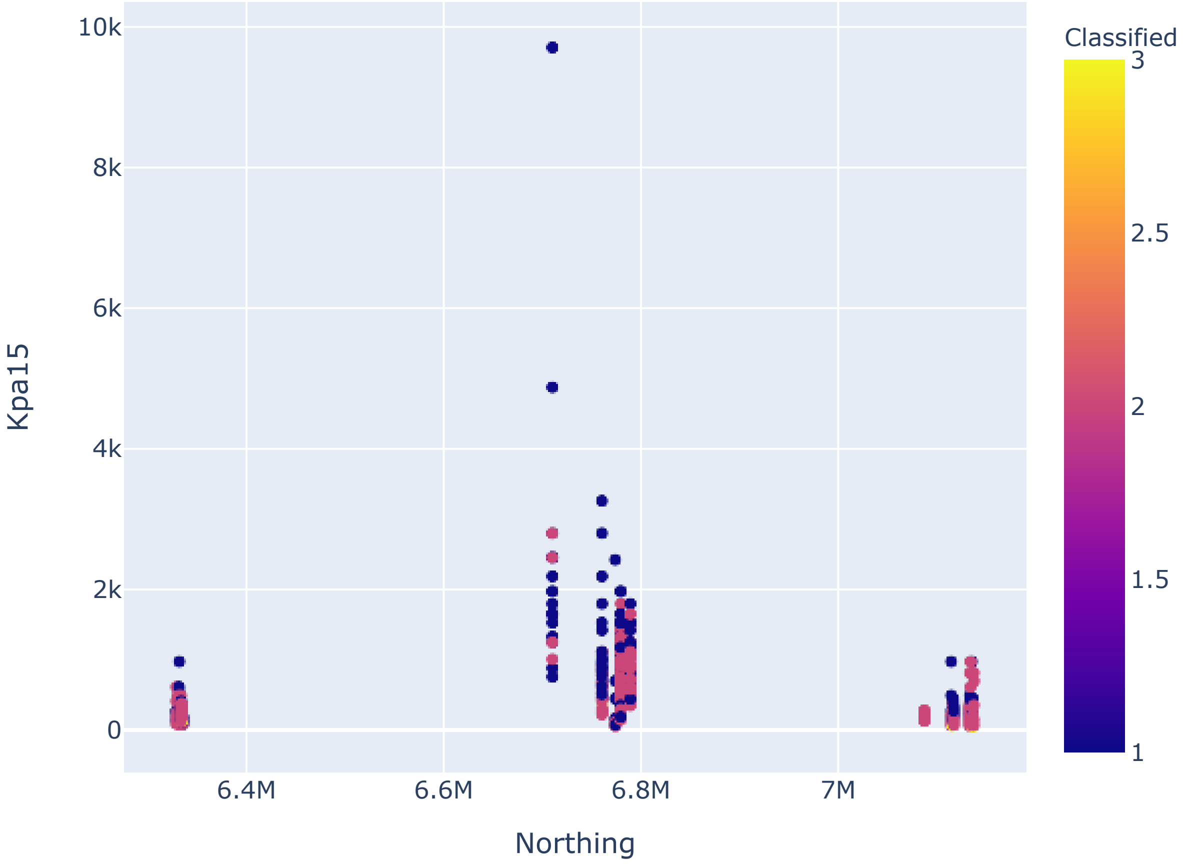


```
# the mean Kpa15 should be highest for soil class 2 though, not 3
```

```
dry = subset[subset['Soil_class_NEW']==1]
mesic = subset[subset['Soil_class_NEW']==2]
wet = subset[subset['Soil_class_NEW']==3]
#dry.head()
drypen = dry['Kpa15']
mesicpen = mesic['Kpa15']
wetpen = wet['Kpa15']

group_labels = ['Dry', 'Mesic', 'Wet']
hist_data = [drypen, mesicpen, wetpen]
colors = ['orange', 'green', 'blue']
# Create distplot with custom bin_size
fighist = ff.create_distplot(hist_data, group_labels, bin_size=.2,show_hist=False,colors=c
fighist.show()

# -> the distribution of William's values makes more sense
```



```
# does that tell us that the values that William extracted from his map ('Classified') mak

# gradient northern-southern Sweden?

fig8 = px.scatter(df, x='Northing', y='Kpa15', color = 'Classified')
fig8.show()
```



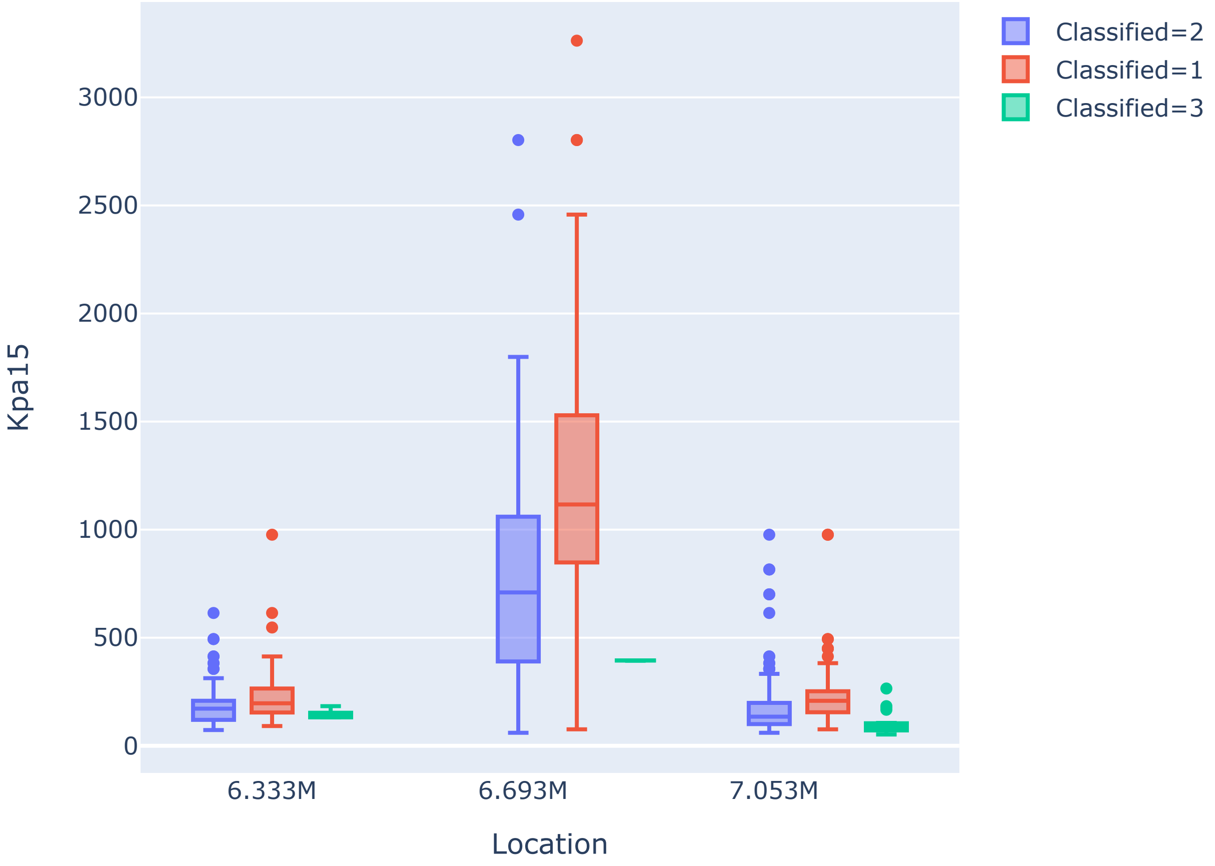
```
# three different regions: north, south and middle
# calculate average location and then make boxplots

df['Location'] = 1
```

```
df['Location2'] = ''
liste_south = []
liste_middle = []
liste_north = []

# choose which locations count as north/middle/south
for i in range(len(df)):
  if df['Northing'][i] > 6.3e+06 and df['Northing'][i] < 6.4e+06:
    liste_south.append(df['Northing'][i])
  if df['Northing'][i] > 6.4e+06 and df['Northing'][i] < 7.0e+06:
    liste_middle.append(df['Northing'][i])
  if df['Northing'][i] > 7.0e+06:
    liste_north.append(df['Northing'][i])

averagelocation_south = sum(liste_south)/len(liste_south)
averagelocation_middle = sum(liste_middle)/len(liste_middle)
averagelocation_north = sum(liste_north)/len(liste_north)


print(averagelocation_south)
print(averagelocation_middle)
print(averagelocation_north)

# create new column with mean location per region
for i in range(len(df)):
  if df['Northing'][i] > 6.3e+06 and df['Northing'][i] < 6.4e+06:
    df['Location'][i] = averagelocation_south
    df['Location2'][i] = 'south'
  if df['Northing'][i] > 6.4e+06 and df['Northing'][i] < 7.0e+06:
    df['Location'][i] = averagelocation_middle
    df['Location2'][i] = 'middle'
  if df['Northing'][i] > 7.0e+06:
    df['Location'][i] = averagelocation_north
    df['Location2'][i] = 'north'

df.head(2)
```

```
6332692.99765012
6770027.667712996
7130231.225802708
```

	Name	Northing	Easting	Elevation	Waypoint	soil_moist	Average_so	F5c
0	1610	6.331031e+06	485406.80102	271.45832	1610	100.0	20.0	26.
1	1611	6.331028e+06	485418.20319	252.01908	1611	100.0	50.0	18.

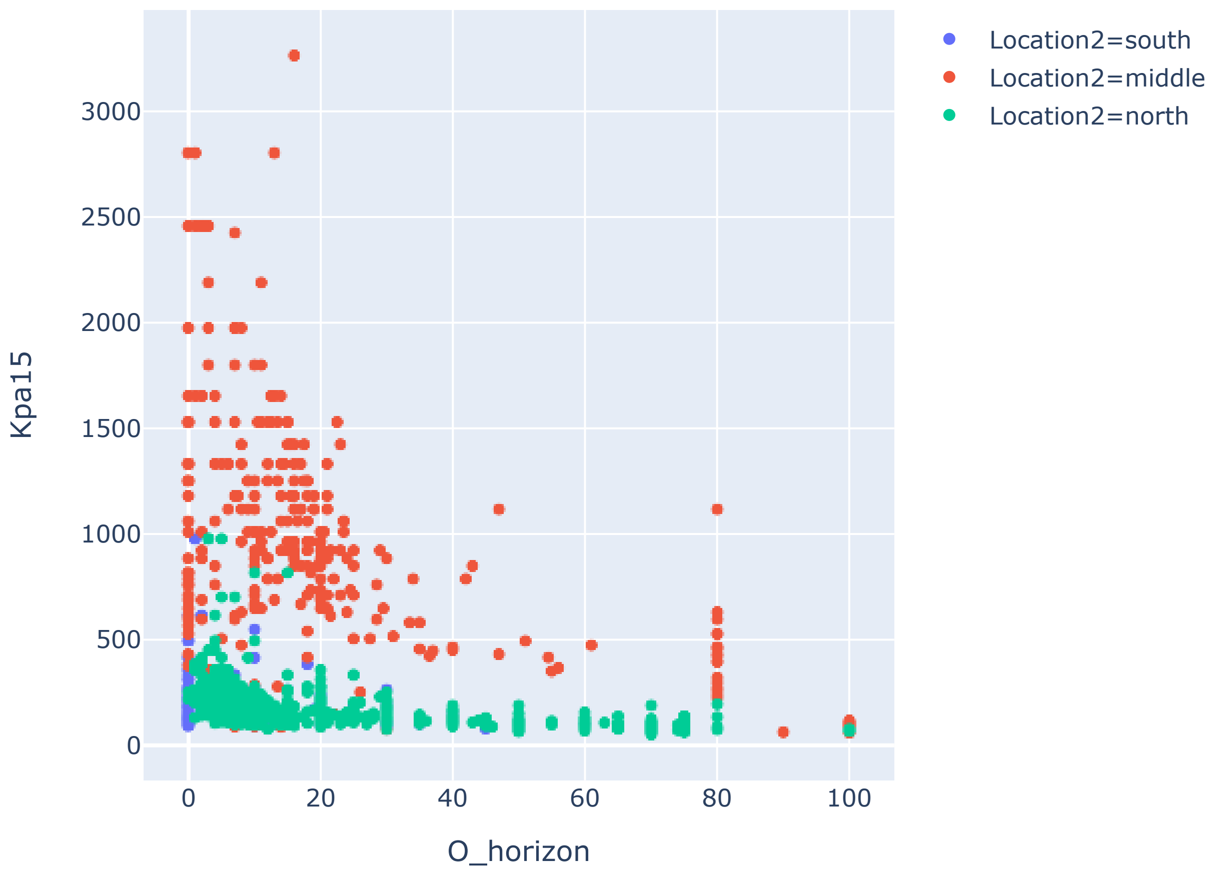
```
# removing penetration index > 4000 again
subset = df.loc[df['Kpa15'] < 4000]
fig9 = px.box(subset, x='Location', y='Kpa15', color = 'Classified')
fig9.show()

# mean Kpa15 is highest in the middle of Sweden -> highest soil bearing capacity there
# lowest soil bearing capacity in the north on soils that were classified as wet according
```



```
# What percentage is classified as "wet"(3)?
# Can we recommend to just avoid those areas?

percentage_wet = (df['Classified'].value_counts()[3]/df['Classified'].count())*100
print(f'In all of Sweden, {percentage_wet:.2f} % are classified as wet.')

# average location north: averagelocation_north
df.groupby(['Location2']).size()
# locations in the south: 162
# locations in the middle: 303
# locations in the north: 598

counter = 0
for i in range(len(df)):
  if (df['Location2'][i] == 'north') and (df['Classified'][i] == 3):
    counter = counter + 1
counter # number of wet locations in the north
percentage_wet_north = counter/598
print(f'Wet soils in the northern parts of Sweden had the lowest mean soil bearing capacit
```

```
In all of Sweden, 2.63 % are classified as wet.
Wet soils in the northern parts of Sweden had the lowest mean soil bearing capacity.
```

```
subset = df.loc[df['Kpa15'] < 4000]
fig10 = px.scatter(subset, x='O_horizon', y='Kpa15', color = 'Location2')
fig10.show()
# the shallower the O-horizon, the higher the bearing capacity
```



✓ 1 s Abgeschlossen um 08:55 ● ×