aml-hw2-q1-n-1

October 22, 2023

Importing Necessary Libraries

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1. Display the statistical values for each of the attributes, along with visualizations (e.g., histogram) of the distributions for each attribute. Are there any attributes that might require special treatment? If so, what special treatment might they require?

[ ]:
```
# Load the data
data = pd.read_excel("aggregateRockData-1.xlsx", usecols=[1],names=["rock␣
↪category"])
features = pd.read_csv("norm540.txt", sep='\t', usecols=list(range(3, 22)))
```

[ ]:
```
# Assign meaningful column names
attribute_names = [
    "Porphyritic texture",
    "Presence of holes",
    "Salient green hue",
    "Pegmatitic texture",
    "Conchoidal fracture",
    "Angular fragments",
    "Rounded fragments",
    "Straight stripes",
    "Curved stripes",
    "Physical layers",
    "Veins",
    "Oily/shimmery texture",
    "Splotchy texture",
    "Single translucent crystal",
    "Multiple cubic crystals",
    "Sandy texture",
    "Fragments (disjunctive)",
    "Stripes (disjunctive)",
    "Crystals (disjunctive)"
]
```

```
features.columns = attribute_names
```

Statistical descriptions

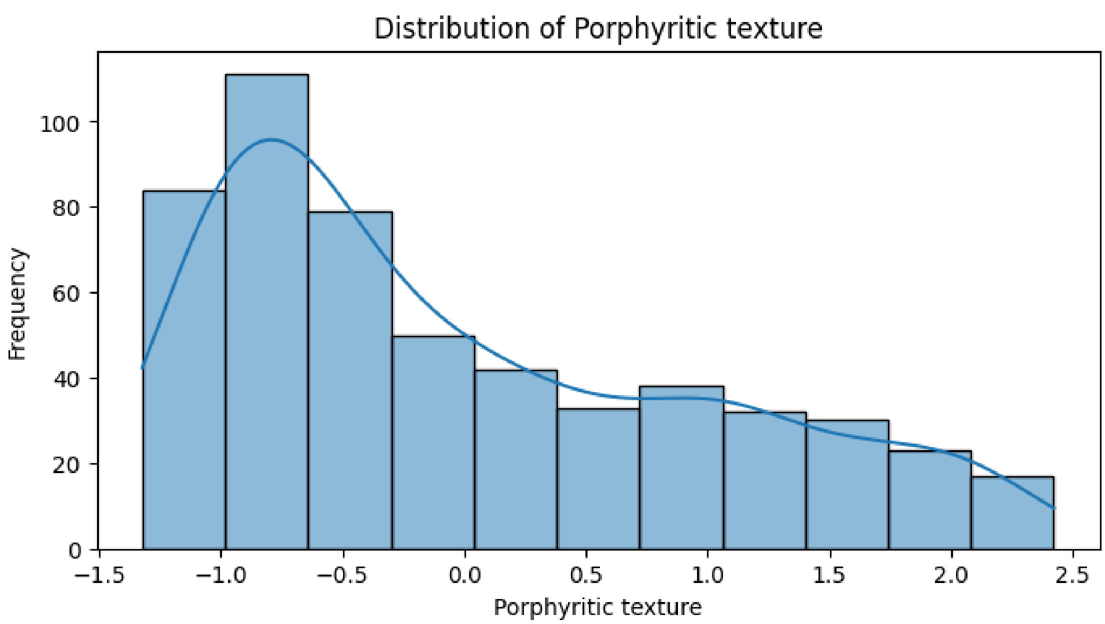
[ ]:
```
# Display summary statistics
statistics = features.describe()
print(statistics)
```

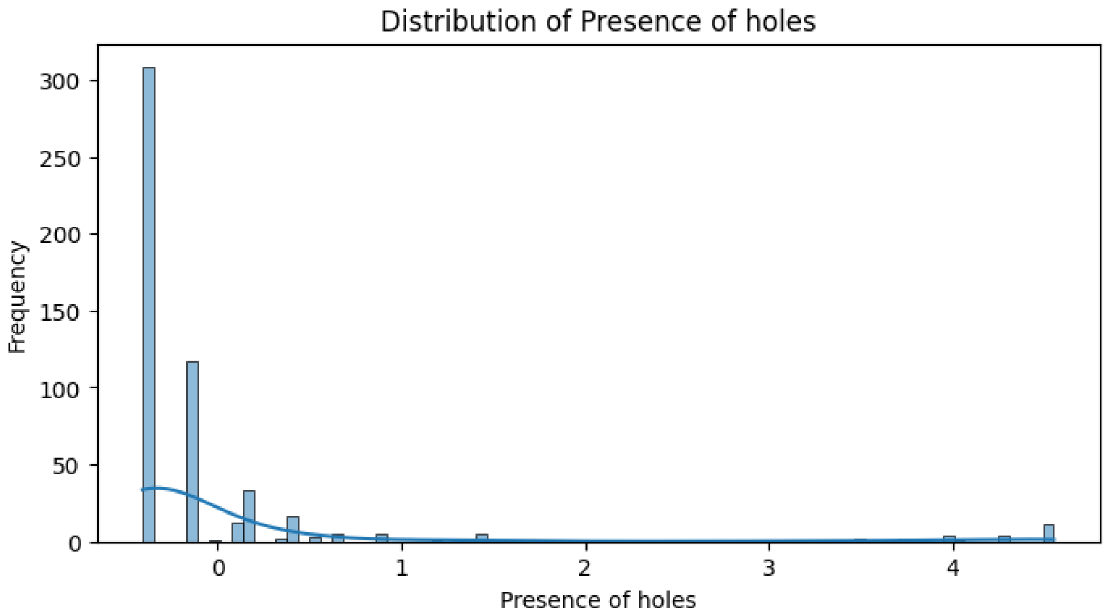
```
       Porphyritic texture  Presence of holes  Salient green hue  \
count           539.000000         539.000000         539.000000
mean             -0.003136           0.000296           0.001199
std               0.998267           1.000905           1.000541
min              -1.321491          -0.407623          -1.187950
25%              -0.823647          -0.407623          -0.761505
50%              -0.300910          -0.407623          -0.375197
75%               0.757010          -0.159688           0.585557
max               2.422299           4.551072           2.750390

       Pegmatitic texture  Conchoidal fracture  Angular fragments  \
count          539.000000           539.000000         539.000000
mean             0.000467             0.001131          -0.001076
std              1.000870             1.000583           1.000616
min             -1.322715            -1.248012          -0.436004
25%             -0.804631            -0.699145          -0.436004
50%             -0.182929            -0.271538          -0.436004
75%              0.576929             0.360298          -0.182021
max              4.175892             3.813059           4.643652

       Rounded fragments  Straight stripes  Curved stripes  Physical layers  \
count         539.000000        539.000000      539.000000       539.000000
mean           -0.000696          0.000654        0.000483         0.001408
std             1.000798          1.000813        1.000866         1.000393
min            -0.405184         -0.352386       -0.260224        -0.759128
25%            -0.405184         -0.352386       -0.260224        -0.759128
50%            -0.405184         -0.352386       -0.260224        -0.299173
75%            -0.145018         -0.352386       -0.260224         0.390760
max             4.798130          4.888957        5.862693         3.610446

            Veins  Oily/shimmery texture  Splotchy texture  \
count  539.000000             539.000000        539.000000
mean     0.000026               0.001003         -0.001756
std      1.000929               1.000657          1.000095
min     -0.512160              -0.540653         -0.846887
25%     -0.512160              -0.540653         -0.846887
50%     -0.512160              -0.540653         -0.249084
75%     -0.013842              -0.165887          0.348718
max      8.457556               3.207009          4.832237
```

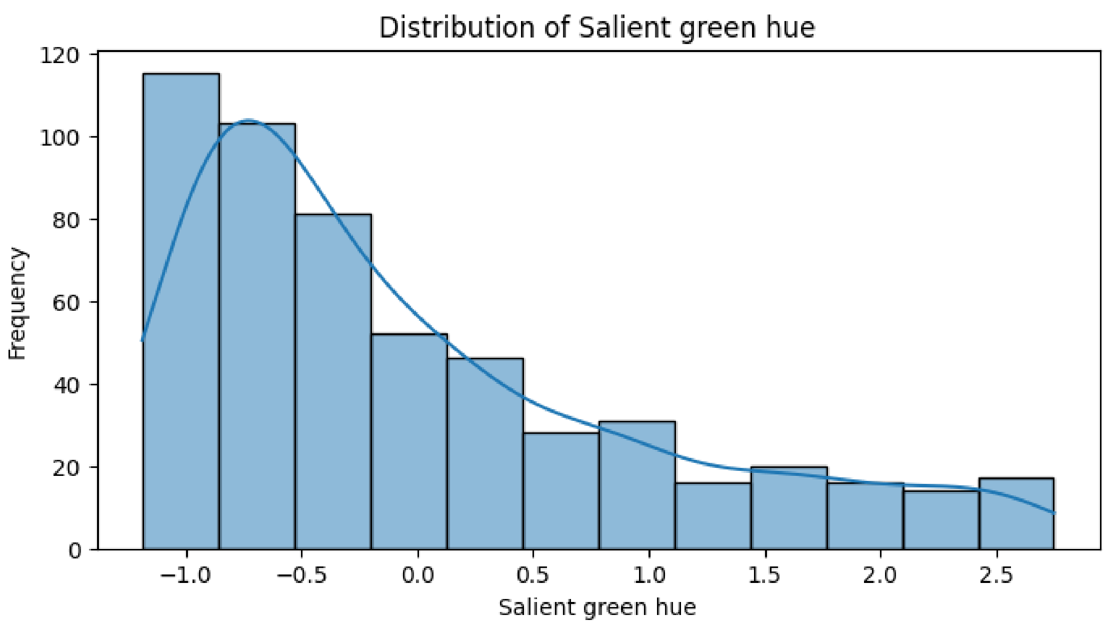
```
       Single translucent crystal  Multiple cubic crystals  Sandy texture  \
count                  539.000000               539.000000     539.000000   
mean                     0.000423                 0.000418       0.000216   
std                      1.000881                 1.000882       1.000916   
min                     -0.227922                -0.225045      -0.685937   
25%                     -0.227922                -0.225045      -0.685937   
50%                     -0.227922                -0.225045      -0.401124   
75%                     -0.227922                -0.225045       0.168500   
max                      7.120010                 7.986072       5.010309   

       Fragments (disjunctive)  Stripes (disjunctive)  Crystals (disjunctive)  
count               539.000000             539.000000              539.000000  
mean                 -0.001180               0.000759                0.000576  
std                   1.000553               1.000773                1.000839  
min                  -0.541391              -0.409247               -0.310419  
25%                  -0.541391              -0.409247               -0.310419  
50%                  -0.541391              -0.409247               -0.310419  
75%                   0.001226              -0.207298               -0.310419  
max                   3.137369               3.629722                5.216791  
```

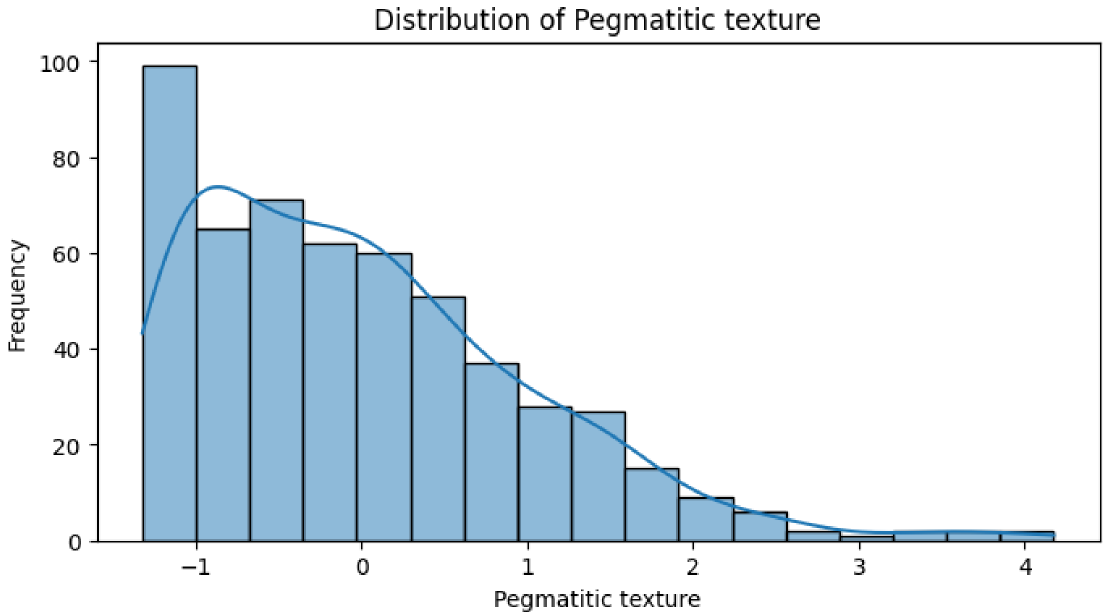
Visualizations

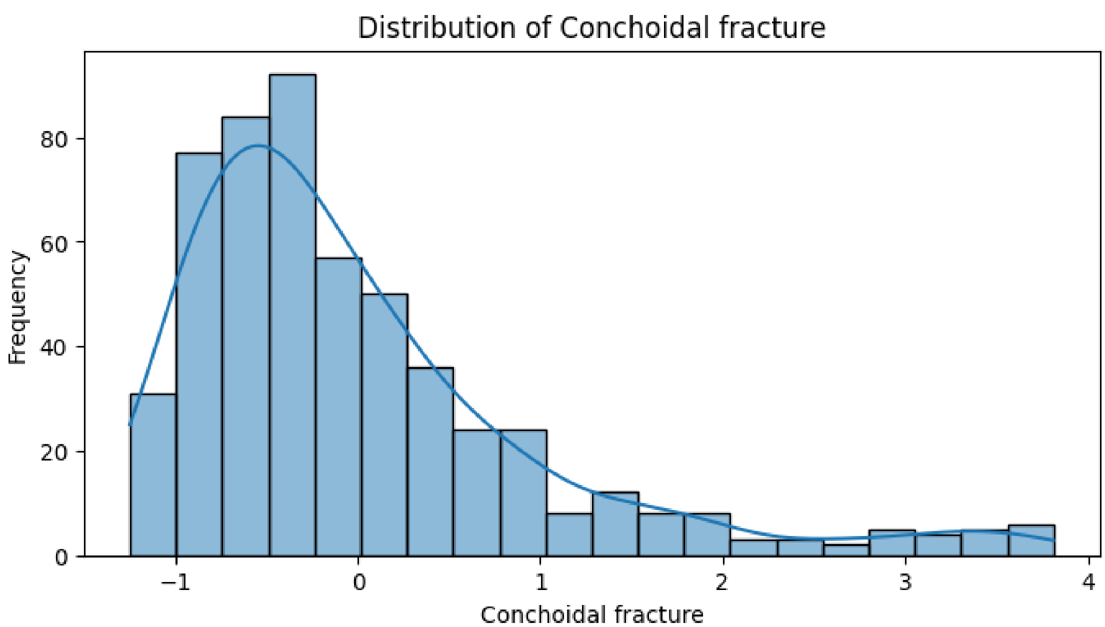
[ ]:
```
# Plot histograms for each attribute
for column in features.columns:
    plt.figure(figsize=(8, 4))
    sns.histplot(features[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel("Frequency")
    plt.show()
```

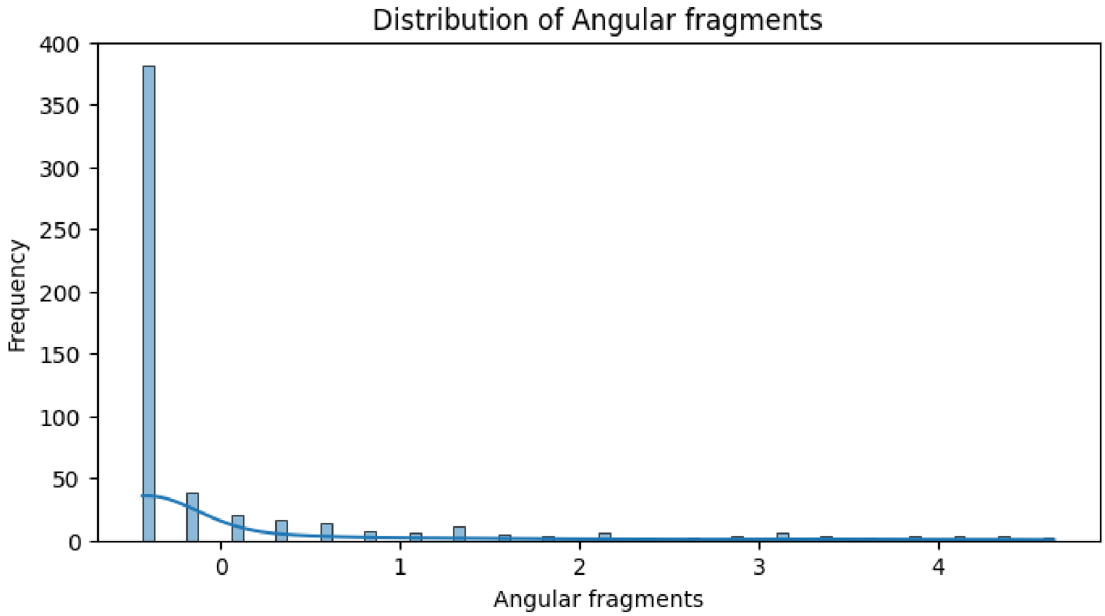


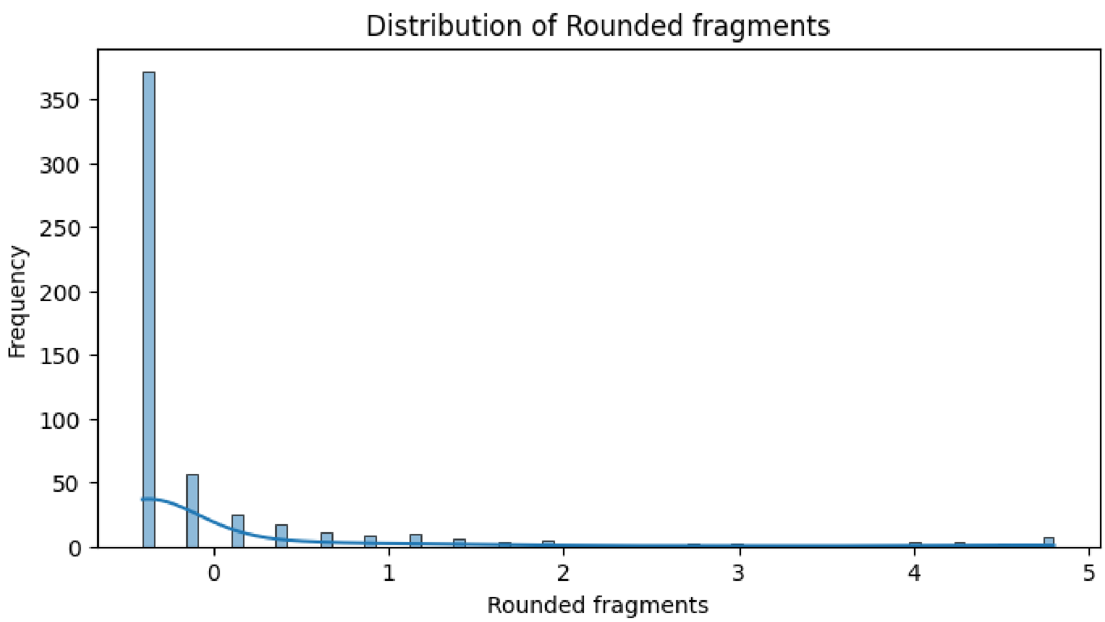

Distribution of Presence of holes
Frequency
300
250
200
150
100
50
0
0
1
2
3
4
Presence of holes

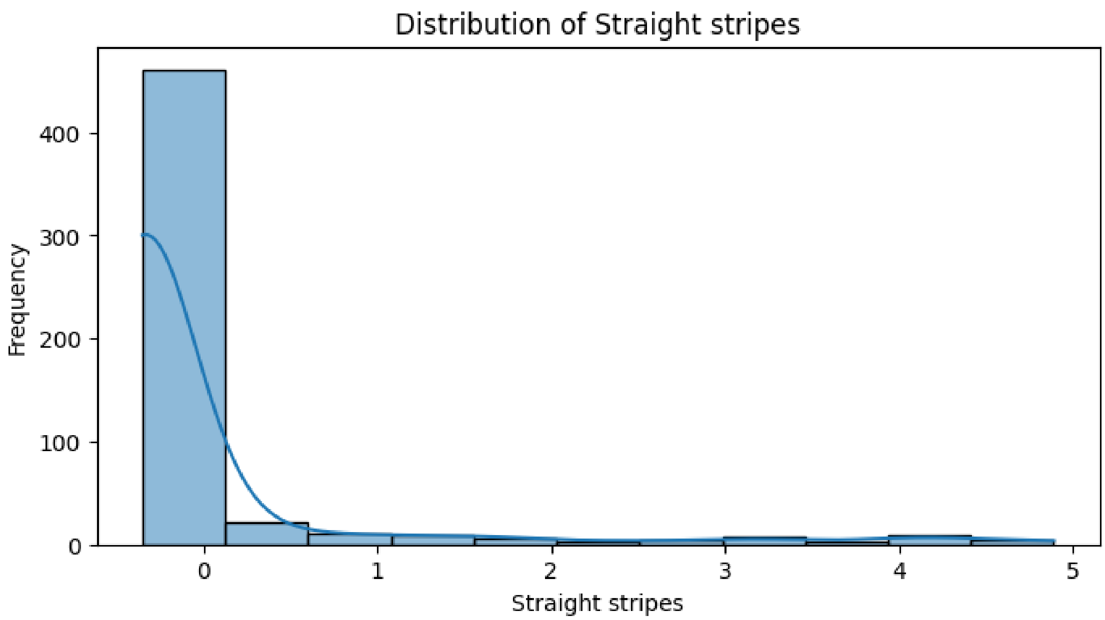

Distribution of Salient green hue
Frequency
120
100
80
60
40
20
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
Salient green hue

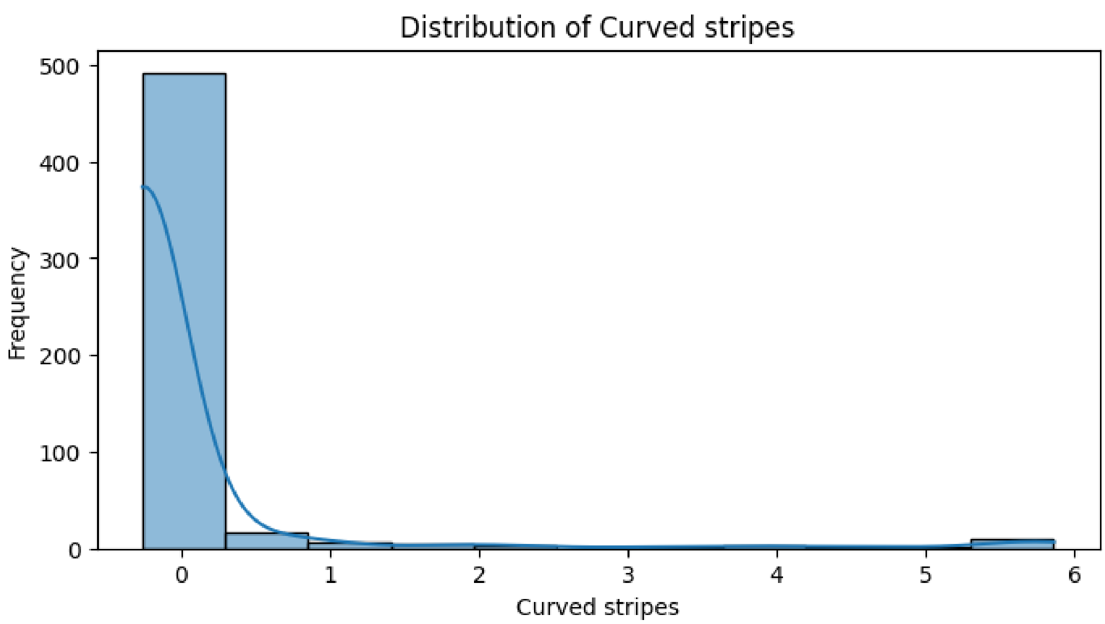

Distribution of Pegmatitic texture
Frequency
Pegmatitic texture
100
80
60
40
20
0
−1
0
1
2
3
4

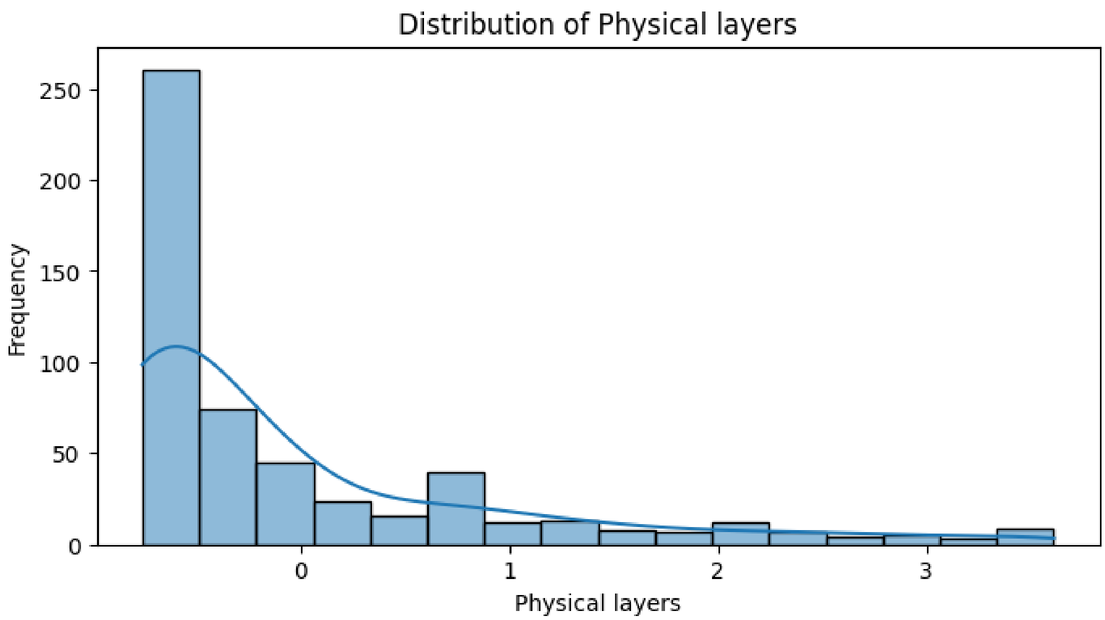

Distribution of Conchoidal fracture
Frequency
Conchoidal fracture
80
60
40
20
0
−1
0
1
2
3
4

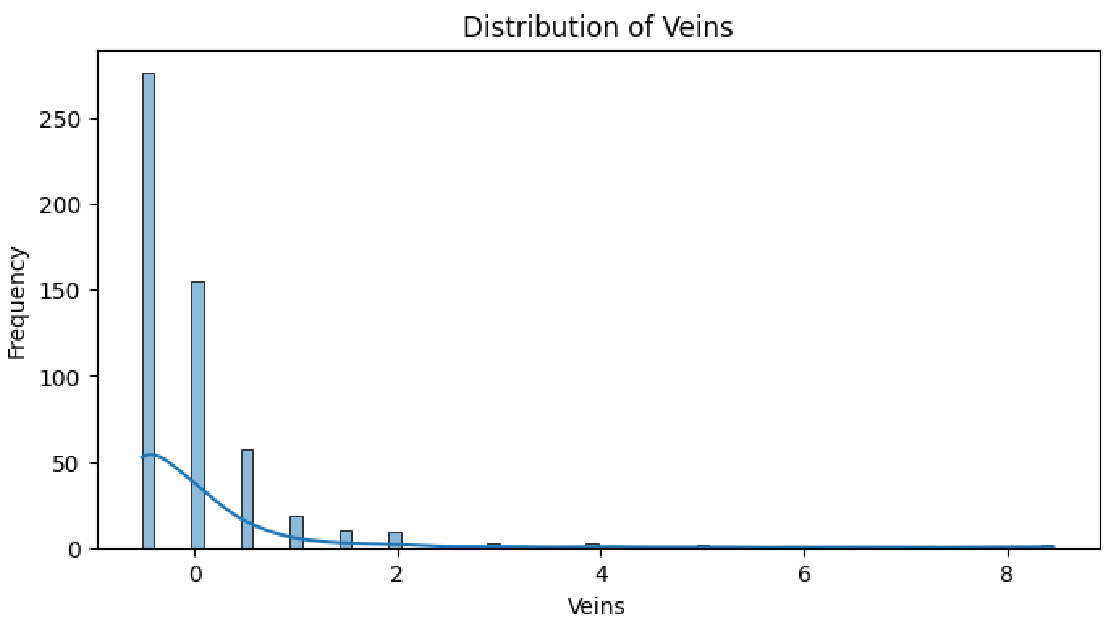

Distribution of Angular fragments
400
350
300
250
200
150
100
50
0
Frequency
0
1
2
3
4
Angular fragments

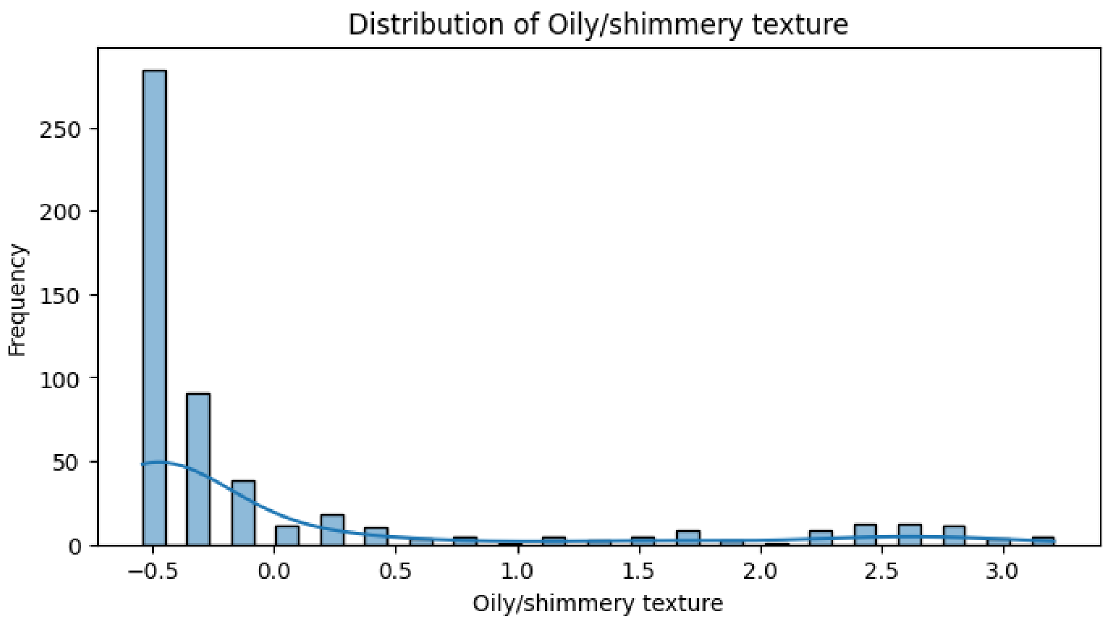

Distribution of Rounded fragments
350
300
250
200
150
100
50
0
Frequency
0
1
2
3
4
5
Rounded fragments


Distribution of Straight stripes
Frequency
400
300
200
100
0
0
1
2
3
4
5
Straight stripes

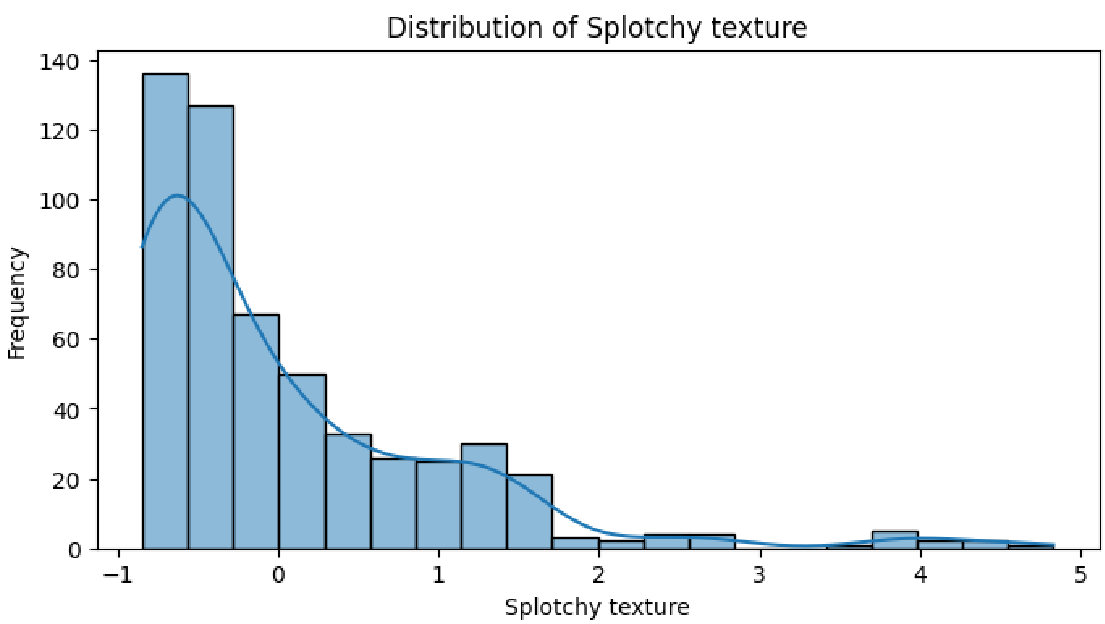

Distribution of Curved stripes
Frequency
500
400
300
200
100
0
0
1
2
3
4
5
6
Curved stripes


Distribution of Physical layers
Frequency
Physical layers
250
200
150
100
50
0
0
1
2
3

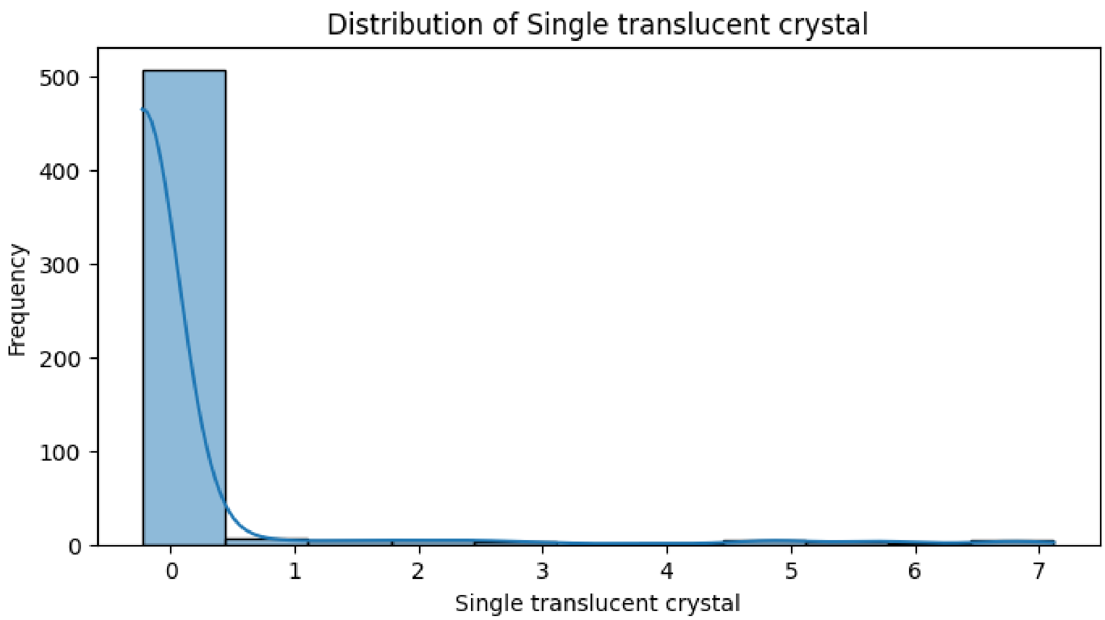

Distribution of Veins
Frequency
Veins
250
200
150
100
50
0
0
2
4
6
8

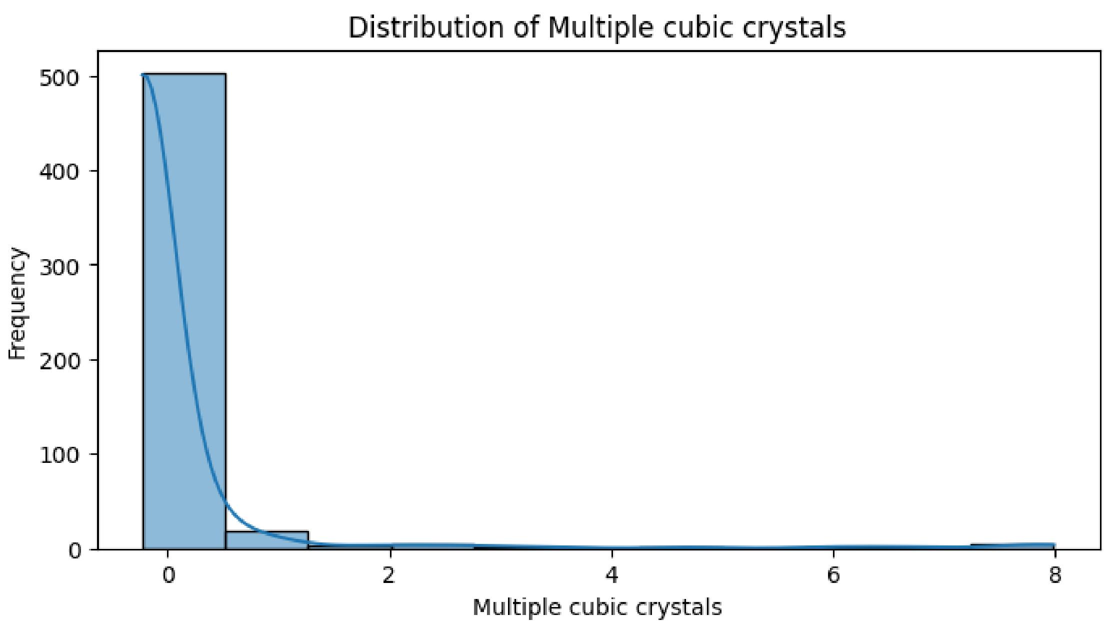


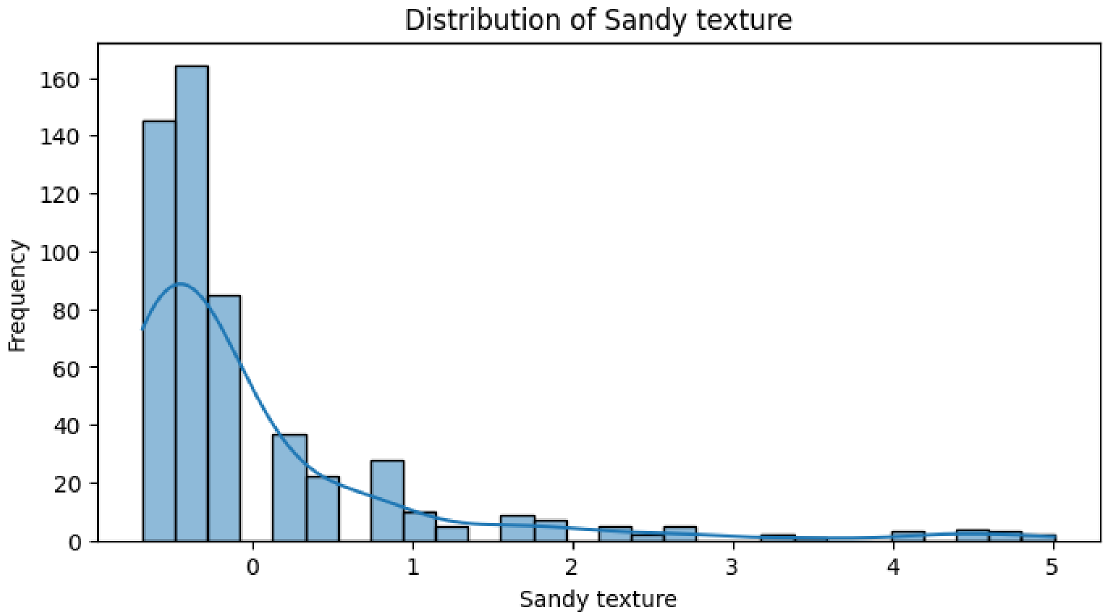
Distribution of Oily/shimmery texture
Frequency
0
50
100
150
200
250
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Oily/shimmery texture

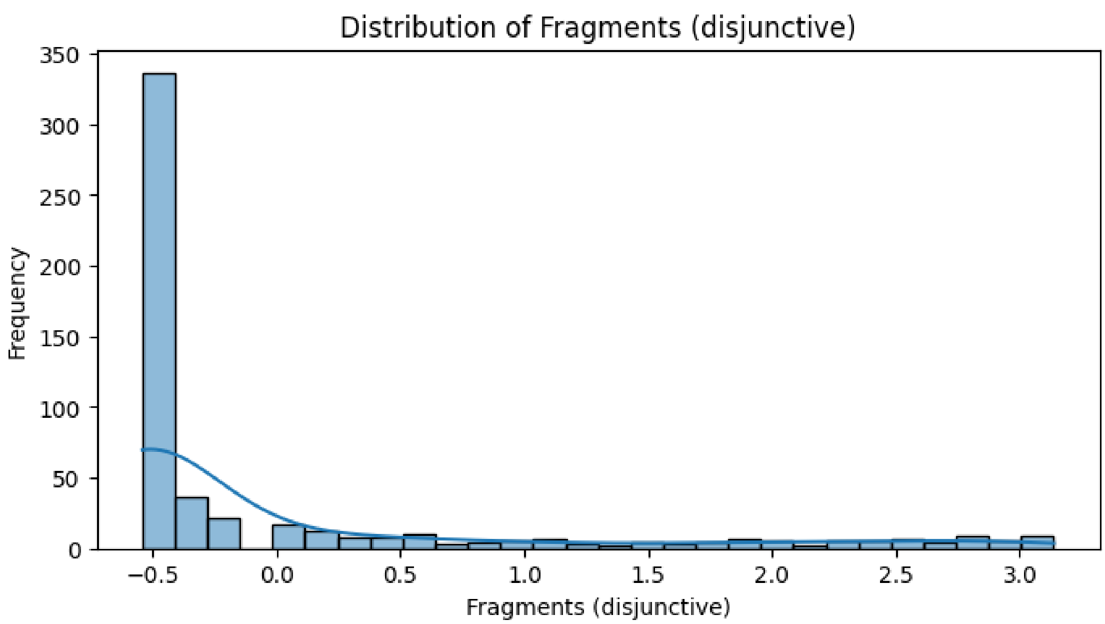


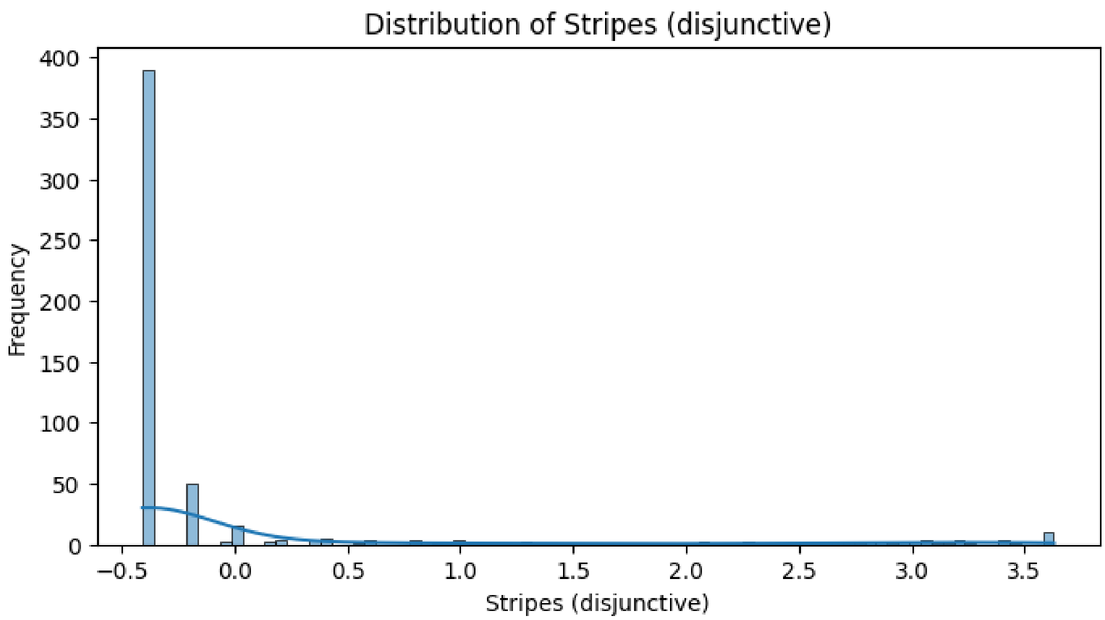
Distribution of Splotchy texture
Frequency
0
20
40
60
80
100
120
140
−1
0
1
2
3
4
5
Splotchy texture

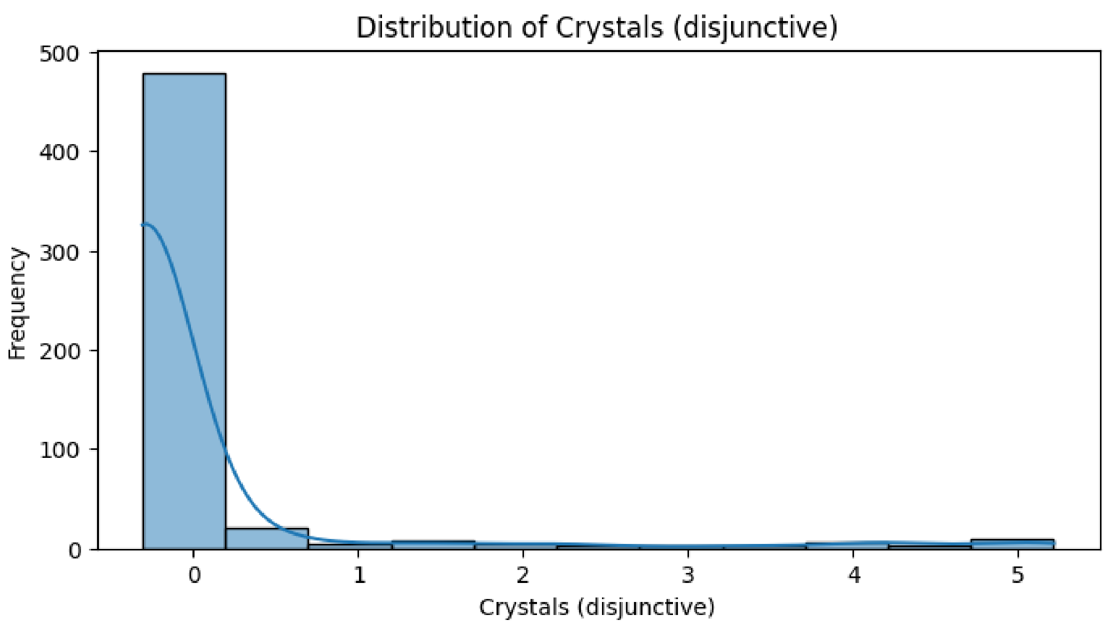

Distribution of Single translucent crystal
Frequency
500
400
300
200
100
0
0
1
2
3
4
5
6
7
Single translucent crystal


Distribution of Multiple cubic crystals
Frequency
500
400
300
200
100
0
0
2
4
6
8
Multiple cubic crystals


Distribution of Sandy texture
Frequency
0
20
40
60
80
100
120
140
160
0
1
2
3
4
5
Sandy texture


Distribution of Fragments (disjunctive)
Frequency
0
50
100
150
200
250
300
350
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Fragments (disjunctive)





1 If any special treatment required

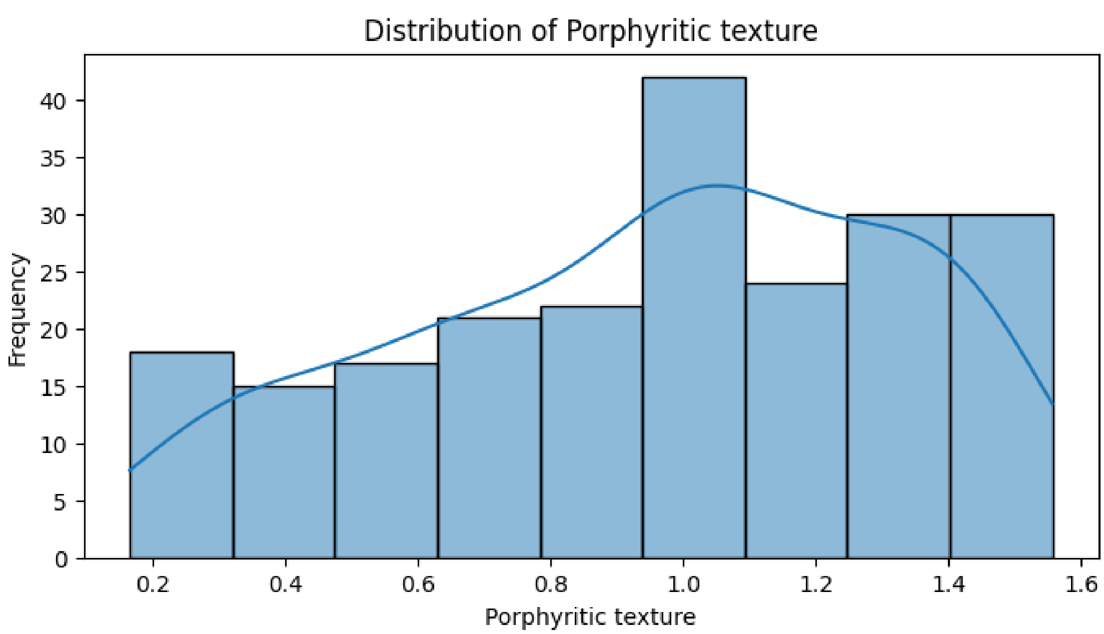
Based on our analysis of the histograms, it becomes apparent that all the attributes exhibit a right-skewed distribution. Notably, certain attributes such as angular fragments, rounded fragments, and veins demonstrate analogous distribution patterns. Conversely, attributes like crystals, stripes, multiple cubic crystals, and single translucent crystals exhibit a marked preponderance of lower values and an absence of higher values in their frequency distribution.

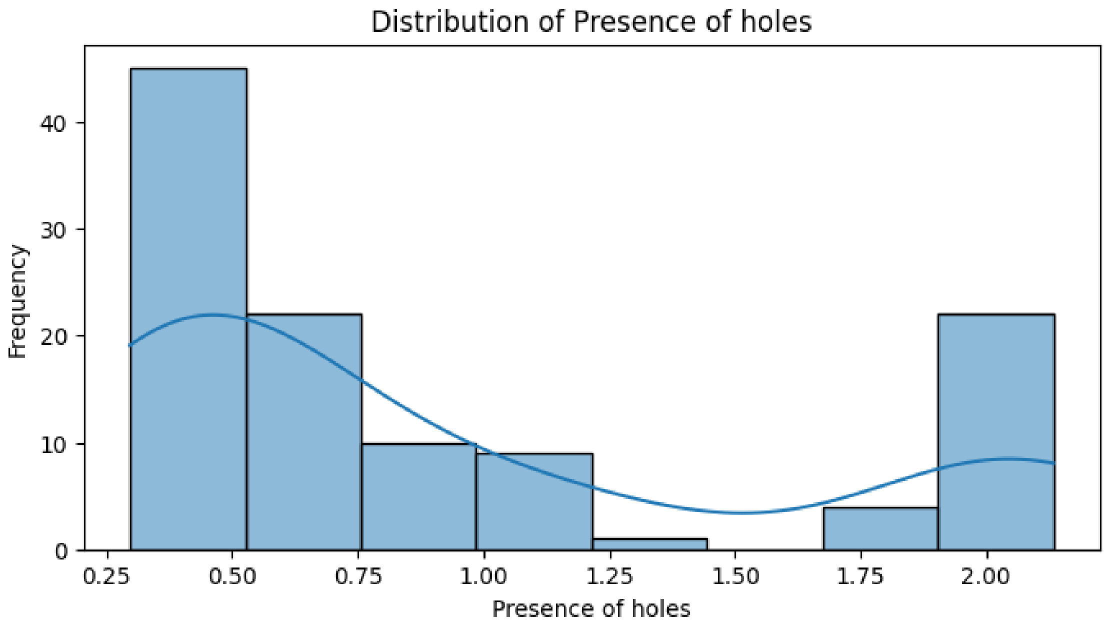
To rectify this skewness in our data, we applied a square root transformation to the attributes. This transformation was chosen with the aim of achieving symmetry in the attribute distributions, given their pronounced right-skewed nature.

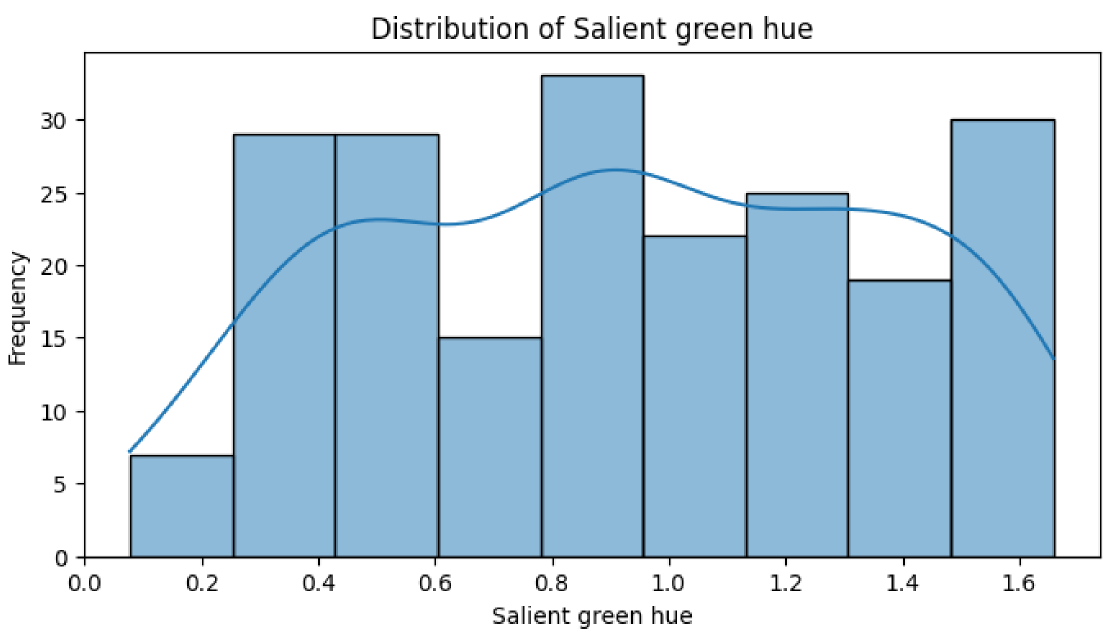
[ ]:
```python
import numpy as np

# Create a list of column names (attributes) in your dataset
attribute_columns = features.columns

# Apply a square root transformation to each attribute
for attr in attribute_columns:
    # Check if the attribute is numeric (e.g., not a label)
    if np.issubdtype(features[attr].dtype, np.number):
        features[attr] = np.sqrt(features[attr])
```

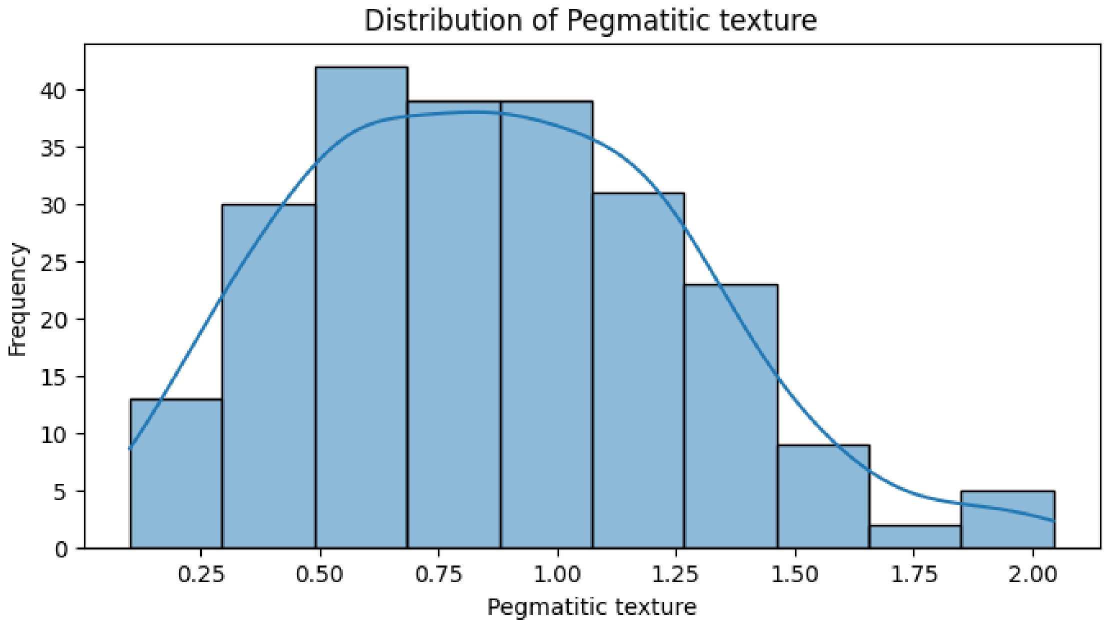
```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:402:
RuntimeWarning: invalid value encountered in sqrt
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

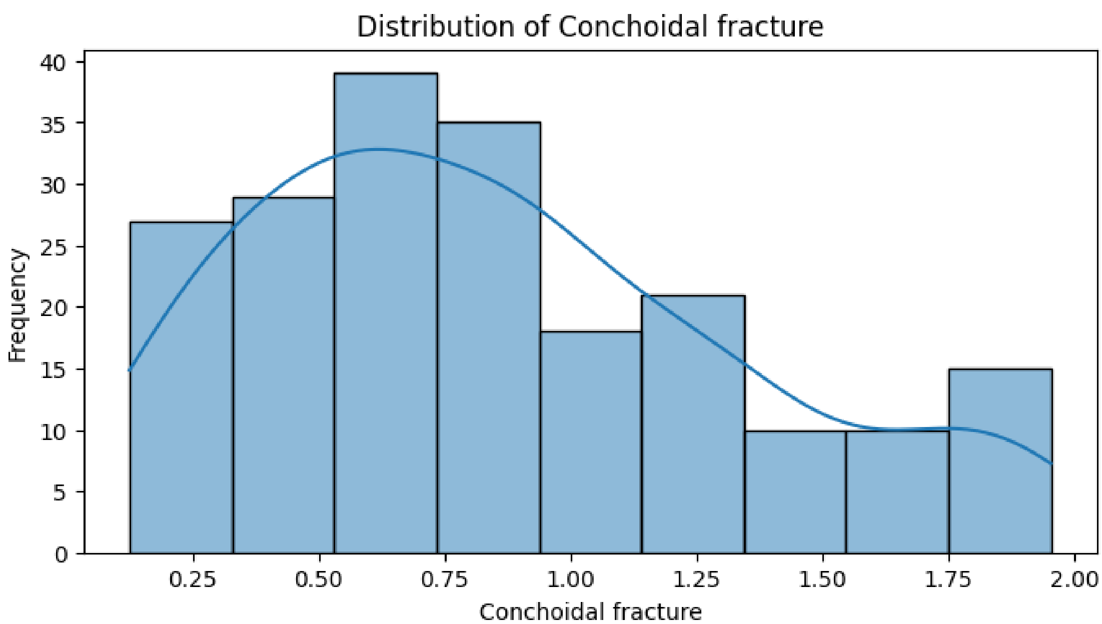
[ ]:
```python
# Plot histograms for each attribute
for column in features.columns:
    plt.figure(figsize=(8, 4))
    sns.histplot(features[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel("Frequency")
    plt.show()
```

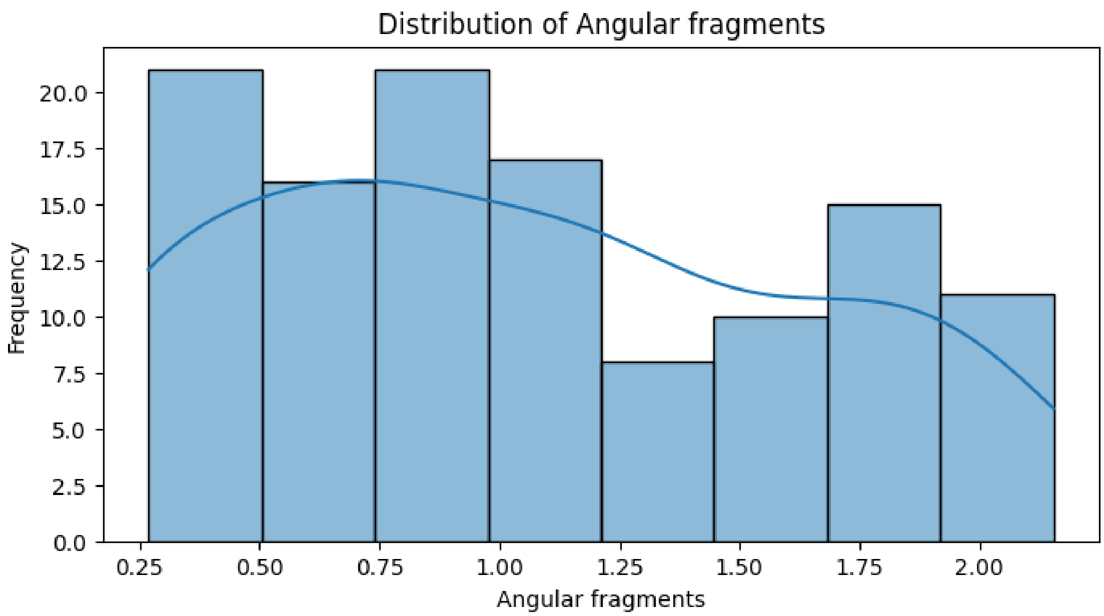


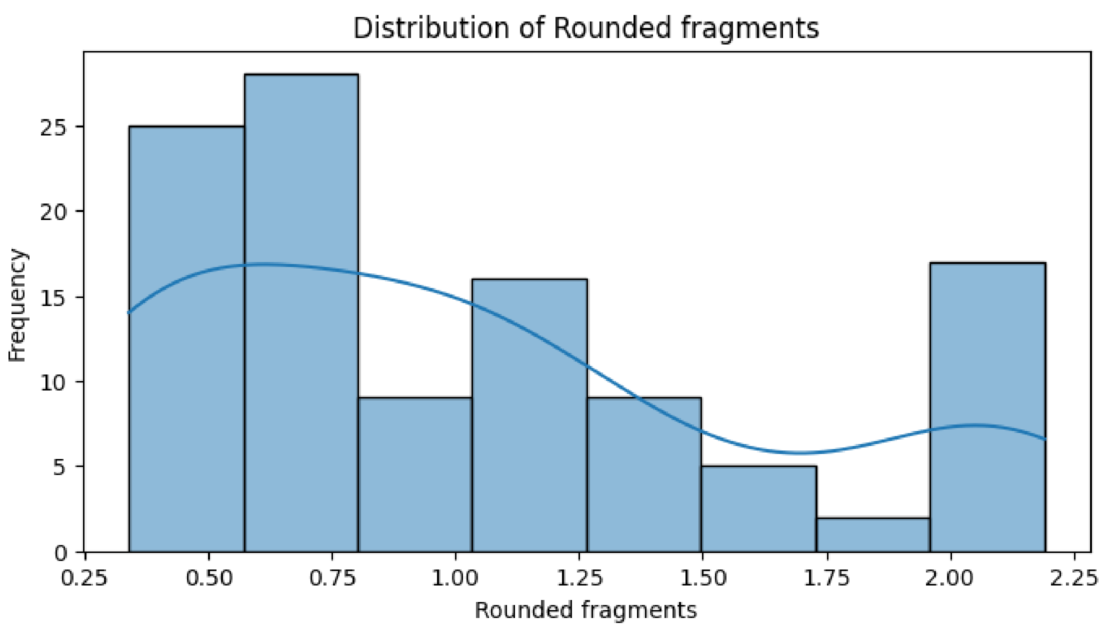

Distribution of Presence of holes
Frequency
0
10
20
30
40
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Presence of holes

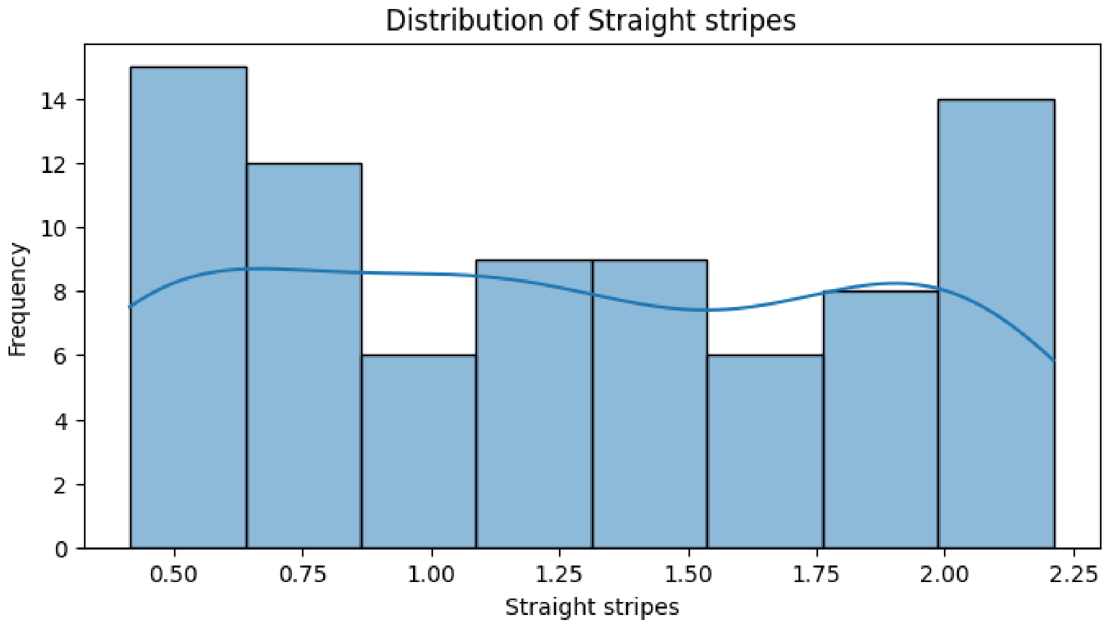

Distribution of Salient green hue
Frequency
0
5
10
15
20
25
30
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Salient green hue

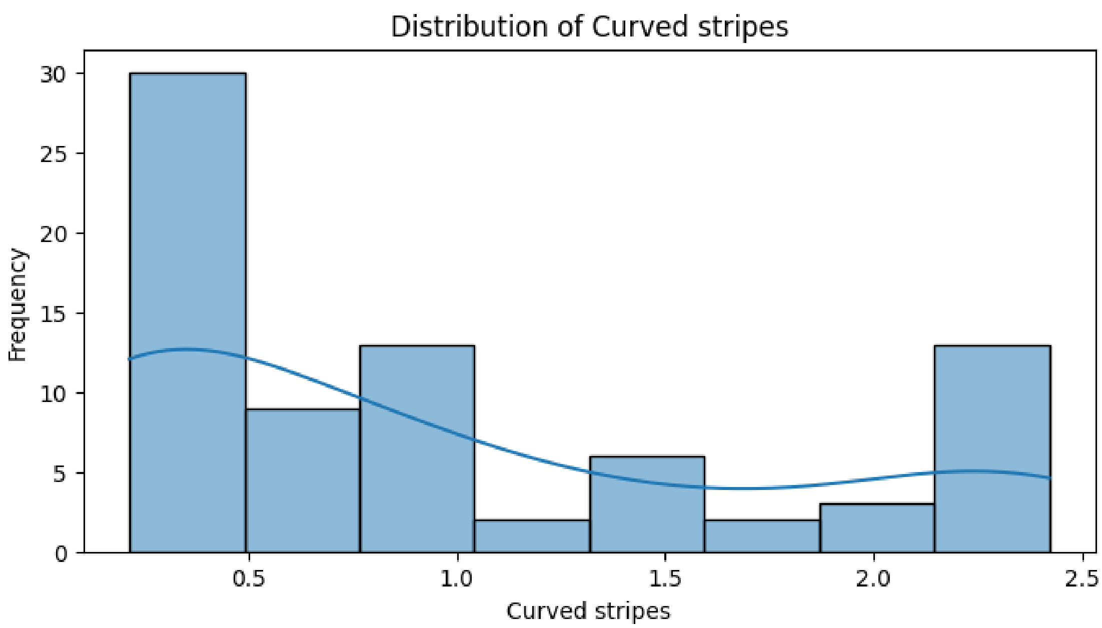

Distribution of Pegmatitic texture
Frequency
Pegmatitic texture
0
5
10
15
20
25
30
35
40
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

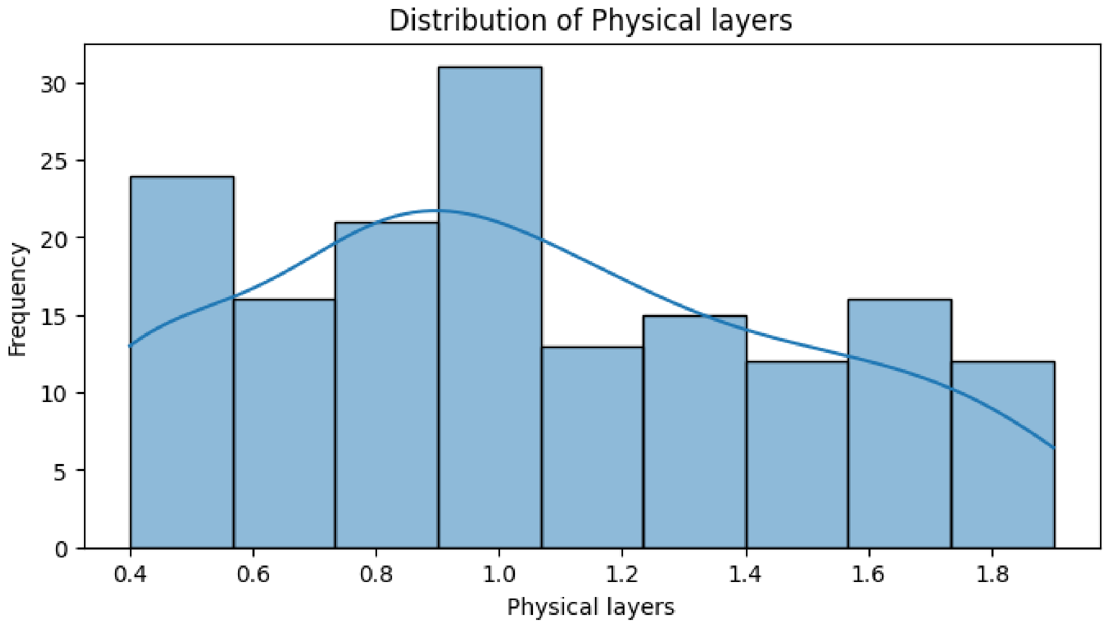

Distribution of Conchoidal fracture
Frequency
Conchoidal fracture
0
5
10
15
20
25
30
35
40
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

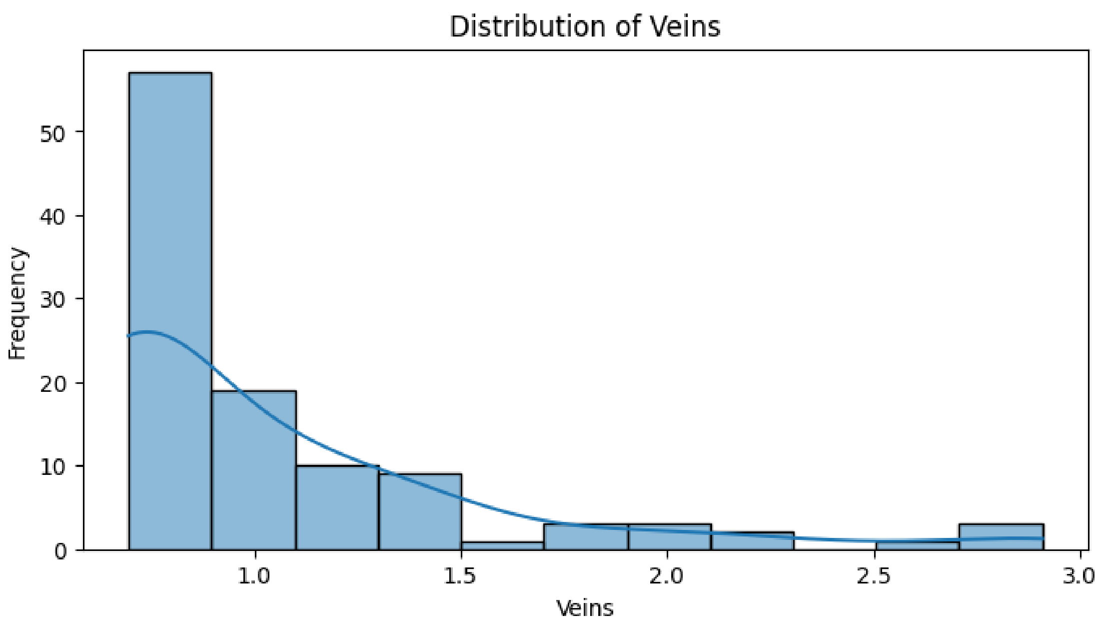

Distribution of Angular fragments
Frequency
Angular fragments
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

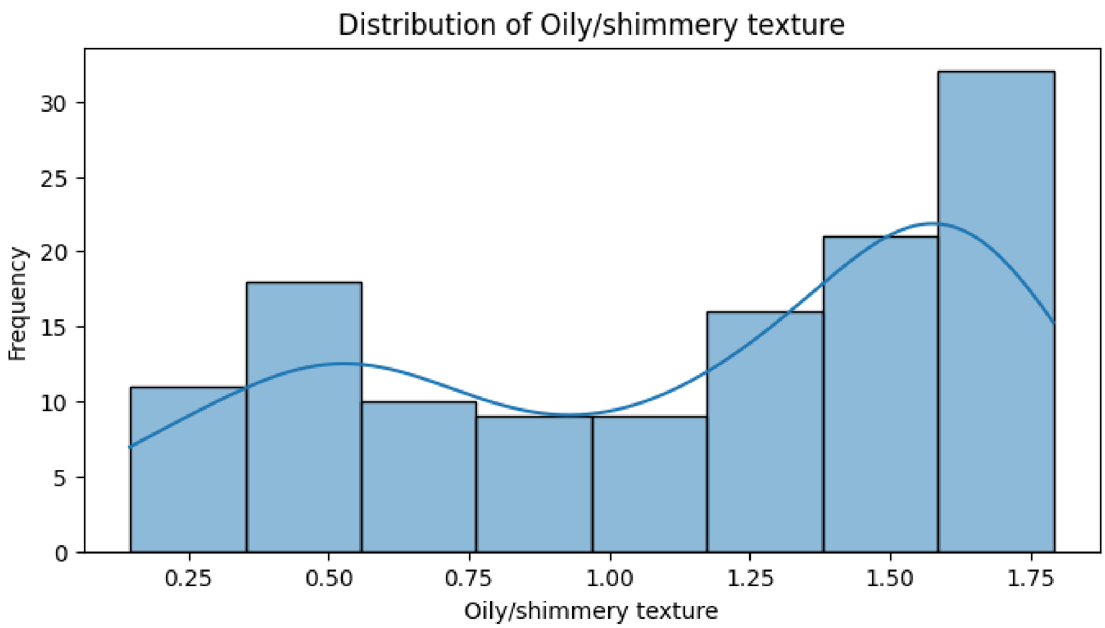

Distribution of Rounded fragments
Frequency
Rounded fragments
0
5
10
15
20
25
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25

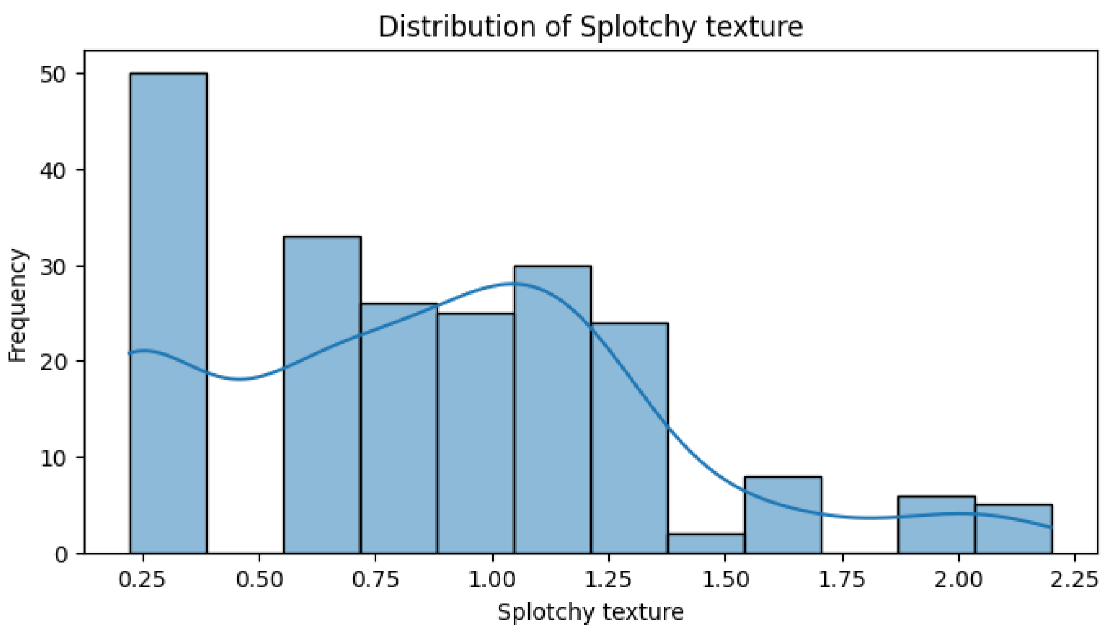

Distribution of Straight stripes
Frequency
0
2
4
6
8
10
12
14
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
Straight stripes

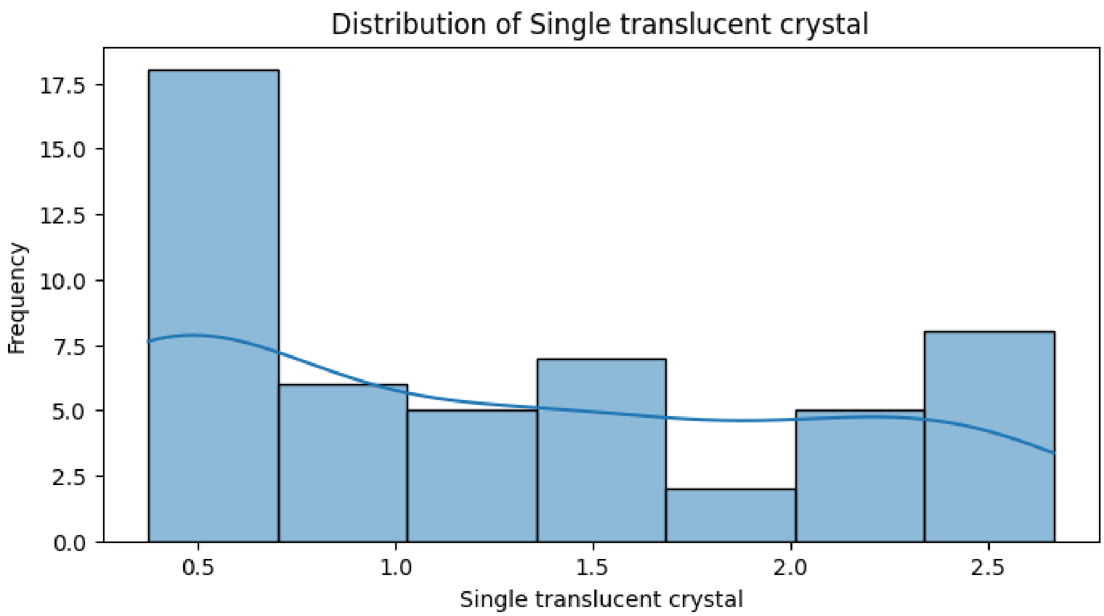

Distribution of Curved stripes
Frequency
0
5
10
15
20
25
30
0.5
1.0
1.5
2.0
2.5
Curved stripes

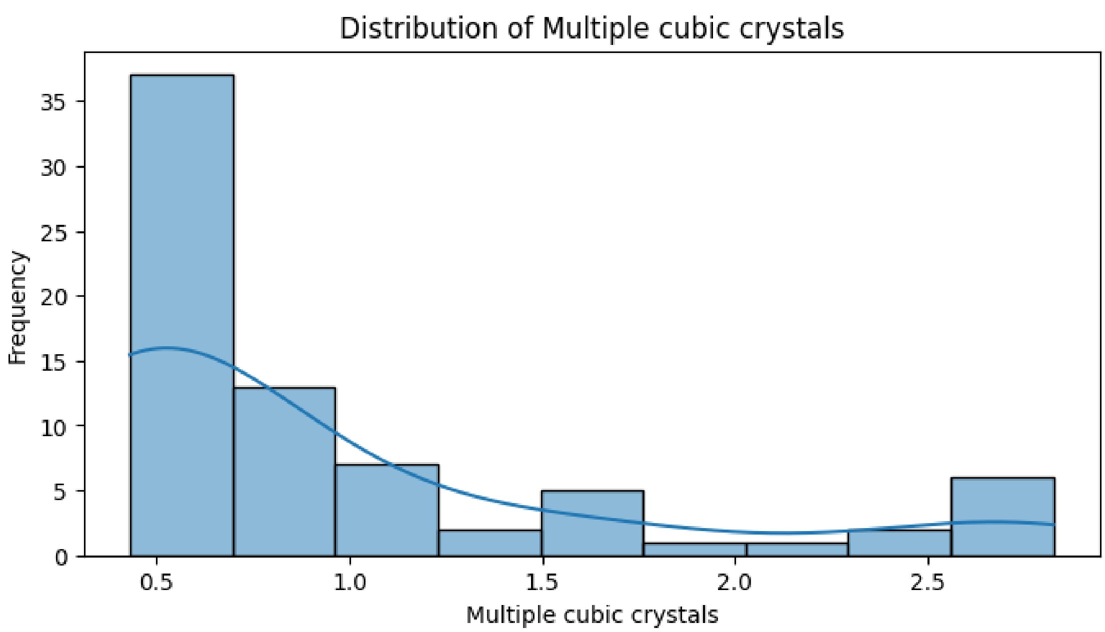

Distribution of Physical layers
Frequency
0
5
10
15
20
25
30
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Physical layers

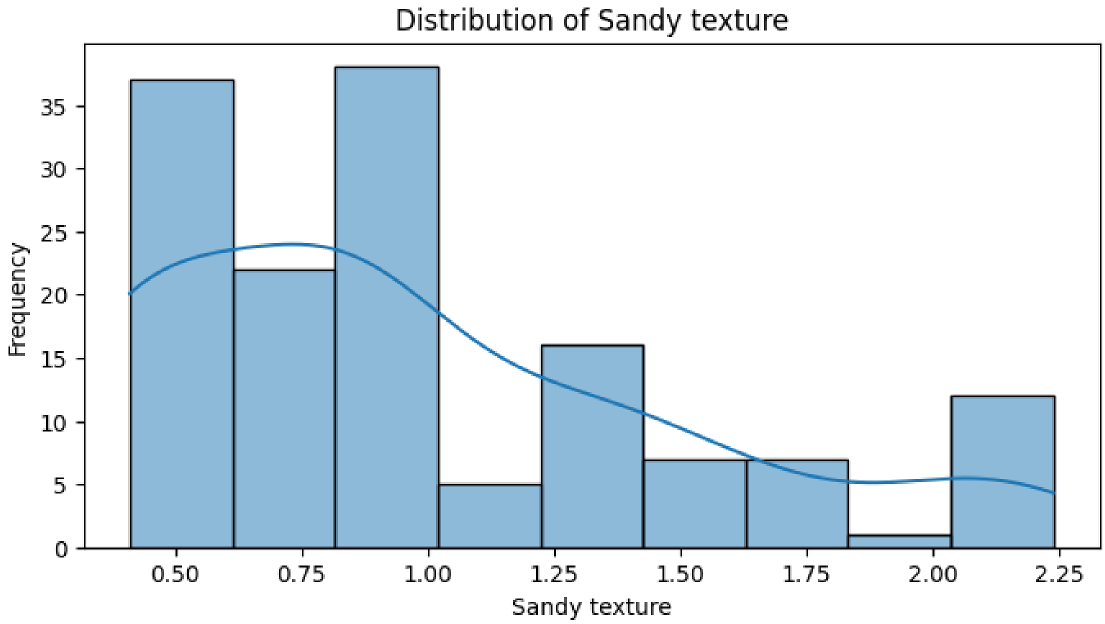

Distribution of Veins
Frequency
0
10
20
30
40
50
1.0
1.5
2.0
2.5
3.0
Veins

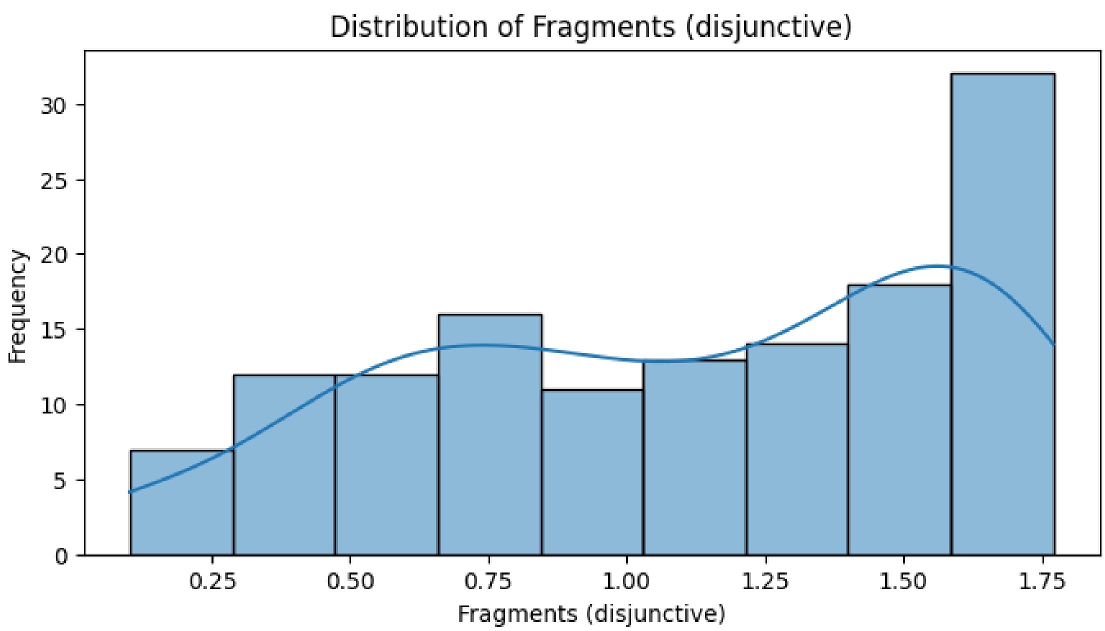

Distribution of Oily/shimmery texture
Frequency
0
5
10
15
20
25
30
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Oily/shimmery texture

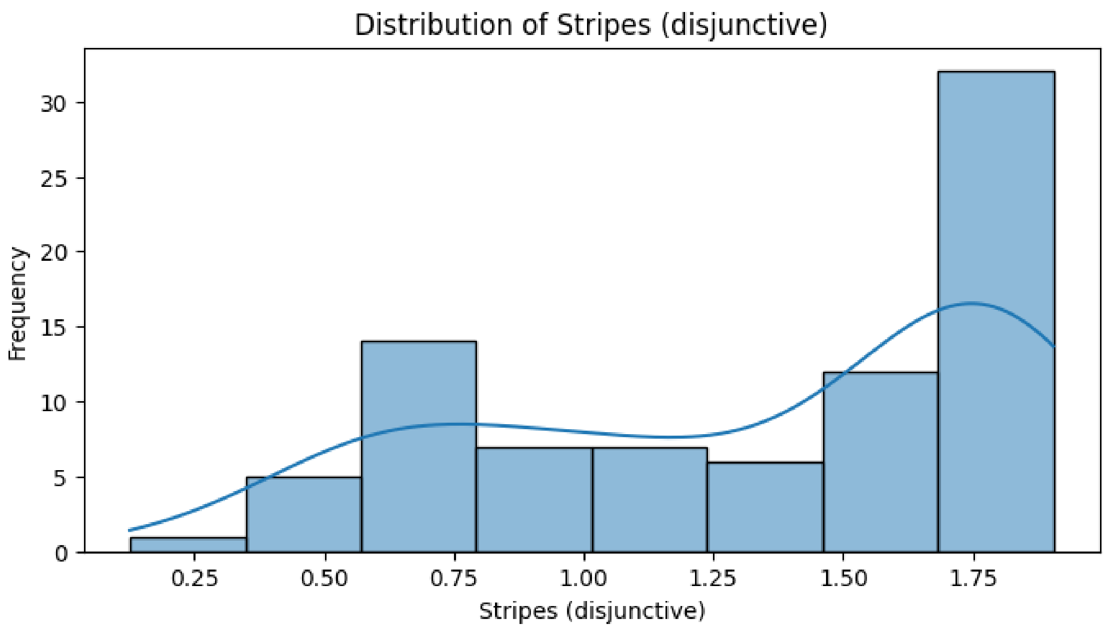

Distribution of Splotchy texture
Frequency
0
10
20
30
40
50
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
Splotchy texture

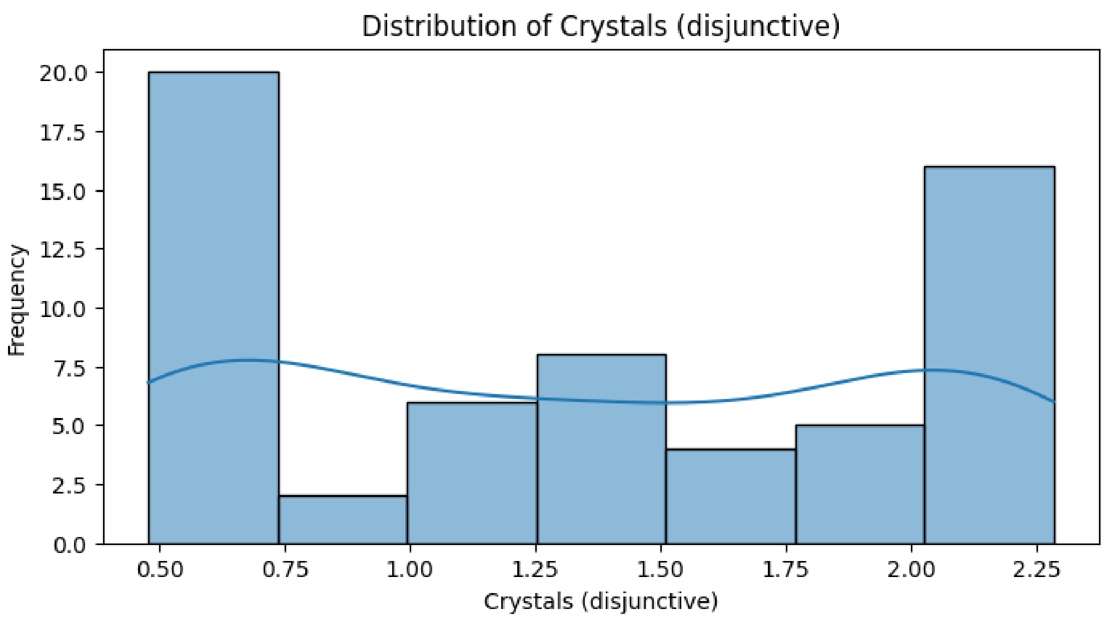

Distribution of Single translucent crystal
Frequency
Single translucent crystal
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.5
1.0
1.5
2.0
2.5


Distribution of Multiple cubic crystals
Frequency
Multiple cubic crystals
0
5
10
15
20
25
30
35
0.5
1.0
1.5
2.0
2.5


Distribution of Sandy texture
Frequency
Sandy texture
0
5
10
15
20
25
30
35
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25


Distribution of Fragments (disjunctive)
Frequency
Fragments (disjunctive)
0
5
10
15
20
25
30
0.25
0.50
0.75
1.00
1.25
1.50
1.75





Form the above histograms, we can see the skewness is removed.

[ ]:
```
#looking for if there are any null values in the dataset
features.isnull().sum()
```

```
[ ]: Porphyritic texture          320
     Presence of holes            426
     Salient green hue            330
     Pegmatitic texture           306
     Conchoidal fracture          335
     Angular fragments            420
     Rounded fragments            428
     Straight stripes             460
     Curved stripes               461
     Physical layers              379
     Veins                        431
     Oily/shimmery texture        413
     Splotchy texture             330
     Single translucent crystal   488
     Multiple cubic crystals      465
     Sandy texture                394
     Fragments (disjunctive)      404
     Stripes (disjunctive)        455
     Crystals (disjunctive)       478
     dtype: int64
```

2. Analyze and discuss the relationships between the data attributes, and between the data attributes and label. This involves computing the Pearson Correlation Coefficient (PCC) and generating scatter plots.

Computing the PCC & Scatter Plots

```
[ ]: correlation_matrix = features.corr()
     print(correlation_matrix)
     # Generate scatter plots for key attribute pairs
     sns.pairplot(features)
     plt.show()
```

```
                            Porphyritic texture  Presence of holes  \
Porphyritic texture                    1.000000          -0.226835
Presence of holes                     -0.226835           1.000000
Salient green hue                      0.220720          -0.016403
Pegmatitic texture                     0.123277          -0.101623
Conchoidal fracture                    0.030752          -0.459924
Angular fragments                      0.479444          -0.048545
Rounded fragments                      0.509510          -0.255673
Straight stripes                      -0.089885           0.713204
Curved stripes                        -0.432069          -0.882783
Physical layers                       -0.854029          -0.364786
Veins                                 -0.091887                NaN
Oily/shimmery texture                 -0.162138          -0.342422
Splotchy texture                      -0.221707          -0.183036
Single translucent crystal             0.078274          -0.815874
```

Multiple cubic crystals	-0.386227	0.255062
Sandy texture	-0.314268	-0.263749
Fragments (disjunctive)	0.704341	-0.115251
Stripes (disjunctive)	-0.287052	0.450846
Crystals (disjunctive)	-0.520408	-0.150203

	Salient green hue	Pegmatitic texture \
Porphyritic texture	0.220720	0.123277
Presence of holes	-0.016403	-0.101623
Salient green hue	1.000000	0.129477
Pegmatitic texture	0.129477	1.000000
Conchoidal fracture	0.036471	-0.103815
Angular fragments	0.061481	0.024943
Rounded fragments	-0.000469	0.144445
Straight stripes	-0.193162	-0.388210
Curved stripes	-0.041192	-0.369645
Physical layers	-0.227131	0.039625
Veins	0.063678	-0.111310
Oily/shimmery texture	-0.260429	-0.390194
Splotchy texture	-0.101093	-0.262243
Single translucent crystal	-0.514440	-0.372889
Multiple cubic crystals	-0.034712	0.202968
Sandy texture	-0.186429	-0.268540
Fragments (disjunctive)	0.051826	0.092301
Stripes (disjunctive)	-0.200757	-0.473403
Crystals (disjunctive)	-0.585575	-0.066517

	Conchoidal fracture	Angular fragments \
Porphyritic texture	0.030752	0.479444
Presence of holes	-0.459924	-0.048545
Salient green hue	0.036471	0.061481
Pegmatitic texture	-0.103815	0.024943
Conchoidal fracture	1.000000	0.100408
Angular fragments	0.100408	1.000000
Rounded fragments	-0.337588	-0.155240
Straight stripes	-0.352130	-0.273148
Curved stripes	-0.269730	0.234508
Physical layers	-0.252711	-0.323212
Veins	0.006561	-0.649544
Oily/shimmery texture	0.417987	-0.449567
Splotchy texture	0.008009	-0.384342
Single translucent crystal	-0.187257	0.957867
Multiple cubic crystals	-0.276077	-0.197830
Sandy texture	-0.226216	-0.322473
Fragments (disjunctive)	-0.301335	0.681826
Stripes (disjunctive)	-0.375378	-0.975556
Crystals (disjunctive)	-0.247676	-0.093359

	Rounded fragments	Straight stripes \
Porphyritic texture	0.509510	-0.089885
Presence of holes	-0.255673	0.713204
Salient green hue	-0.000469	-0.193162
Pegmatitic texture	0.144445	-0.388210
Conchoidal fracture	-0.337588	-0.352130
Angular fragments	-0.155240	-0.273148
Rounded fragments	1.000000	NaN
Straight stripes	NaN	1.000000
Curved stripes	0.183589	-0.340561
Physical layers	0.173212	-0.217668
Veins	0.650269	0.053307
Oily/shimmery texture	0.220924	-0.142643
Splotchy texture	-0.512701	0.060216
Single translucent crystal	-1.000000	-0.569614
Multiple cubic crystals	-0.237033	-0.549232
Sandy texture	-0.365800	-0.113447
Fragments (disjunctive)	0.650255	0.787314
Stripes (disjunctive)	NaN	0.774447
Crystals (disjunctive)	0.243661	0.260333

	Curved stripes	Physical layers	Veins \
Porphyritic texture	-0.432069	-0.854029	-0.091887
Presence of holes	-0.882783	-0.364786	NaN
Salient green hue	-0.041192	-0.227131	0.063678
Pegmatitic texture	-0.369645	0.039625	-0.111310
Conchoidal fracture	-0.269730	-0.252711	0.006561
Angular fragments	0.234508	-0.323212	-0.649544
Rounded fragments	0.183589	0.173212	0.650269
Straight stripes	-0.340561	-0.217668	0.053307
Curved stripes	1.000000	-0.363441	-0.050845
Physical layers	-0.363441	1.000000	-0.209650
Veins	-0.050845	-0.209650	1.000000
Oily/shimmery texture	-0.218811	-0.236097	0.212375
Splotchy texture	-0.220838	-0.165116	-0.254802
Single translucent crystal	-0.387456	-0.382353	-0.028522
Multiple cubic crystals	NaN	-0.090518	0.728863
Sandy texture	-0.241869	-0.210196	-0.167760
Fragments (disjunctive)	0.031828	-0.173680	0.013013
Stripes (disjunctive)	0.497235	-0.316920	-0.116757
Crystals (disjunctive)	-1.000000	-0.162053	-0.051784

	Oily/shimmery texture	Splotchy texture \
Porphyritic texture	-0.162138	-0.221707
Presence of holes	-0.342422	-0.183036
Salient green hue	-0.260429	-0.101093
Pegmatitic texture	-0.390194	-0.262243
Conchoidal fracture	0.417987	0.008009

Angular fragments	-0.449567	-0.384342
Rounded fragments	0.220924	-0.512701
Straight stripes	-0.142643	0.060216
Curved stripes	-0.218811	-0.220838
Physical layers	-0.236097	-0.165116
Veins	0.212375	-0.254802
Oily/shimmery texture	1.000000	-0.039702
Splotchy texture	-0.039702	1.000000
Single translucent crystal	-0.504385	-0.109616
Multiple cubic crystals	-0.395654	-0.489678
Sandy texture	-0.484977	-0.189617
Fragments (disjunctive)	-0.247218	-0.425904
Stripes (disjunctive)	-0.220983	0.411933
Crystals (disjunctive)	-0.376955	0.003682

	Single translucent crystal \
Porphyritic texture	0.078274
Presence of holes	-0.815874
Salient green hue	-0.514440
Pegmatitic texture	-0.372889
Conchoidal fracture	-0.187257
Angular fragments	0.957867
Rounded fragments	-1.000000
Straight stripes	-0.569614
Curved stripes	-0.387456
Physical layers	-0.382353
Veins	-0.028522
Oily/shimmery texture	-0.504385
Splotchy texture	-0.109616
Single translucent crystal	1.000000
Multiple cubic crystals	-0.482874
Sandy texture	NaN
Fragments (disjunctive)	0.613654
Stripes (disjunctive)	-0.625776
Crystals (disjunctive)	0.563577

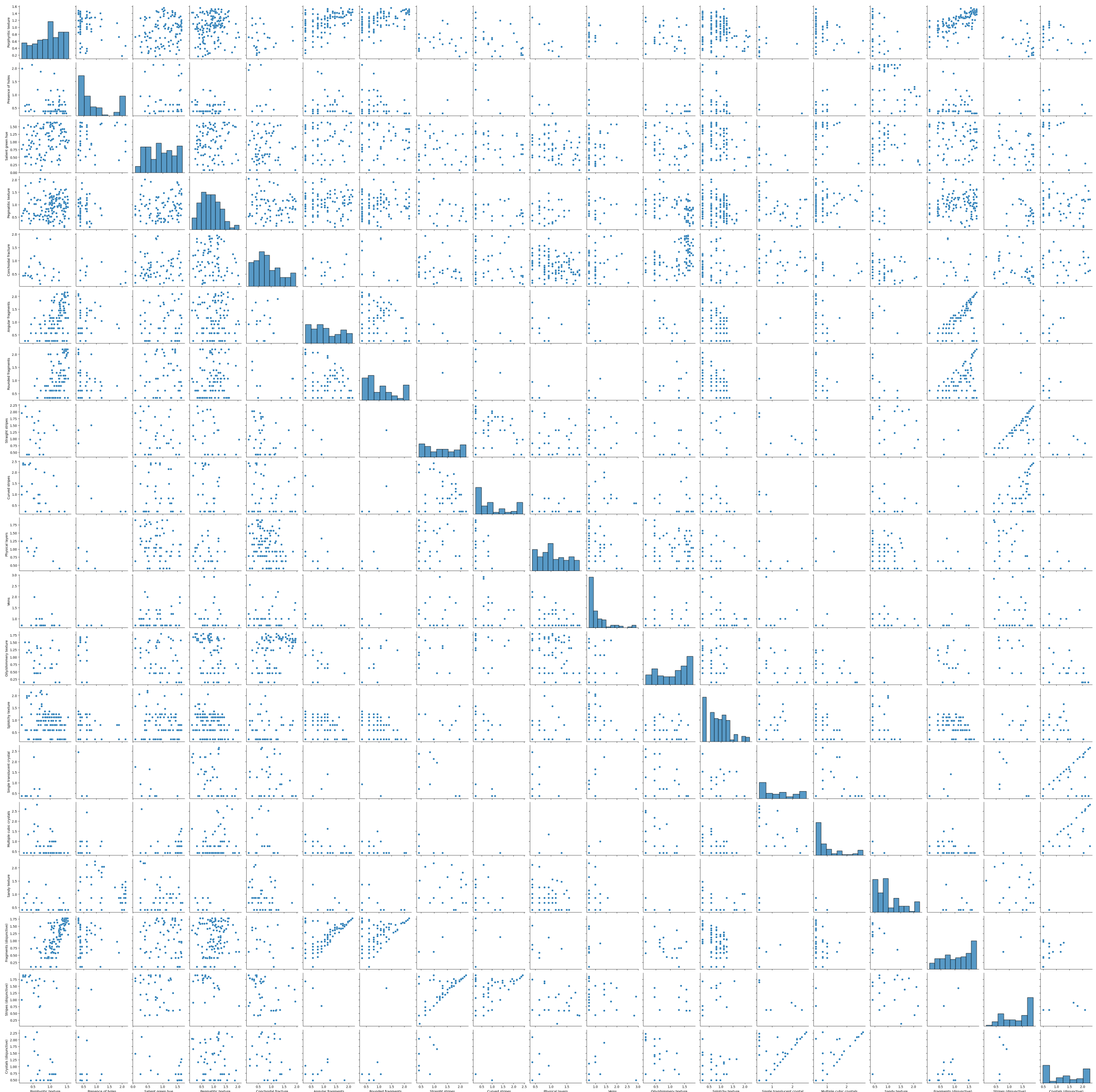
	Multiple cubic crystals	Sandy texture \
Porphyritic texture	-0.386227	-0.314268
Presence of holes	0.255062	-0.263749
Salient green hue	-0.034712	-0.186429
Pegmatitic texture	0.202968	-0.268540
Conchoidal fracture	-0.276077	-0.226216
Angular fragments	-0.197830	-0.322473
Rounded fragments	-0.237033	-0.365800
Straight stripes	-0.549232	-0.113447
Curved stripes	NaN	-0.241869
Physical layers	-0.090518	-0.210196
Veins	0.728863	-0.167760

Oily/shimmery texture	-0.395654	-0.484977
Splotchy texture	-0.489678	-0.189617
Single translucent crystal	-0.482874	NaN
Multiple cubic crystals	1.000000	-0.333333
Sandy texture	-0.333333	1.000000
Fragments (disjunctive)	-0.217992	-0.335500
Stripes (disjunctive)	NaN	-0.168958
Crystals (disjunctive)	0.771804	NaN

	Fragments (disjunctive)	Stripes (disjunctive) \
Porphyritic texture	0.704341	-0.287052
Presence of holes	-0.115251	0.450846
Salient green hue	0.051826	-0.200757
Pegmatitic texture	0.092301	-0.473403
Conchoidal fracture	-0.301335	-0.375378
Angular fragments	0.681826	-0.975556
Rounded fragments	0.650255	NaN
Straight stripes	0.787314	0.774447
Curved stripes	0.031828	0.497235
Physical layers	-0.173680	-0.316920
Veins	0.013013	-0.116757
Oily/shimmery texture	-0.247218	-0.220983
Splotchy texture	-0.425904	0.411933
Single translucent crystal	0.613654	-0.625776
Multiple cubic crystals	-0.217992	NaN
Sandy texture	-0.335500	-0.168958
Fragments (disjunctive)	1.000000	1.000000
Stripes (disjunctive)	1.000000	1.000000
Crystals (disjunctive)	0.230808	0.427913

	Crystals (disjunctive)
Porphyritic texture	-0.520408
Presence of holes	-0.150203
Salient green hue	-0.585575
Pegmatitic texture	-0.066517
Conchoidal fracture	-0.247676
Angular fragments	-0.093359
Rounded fragments	0.243661
Straight stripes	0.260333
Curved stripes	-1.000000
Physical layers	-0.162053
Veins	-0.051784
Oily/shimmery texture	-0.376955
Splotchy texture	0.003682
Single translucent crystal	0.563577
Multiple cubic crystals	0.771804
Sandy texture	NaN
Fragments (disjunctive)	0.230808

```
Stripes (disjunctive)                       0.427913
Crystals (disjunctive)                      1.000000
```



PCC between label (rock category) and attributes

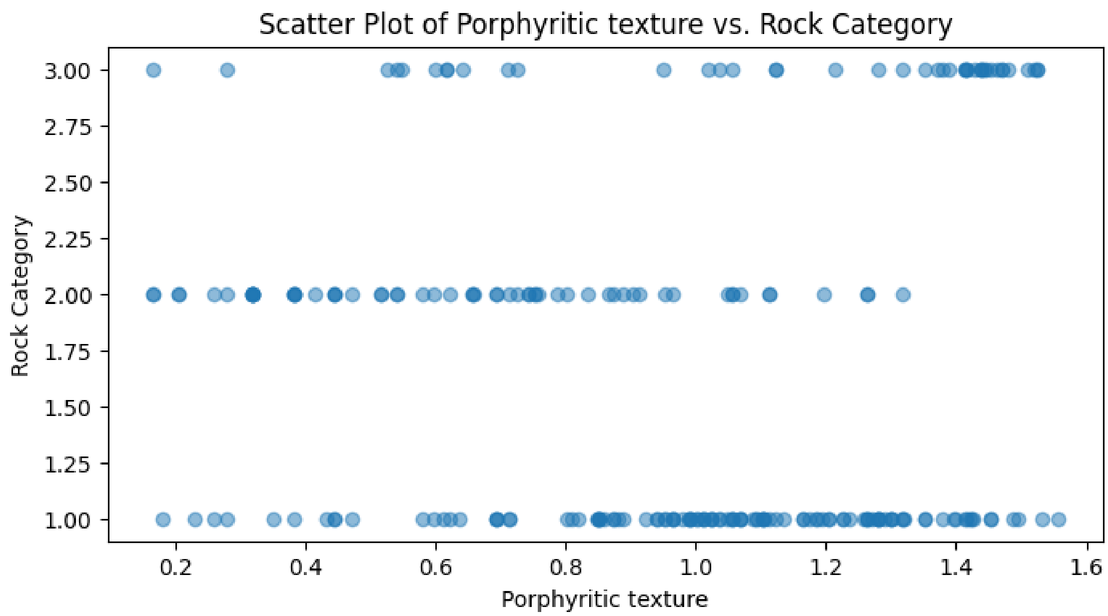
```
[ ]: # Compute PCC between label (rock category) and attributes
correlation_matrix = pd.concat([data, features], axis=1).corr()
correlation_with_label = correlation_matrix.iloc[0, 1:]

# Display PCC values for each attribute
print("Pearson Correlation Coefficients with Label:")
print(correlation_with_label)
```

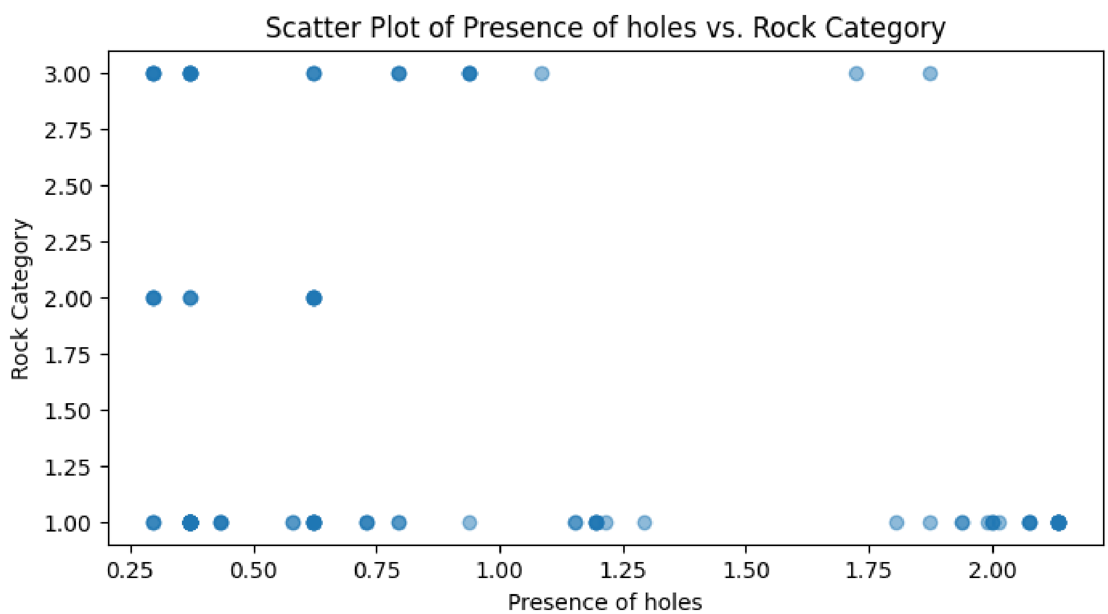
```
Pearson Correlation Coefficients with Label:
```

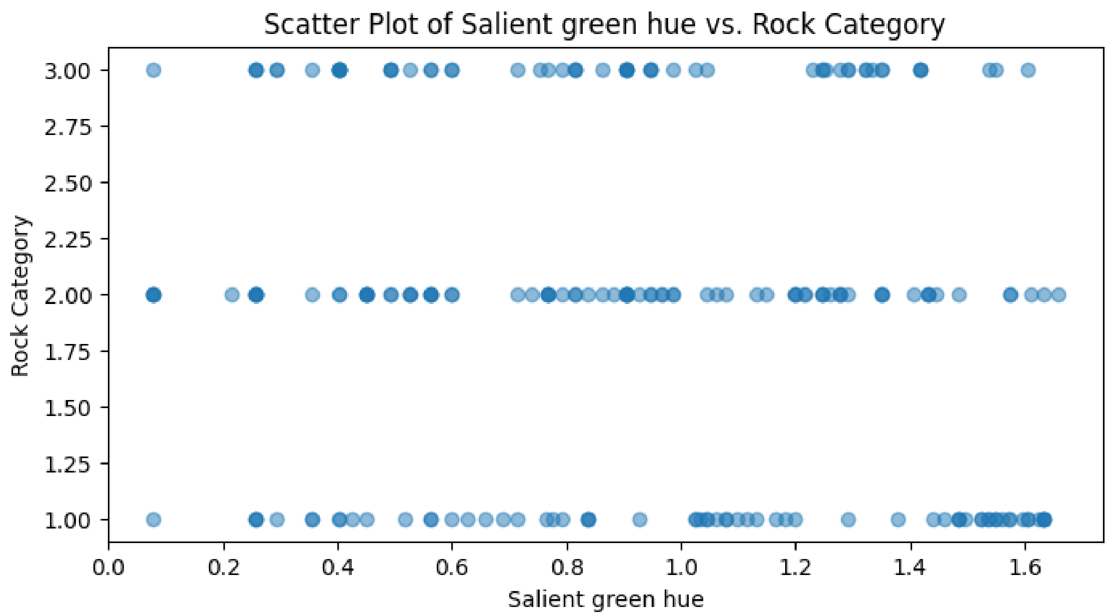
```
Porphyritic texture          -0.005857
Presence of holes            -0.290920
Salient green hue            -0.167203
Pegmatitic texture            0.000258
Conchoidal fracture          -0.139519
Angular fragments             0.240665
Rounded fragments             0.406914
Straight stripes              0.032348
Curved stripes               -0.166212
Physical layers              -0.080302
Veins                         0.029302
Oily/shimmery texture        -0.179055
Splotchy texture             -0.106322
Single translucent crystal    0.493506
Multiple cubic crystals       0.588832
Sandy texture                 0.283502
Fragments (disjunctive)       0.477796
Stripes (disjunctive)        -0.202934
Crystals (disjunctive)        0.697814
Name: rock category, dtype: float64
```

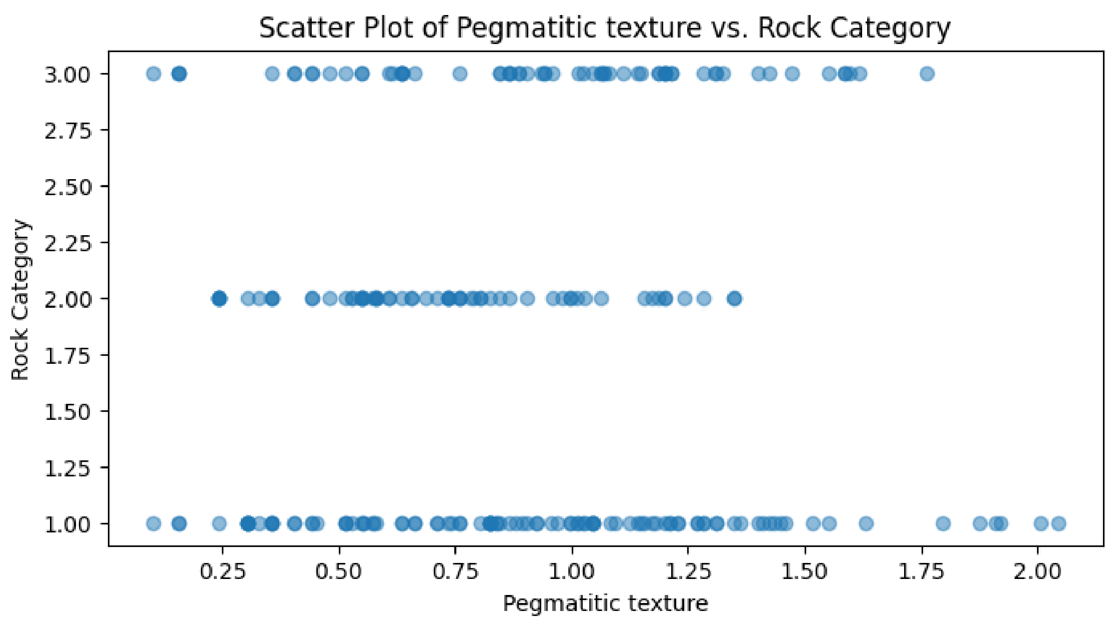
[ ]:

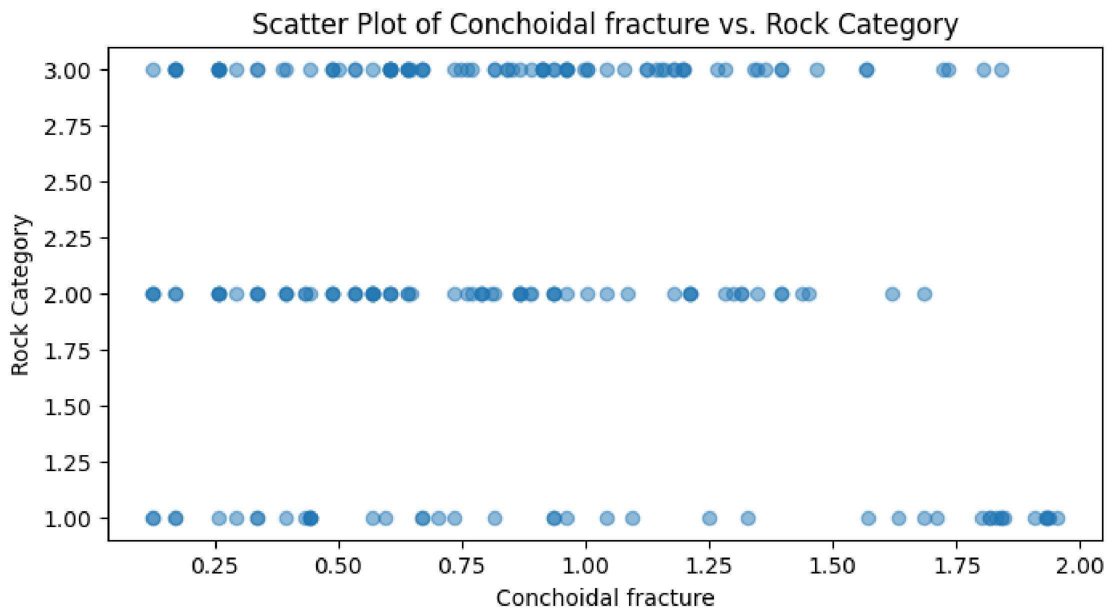
```
# Combine the label and features into a single DataFrame
df = pd.concat([data, features], axis=1)
# Generate scatter plots for each attribute against the "rock category" label
for column in features.columns:
    plt.figure(figsize=(8, 4))
    plt.scatter(df[column], df["rock category"], alpha=0.5)
    plt.title(f'Scatter Plot of {column} vs. Rock Category')
    plt.xlabel(column)
    plt.ylabel("Rock Category")
    plt.show()
```

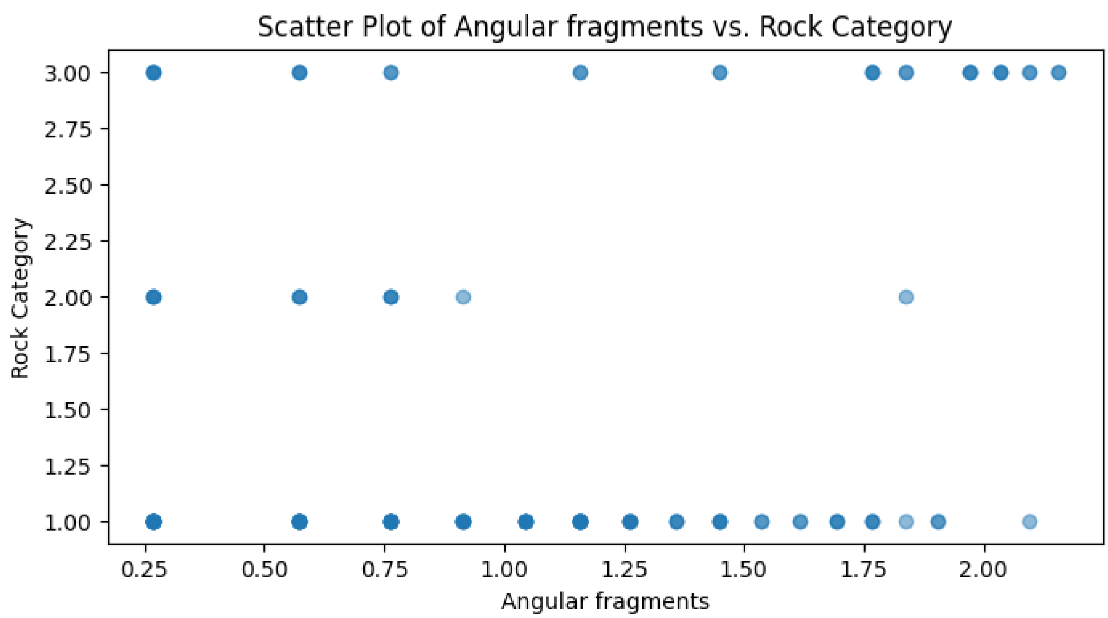

Scatter Plot of Porphyritic texture vs. Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
Rock Category
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Porphyritic texture

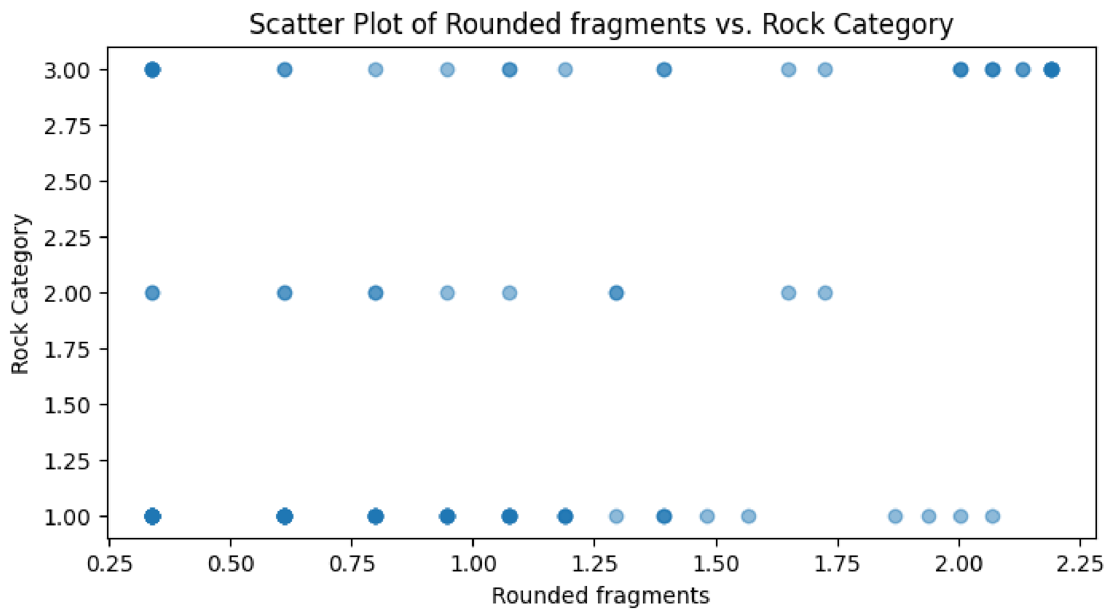

Scatter Plot of Presence of holes vs. Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
Rock Category
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Presence of holes

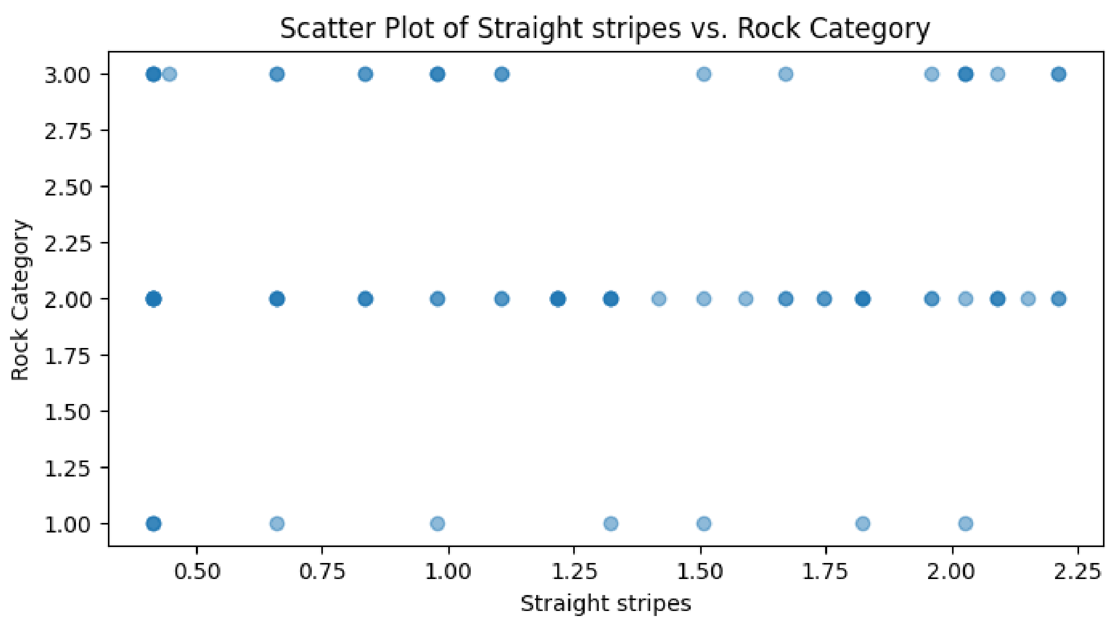

Scatter Plot of Salient green hue vs. Rock Category
Rock Category
Salient green hue
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

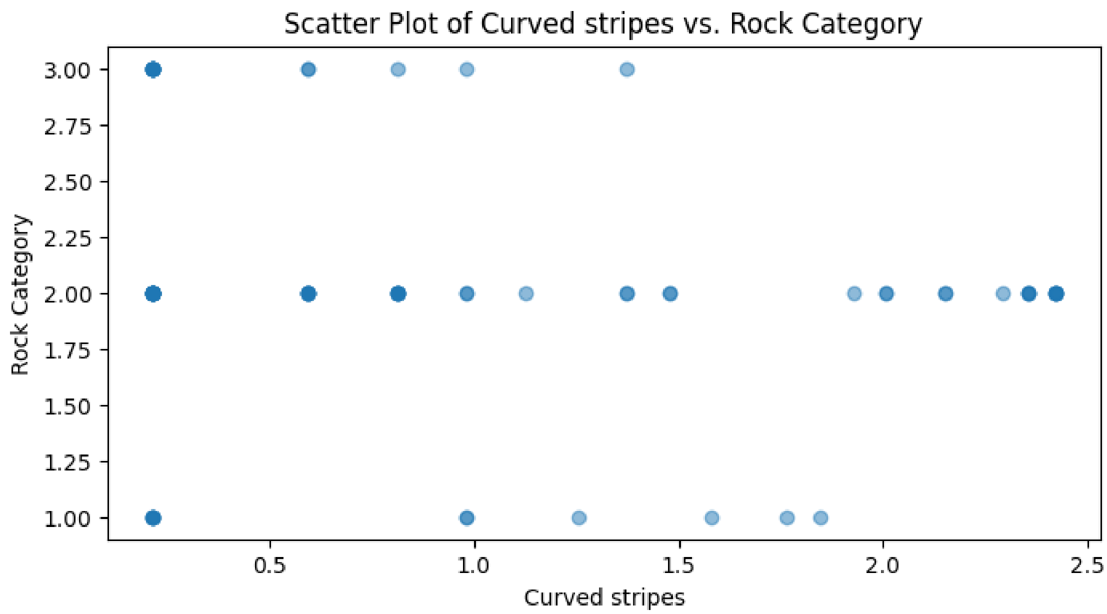

Scatter Plot of Pegmatitic texture vs. Rock Category
Rock Category
Pegmatitic texture
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

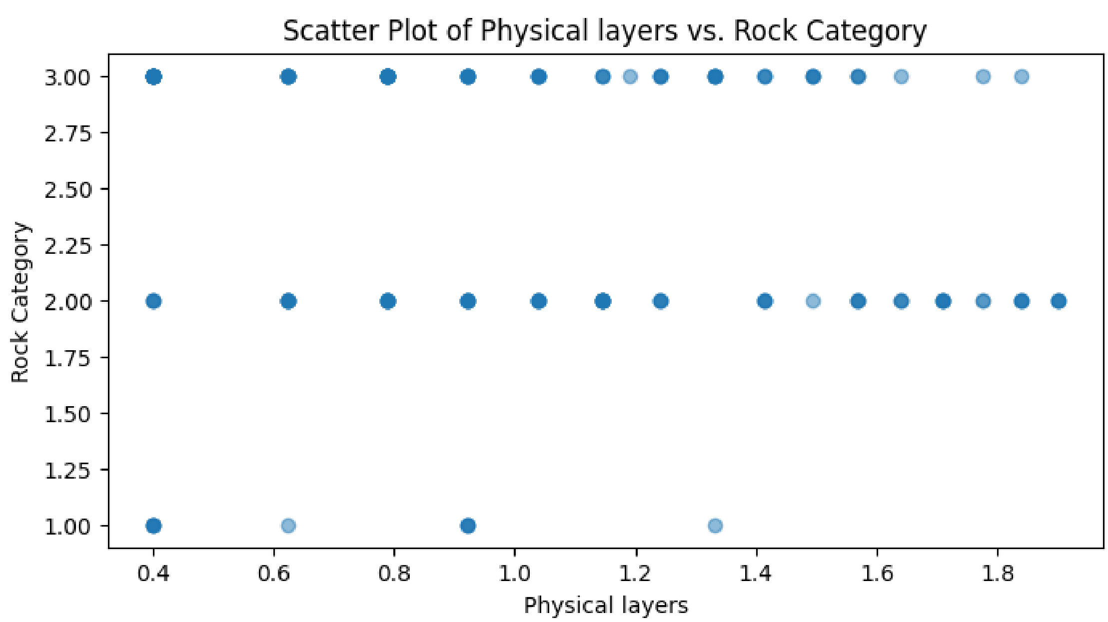

Scatter Plot of Conchoidal fracture vs. Rock Category
Rock Category
Conchoidal fracture
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

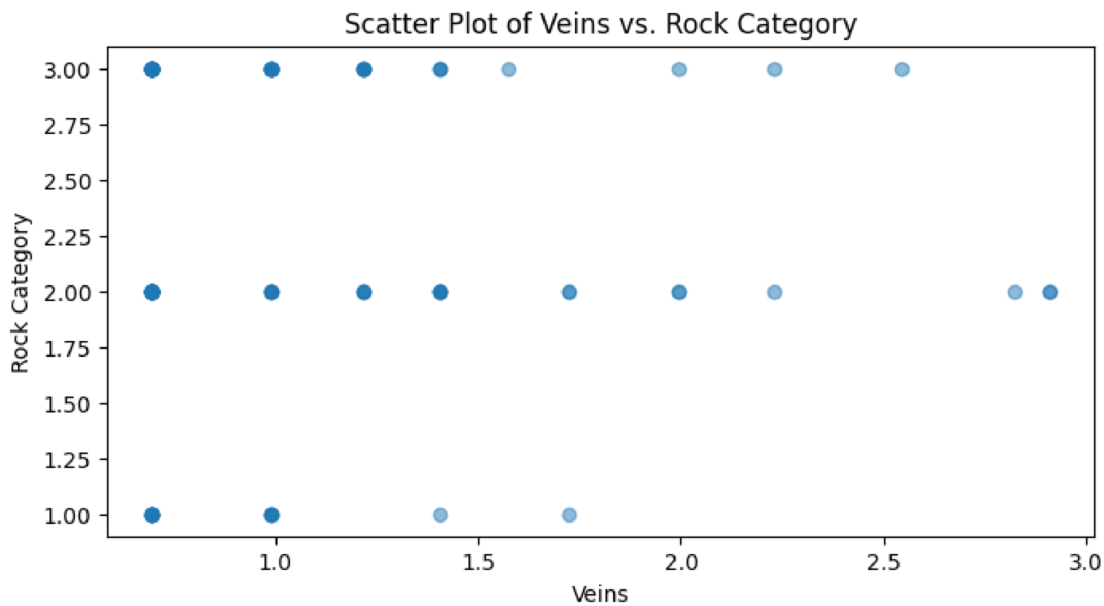

Scatter Plot of Angular fragments vs. Rock Category
Rock Category
Angular fragments
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Scatter Plot of Rounded fragments vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
Rounded fragments

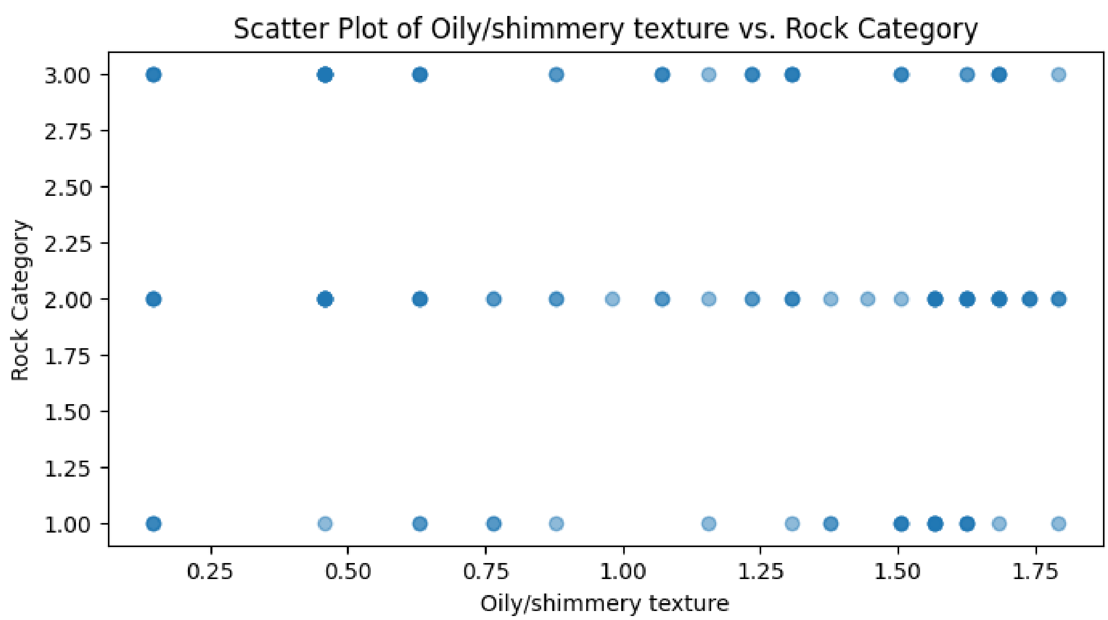

Scatter Plot of Straight stripes vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
Straight stripes


Scatter Plot of Curved stripes vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.5
1.0
1.5
2.0
2.5
Curved stripes

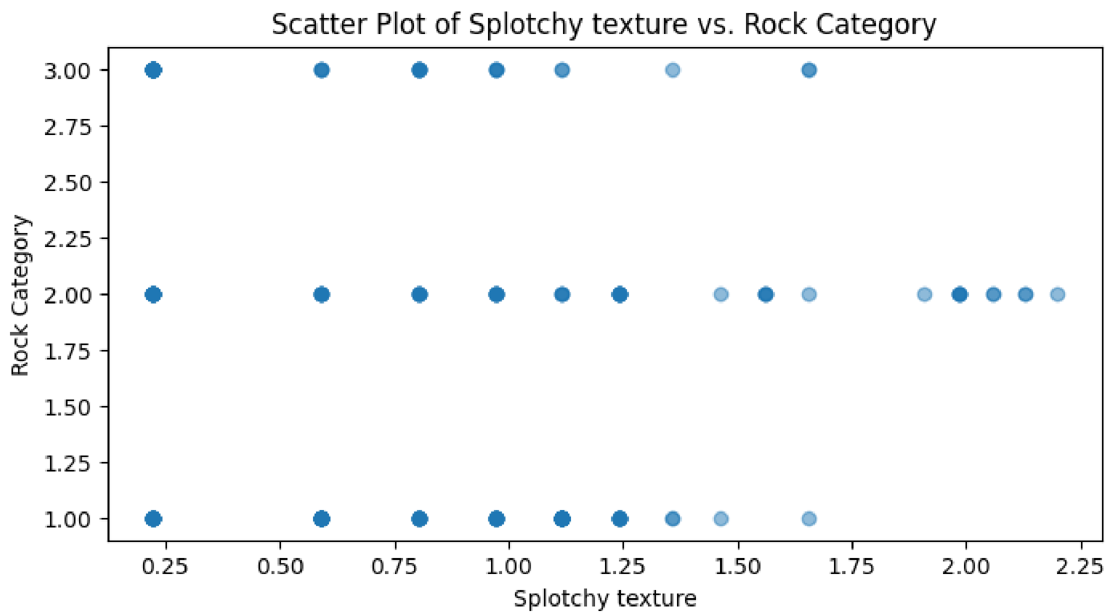

Scatter Plot of Physical layers vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Physical layers

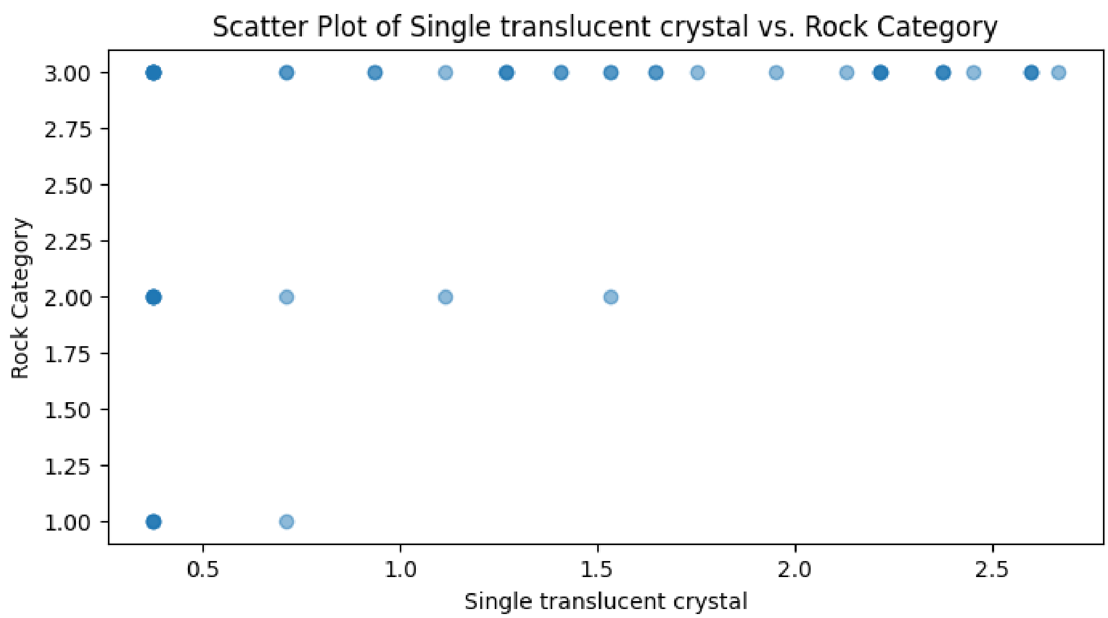


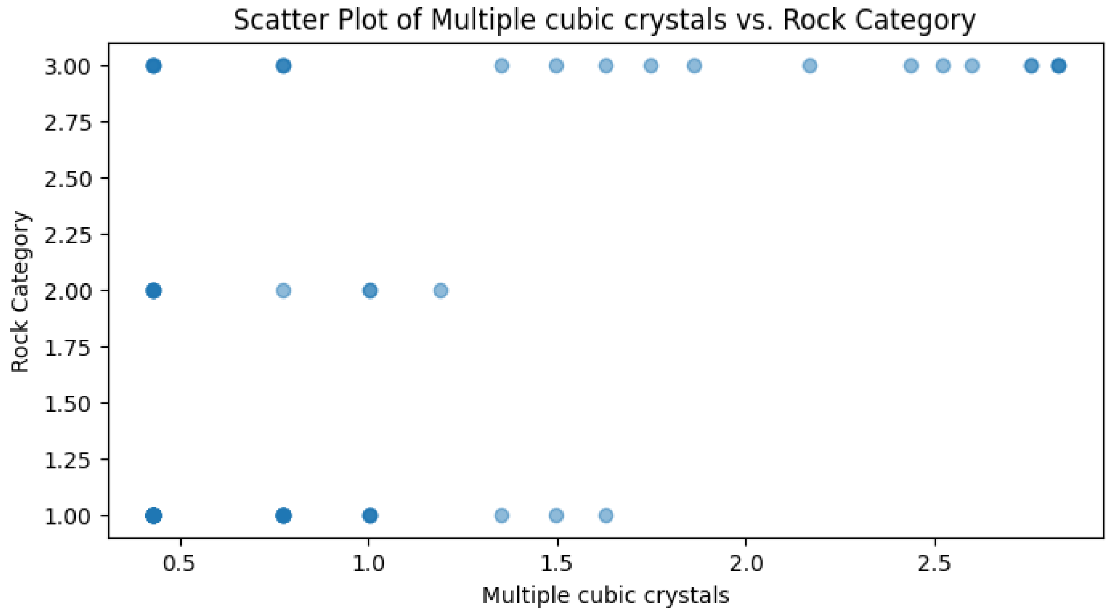
Scatter Plot of Veins vs. Rock Category
Rock Category
Veins
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
1.0
1.5
2.0
2.5
3.0

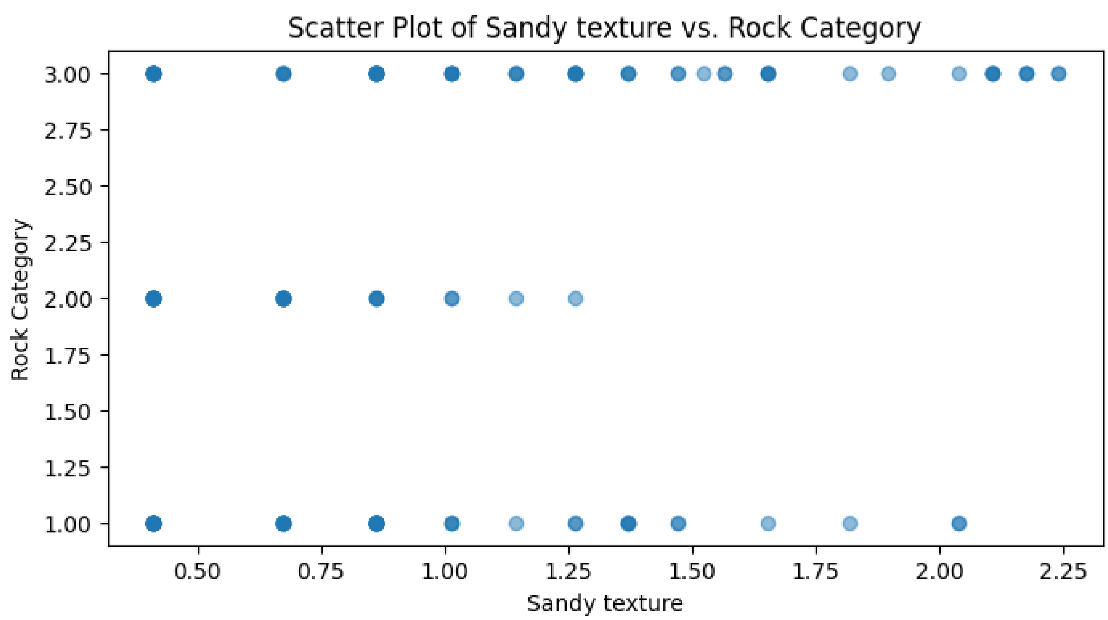


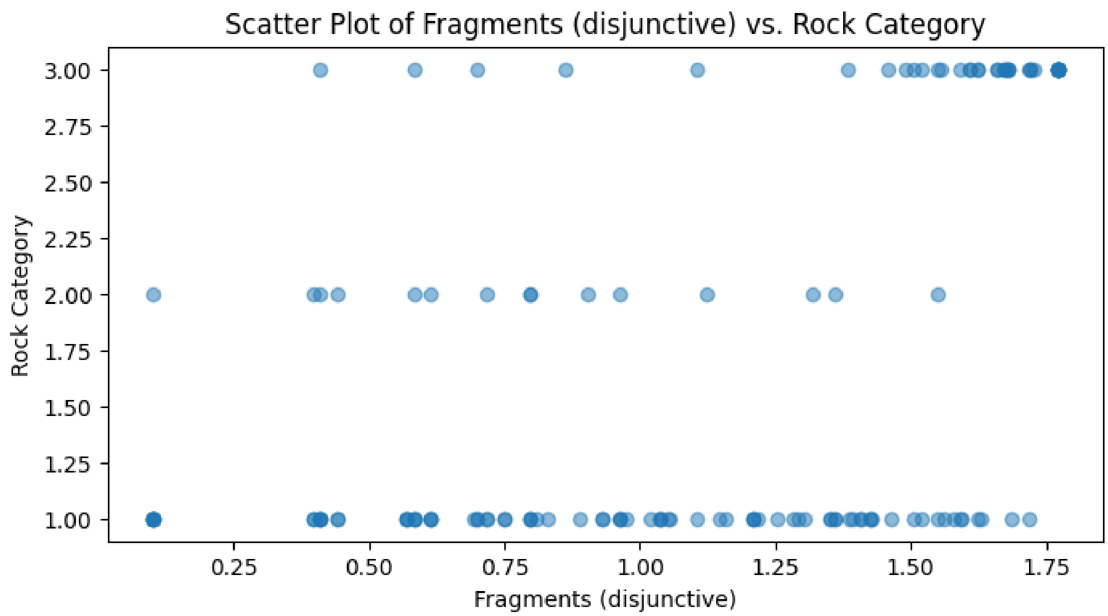
Scatter Plot of Oily/shimmery texture vs. Rock Category
Rock Category
Oily/shimmery texture
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

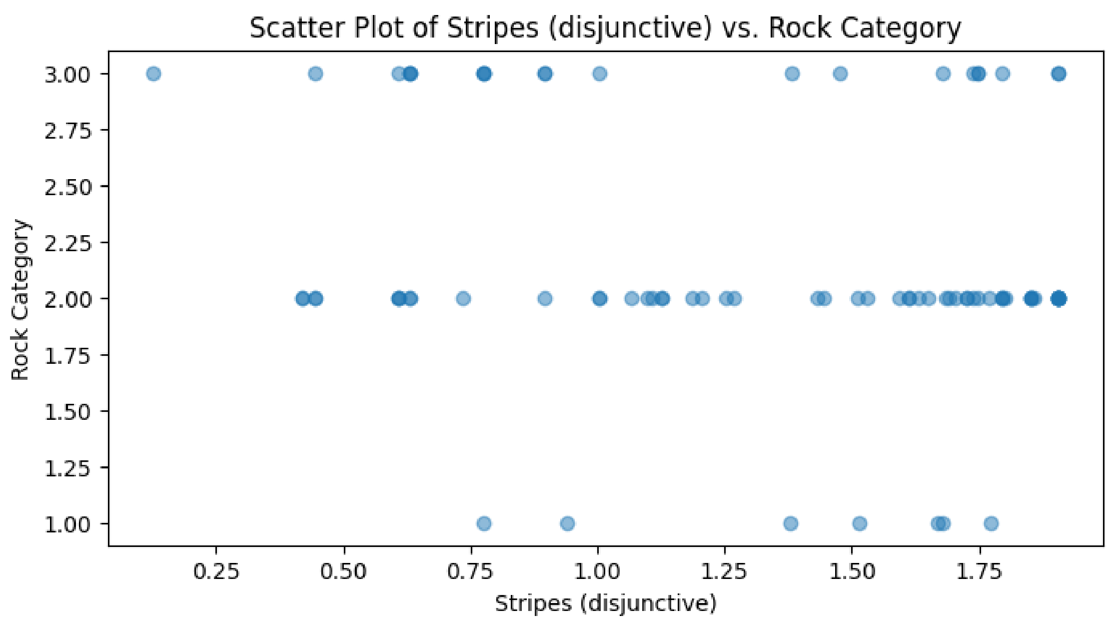

Scatter Plot of Splotchy texture vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
Splotchy texture

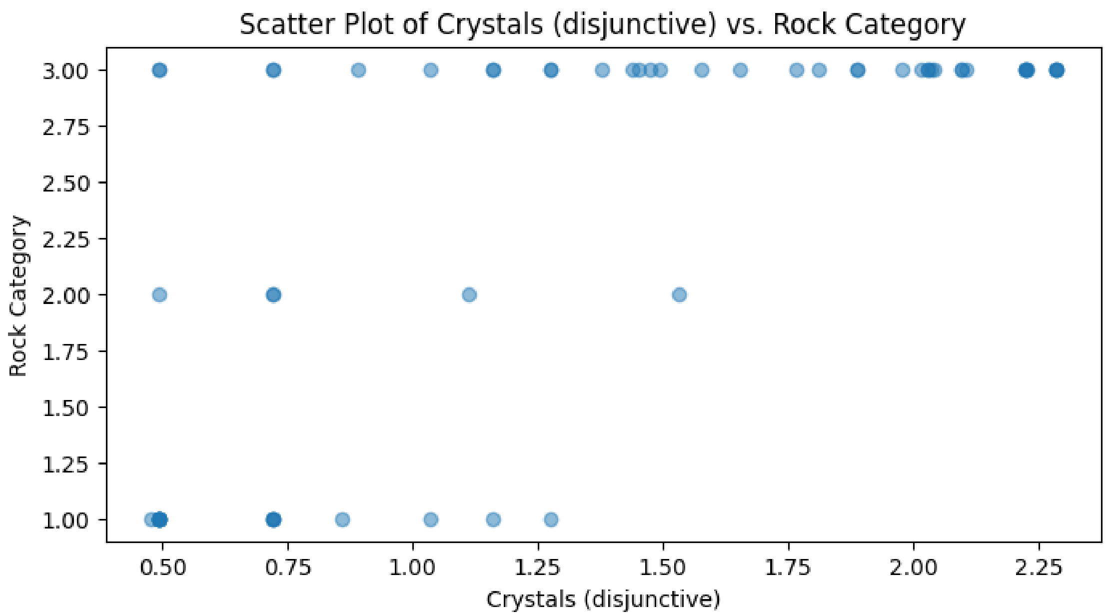

Scatter Plot of Single translucent crystal vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.5
1.0
1.5
2.0
2.5
Single translucent crystal


Scatter Plot of Multiple cubic crystals vs. Rock Category
Rock Category
Multiple cubic crystals
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.5
1.0
1.5
2.0
2.5


Scatter Plot of Sandy texture vs. Rock Category
Rock Category
Sandy texture
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25


Scatter Plot of Fragments (disjunctive) vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Fragments (disjunctive)


Scatter Plot of Stripes (disjunctive) vs. Rock Category
Rock Category
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Stripes (disjunctive)



Relationships between the attributes and the label:

- Strong positive correlations with the label are observed for attributes like "Crystals (disjunctive)," "Multiple cubic crystals," and "Single translucent crystal," indicating that higher values of these attributes correspond to higher values of the label.
- Moderate positive correlations are found for attributes such as "Presence of holes," "Sandy texture," and "Fragments (disjunctive)."
- Attributes like "Rounded fragments" and "Angular fragments" also exhibit positive correlations but with a more modest strength.
- On the other hand, attributes with negative correlations, such as "Veins," are inversely related to the label, with higher attribute values associated with lower label values.
- Some attributes, like "Porphyritic texture" and "Pegmatitic texture," exhibit minimal correlations with the label.

These correlation coefficients provide valuable insights into the relationships between attributes and the label, helping to identify which attributes are more influential in predicting the label's values.

Dropping the attributes which are less correlated with label

[ ]:
```
# Drop the specified columns from the DataFrame
features_updated=features.copy()
features_updated.drop(columns=[features_updated.columns[0], features_updated.
↪columns[3], features_updated.columns[7], features_updated.columns[9],␣
↪features_updated.columns[10]],axis=1)
```

[ ]:
```
     Salient green hue  Conchoidal fracture  Rounded fragments  \
0            -0.530724            -0.482150           0.375313
1             0.858984            -0.443857          -0.405184
2            -0.415333            -1.120369           4.017633
3             1.129901            -1.082076           3.757467
4            -0.570860            -0.609794           0.895644
..                 ...                  ...                ...
534          -0.435401            -1.126751          -0.405184
535          -0.957168             0.066717          -0.405184
536           0.066299            -1.088458          -0.405184
537          -0.029024             0.366680          -0.405184
538          -0.686250             1.387829          -0.405184

     Curved stripes  Oily/shimmery texture  Single translucent crystal  \
0         -0.260224              -0.540653                   -0.227922
1         -0.260224              -0.540653                   -0.227922
2         -0.260224              -0.540653                   -0.227922
3         -0.260224              -0.540653                   -0.227922
4         -0.260224              -0.353270                   -0.227922
..              ...                    ...                         ...
534       -0.260224              -0.540653                   -0.227922
535        0.045922              -0.540653                   -0.227922
536       -0.260224              -0.540653                   -0.227922
537       -0.260224              -0.353270                   -0.227922
538        0.045922              -0.165887                   -0.227922

     Fragments (disjunctive)  Stripes (disjunctive)  Crystals (disjunctive)
0                   2.042938              -0.409247               -0.034059
1                   1.665865              -0.409247               -0.310419
2                   2.640737              -0.409247               -0.310419
3                   2.659131              -0.409247               -0.310419
4                   1.481927              -0.409247               -0.310419
..                       ...                    ...                     ...
534                -0.541391              -0.409247                5.216791
535                -0.541391               3.054169               -0.310419
536                -0.541391              -0.409247               -0.310419
537                -0.541391              -0.409247               -0.310419
538                -0.541391               0.368255               -0.310419

[539 rows x 9 columns]
```

[ ]:
```
features_updated.head()
```

[ ]:
```
   Presence of holes  Salient green hue  Conchoidal fracture  \
0          -0.159688          -0.530724            -0.482150
1          -0.407623           0.858984            -0.443857
2          -0.407623          -0.415333            -1.120369
```

```
3          -0.159688          1.129901          -1.082076
4           0.385768         -0.570860          -0.609794

   Angular fragments  Rounded fragments  Curved stripes  \
0           2.865772           0.375313       -0.260224
1           2.611790          -0.405184       -0.260224
2           0.071962           4.017633       -0.260224
3           1.341876           3.757467       -0.260224
4           1.595858           0.895644       -0.260224

   Oily/shimmery texture  Splotchy texture  Single translucent crystal  \
0              -0.540653         -0.249084                   -0.227922
1              -0.540653          1.245422                   -0.227922
2              -0.540653         -0.249084                   -0.227922
3              -0.540653         -0.249084                   -0.227922
4              -0.353270          0.647619                   -0.227922

   Multiple cubic crystals  Sandy texture  Fragments (disjunctive)  \
0                 0.185510      -0.401124                 2.042938
1                -0.225045      -0.401124                 1.665865
2                -0.225045      -0.116312                 2.640737
3                -0.225045      -0.401124                 2.659131
4                -0.225045      -0.401124                 1.481927

   Stripes (disjunctive)  Crystals (disjunctive)
0              -0.409247               -0.034059
1              -0.409247               -0.310419
2              -0.409247               -0.310419
3              -0.409247               -0.310419
4              -0.409247               -0.310419
```

[ ]:
```python
# Calculate the median for each column in your DataFrame
med=data.median()
medians = features_updated.median()

# Replace null values with the respective column's median
data_filled=data.fillna(med)
features_filled = features_updated.fillna(medians)

# Now, features_filled contains no null values, and missing values have been
 ↪replaced with medians.
```

[ ]:
```python
features_updated.describe()
```

[ ]:
```
       Presence of holes  Salient green hue  Conchoidal fracture  \
count         539.000000         539.000000           539.000000
mean            0.000296           0.001199             0.001131
```

```
std              1.000905            1.000541             1.000583
min             -0.407623           -1.187950            -1.248012
25%             -0.407623           -0.761505            -0.699145
50%             -0.407623           -0.375197            -0.271538
75%             -0.159688            0.585557             0.360298
max              4.551072            2.750390             3.813059

       Angular fragments  Rounded fragments  Curved stripes  \
count         539.000000         539.000000      539.000000
mean           -0.001076          -0.000696        0.000483
std             1.000616           1.000798        1.000866
min            -0.436004          -0.405184       -0.260224
25%            -0.436004          -0.405184       -0.260224
50%            -0.436004          -0.405184       -0.260224
75%            -0.182021          -0.145018       -0.260224
max             4.643652           4.798130        5.862693

       Oily/shimmery texture  Splotchy texture  Single translucent crystal  \
count             539.000000        539.000000                  539.000000
mean                0.001003         -0.001756                    0.000423
std                 1.000657          1.000095                    1.000881
min                -0.540653         -0.846887                   -0.227922
25%                -0.540653         -0.846887                   -0.227922
50%                -0.540653         -0.249084                   -0.227922
75%                -0.165887          0.348718                   -0.227922
max                 3.207009          4.832237                    7.120010

       Multiple cubic crystals  Sandy texture  Fragments (disjunctive)  \
count               539.000000     539.000000               539.000000
mean                  0.000418       0.000216                -0.001180
std                   1.000882       1.000916                 1.000553
min                  -0.225045      -0.685937                -0.541391
25%                  -0.225045      -0.685937                -0.541391
50%                  -0.225045      -0.401124                -0.541391
75%                  -0.225045       0.168500                 0.001226
max                   7.986072       5.010309                 3.137369

       Stripes (disjunctive)  Crystals (disjunctive)
count             539.000000              539.000000
mean                0.000759                0.000576
std                 1.000773                1.000839
min                -0.409247               -0.310419
25%                -0.409247               -0.310419
50%                -0.409247               -0.310419
75%                -0.207298               -0.310419
max                 3.629722                5.216791
```

[ ]:
```
data.describe()
```

[ ]:
```
       rock category
count     539.000000
mean        2.001855
std         0.816874
min         1.000000
25%         1.000000
50%         2.000000
75%         3.000000
max         3.000000
```

3. Select 20% of the data for testing and 20% for validation and use the remaining 60% of the data for training. Describe how you did that and verify that your test and validation portions of the data are representative of the entire dataset.

Splitting the data in testing , validation and training sets correctly

[ ]:
```
from sklearn.model_selection import train_test_split

# Split the data into training (60%), validation (20%), and testing (20%)
X_train, X_temp, y_train, y_temp = train_test_split(features_updated, data,␣
 ↪test_size=0.4, random_state=42,shuffle=True)
X_validation, X_test, y_validation, y_test = train_test_split(X_temp, y_temp,␣
 ↪test_size=0.5, random_state=42)
```

Verification of splitting

[ ]:
```
X_train.describe()
```

[ ]:
```
       Presence of holes  Salient green hue  Conchoidal fracture  \
count         323.000000         323.000000           323.000000
mean            0.019470          -0.011690             0.022813
std             1.034486           1.009894             0.992720
min            -0.407623          -1.187950            -1.165044
25%            -0.407623          -0.781573            -0.654469
50%            -0.407623          -0.375197            -0.252392
75%            -0.159688           0.588066             0.366680
max             4.551072           2.750390             3.749237

       Angular fragments  Rounded fragments  Curved stripes  \
count         323.000000         323.000000      323.000000
mean            0.012987          -0.083803        0.003270
std             1.054023           0.845165        1.012893
min            -0.436004          -0.405184       -0.260224
25%            -0.436004          -0.405184       -0.260224
50%            -0.436004          -0.405184       -0.260224
75%            -0.182021          -0.145018       -0.260224
```

```
max             4.643652           4.798130        5.862693

       Oily/shimmery texture  Splotchy texture  Single translucent crystal  \
count             323.000000        323.000000                  323.000000
mean                0.024427          0.003547                   -0.003844
std                 1.023931          1.001127                    0.989319
min                -0.540653         -0.846887                   -0.227922
25%                -0.540653         -0.547986                   -0.227922
50%                -0.540653         -0.249084                   -0.227922
75%                -0.165887          0.348718                   -0.227922
max                 3.207009          4.832237                    7.120010

       Multiple cubic crystals  Sandy texture  Fragments (disjunctive)  \
count               323.000000     323.000000               323.000000
mean                  0.022814       0.030526                -0.040060
std                   1.075398       1.084515                 0.959862
min                  -0.225045      -0.685937                -0.541391
25%                  -0.225045      -0.685937                -0.541391
50%                  -0.225045      -0.401124                -0.541391
75%                  -0.225045       0.168500                -0.090743
max                   7.986072       5.010309                 3.137369

       Stripes (disjunctive)  Crystals (disjunctive)
count             323.000000              323.000000
mean                0.008846                0.012401
std                 1.010766                1.025927
min                -0.409247               -0.310419
25%                -0.409247               -0.310419
50%                -0.409247               -0.310419
75%                -0.207298               -0.310419
max                 3.629722                5.216791
```

[ ]:
```
X_test.describe()
```

[ ]:
```
       Presence of holes  Salient green hue  Conchoidal fracture  \
count         108.000000         108.000000           108.000000
mean            0.036364          -0.010907            -0.045266
std             1.094160           1.000053             0.985657
min            -0.407623          -1.147814            -1.248012
25%            -0.407623          -0.850557            -0.699145
50%            -0.407623          -0.299942            -0.293876
75%            -0.159688           0.412471             0.239036
max             4.551072           2.640017             3.640740

       Angular fragments  Rounded fragments  Curved stripes  \
count         108.000000         108.000000      108.000000
mean           -0.026809           0.158508       -0.002268
```

```
std              0.938079           1.245173         1.008152
min             -0.436004          -0.405184        -0.260224
25%             -0.436004          -0.405184        -0.260224
50%             -0.436004          -0.405184        -0.260224
75%             -0.182021          -0.145018        -0.260224
max              3.373738           4.798130         5.862693

       Oily/shimmery texture  Splotchy texture  Single translucent crystal  \
count             108.000000        108.000000                  108.000000
mean               -0.065255          0.038746                    0.020411
std                 0.933029          1.089590                    1.009385
min                -0.540653         -0.846887                   -0.227922
25%                -0.540653         -0.547986                   -0.227922
50%                -0.540653         -0.249084                   -0.227922
75%                -0.165887          0.348718                   -0.227922
max                 3.207009          4.533336                    6.017821

       Multiple cubic crystals  Sandy texture  Fragments (disjunctive)  \
count               108.000000     108.000000               108.000000
mean                 -0.137612      -0.060932                 0.066286
std                   0.307698       0.803175                 1.101150
min                  -0.225045      -0.685937                -0.541391
25%                  -0.225045      -0.401124                -0.541391
50%                  -0.225045      -0.401124                -0.541391
75%                  -0.225045      -0.045109                 0.010423
max                   1.827734       5.010309                 3.137369

       Stripes (disjunctive)  Crystals (disjunctive)
count             108.000000              108.000000
mean                0.014003               -0.070395
std                 1.020383                0.789449
min                -0.409247               -0.310419
25%                -0.409247               -0.310419
50%                -0.409247               -0.310419
75%                -0.207298               -0.310419
max                 3.629722                4.387709
```

[ ]:
```
X_validation.describe()
```

[ ]:
```
       Presence of holes  Salient green hue  Conchoidal fracture  \
count         108.000000         108.000000           108.000000
mean           -0.093113           0.051852            -0.017315
std             0.779900           0.980274             1.045349
min            -0.407623          -1.032423            -1.126751
25%            -0.407623          -0.686250            -0.708718
50%            -0.407623          -0.375197            -0.376845
75%            -0.159688           0.584303             0.322005
```

```
max              4.551072            2.670119               3.813059

       Angular fragments  Rounded fragments  Curved stripes  \
count         108.000000         108.000000      108.000000
mean           -0.017403           0.088649       -0.005102
std             0.899685           1.132048        0.965854
min            -0.436004          -0.405184       -0.260224
25%            -0.436004          -0.405184       -0.260224
50%            -0.436004          -0.405184       -0.260224
75%             0.071962          -0.145018       -0.260224
max             4.135686           4.798130        5.556547

       Oily/shimmery texture  Splotchy texture  Single translucent crystal  \
count             108.000000        108.000000                  108.000000
mean               -0.002794         -0.058120                   -0.006804
std                 1.001448          0.905929                    1.035433
min                -0.540653         -0.846887                   -0.227922
25%                -0.540653         -0.846887                   -0.227922
50%                -0.540653         -0.547986                   -0.227922
75%                -0.165887          0.647619                   -0.227922
max                 3.207009          3.935533                    6.752614

       Multiple cubic crystals  Sandy texture  Fragments (disjunctive)  \
count               108.000000     108.000000               108.000000
mean                  0.071467      -0.029286                 0.047636
std                   1.198391       0.919365                 1.018277
min                  -0.225045      -0.685937                -0.541391
25%                  -0.225045      -0.401124                -0.541391
50%                  -0.225045      -0.401124                -0.541391
75%                  -0.225045      -0.045109                 0.166770
max                   7.986072       4.725496                 3.137369

       Stripes (disjunctive)  Crystals (disjunctive)
count             108.000000              108.000000
mean               -0.036671                0.036183
std                 0.958309                1.113127
min                -0.409247               -0.310419
25%                -0.409247               -0.310419
50%                -0.409247               -0.310419
75%                -0.358760               -0.310419
max                 3.629722                5.216791
```

If we look at the data given by test and validate describe , the mean , median and standard deviation and the quartile range looks similiar. This means test, validate portion of the data is the representative of the entire dataset.

4.Train different classifiers and tweak the hyperparameters to improve performance (you can use the grid search if you want or manually try different values). Report

training, validation and testing performance (classification accuracy, precision, recall and F1 score) and discuss the impact of the hyperparameters

Multinomial Logistic Regression

Model is implemented correctly

[ ]:
```
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
import warnings
warnings.filterwarnings('ignore')
```

Different hyperparameters (C, solver,max number of iterations) have been tried

[ ]:
```
# Create a Multinomial Logistic Regression model
model1 = LogisticRegression(multi_class='multinomial')

# Define a parameter grid for hyperparameter tuning
param_grid1 = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [10, 100, 1000]
}
```

[ ]:
```
# Initialize GridSearchCV with the model and parameter grid
grid_search_logistic = GridSearchCV(model1, param_grid1, cv=5,␣
 ↪scoring='accuracy')

# Fit the GridSearchCV to the training data
grid_search_logistic.fit(X_train, y_train.values.ravel())
```

[ ]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(multi_class='multinomial'),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
                         'max_iter': [10, 100, 1000],
                         'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag',
                                    'saga']},
             scoring='accuracy')
```

Discussion on the impact of different hyper parameters has been done

[ ]:
```
print("Best Hyperparameters:")
print(grid_search_logistic.best_params_)
```

```
Best Hyperparameters:
{'C': 1, 'max_iter': 10, 'solver': 'lbfgs'}
```

I used grid search and found hyperparameters with C=1 and solver= newton-cg are the best ones.

Training, Validation and Testing Performance have been reported

```
[ ]: # Use the best model from the grid search
best_model_logistic= grid_search_logistic.best_estimator_

# Function to report performance metrics
def report_metrics(X, y, label):
    y_pred = best_model_logistic.predict(X)
    accuracy = accuracy_score(y, y_pred)
    precision = precision_score(y, y_pred, average='weighted')
    recall = recall_score(y, y_pred, average='weighted')
    f1 = f1_score(y, y_pred, average='weighted')
    print(f"{label} Performance Metrics:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

# Report performance metrics for training, validation, and testing
report_metrics(X_train, y_train, "Training")
report_metrics(X_validation, y_validation, "Validation")
report_metrics(X_test, y_test, "Testing")
```

```
Training Performance Metrics:
Accuracy: 0.6935
Precision: 0.6994
Recall: 0.6935
F1 Score: 0.6922
Validation Performance Metrics:
Accuracy: 0.5833
Precision: 0.6654
Recall: 0.5833
F1 Score: 0.5696
Testing Performance Metrics:
Accuracy: 0.6111
Precision: 0.6433
Recall: 0.6111
F1 Score: 0.6040
```

Support vector machines

Model is implemented correctly

```
[ ]: from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
  ↪f1_score
```

Different hyperparameters (C, Kernel, Gamma, degree) have been tried

[ ]:
```python
# Create an SVM model
model2 = SVC()

# Define a parameter grid for hyperparameter tuning
param_grid2 = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'degree': [2, 3, 4],  # Degree of polynomial kernel (only for 'poly' kernel)
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1, 10]
}
```

[ ]:
```python
# Initialize GridSearchCV with the model and parameter grid
grid_search_svm= GridSearchCV(model2, param_grid2, cv=5, scoring='accuracy')

# Fit the GridSearchCV to the training data
grid_search_svm.fit(X_train, y_train.values.ravel())
```

[ ]:
```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
                         'degree': [2, 3, 4],
                         'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1, 10],
                         'kernel': ['linear', 'poly', 'rbf', 'sigmoid']},
             scoring='accuracy')
```

Discussion on the impact of different hyper parameters has been done

[ ]:
```python
print("Best Hyperparameters:")
print(grid_search_svm.best_params_)
```

```
Best Hyperparameters:
{'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}
```

I used grid search and found hyperparameters with ‘C’: 10, ‘degree’: 2, ‘gamma’: ‘scale’, ‘kernel’: ‘rbf’ are the best ones.

Training, Validation and Testing Performance have been reported

[ ]:
```python
# Use the best model from the grid search
best_model_svm = grid_search_svm.best_estimator_

# Report performance metrics for training, validation, and testing
def report_metrics(X, y, label):
    y_pred = best_model_svm.predict(X)
    accuracy = accuracy_score(y, y_pred)
    precision = precision_score(y, y_pred, average='weighted')
    recall = recall_score(y, y_pred, average='weighted')
    f1 = f1_score(y, y_pred, average='weighted')
    print(f"{label} Performance Metrics:")
    print(f"Accuracy: {accuracy:.4f}")
```

```
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
report_metrics(X_train, y_train, "Training")
report_metrics(X_validation, y_validation, "Validation")
report_metrics(X_test, y_test, "Testing")
```

```
Training Performance Metrics:
Accuracy: 0.8607
Precision: 0.8605
Recall: 0.8607
F1 Score: 0.8606
Validation Performance Metrics:
Accuracy: 0.7500
Precision: 0.7822
Recall: 0.7500
F1 Score: 0.7504
Testing Performance Metrics:
Accuracy: 0.7315
Precision: 0.7544
Recall: 0.7315
F1 Score: 0.7336
```

Random Forest classifier

Model is implemented correctly

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
```

Different hyperparameters(no. of trees, max depth ,the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node) have been tried:

[ ]:
```
# Create a Random Forest classifier model
model3 = RandomForestClassifier()

# Define a parameter grid for hyperparameter tuning
param_grid3 = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

[ ]:
```
# Initialize GridSearchCV with the model and parameter grid
grid_search_rf= GridSearchCV(model3, param_grid3, cv=5, scoring='accuracy')

# Fit the GridSearchCV to the training data
grid_search_rf.fit(X_train, y_train.values.ravel())
```

[ ]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [None, 10, 20, 30],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [10, 50, 100, 200]},
             scoring='accuracy')
```

Discussion on the impact of different hyper parameters has been done

[ ]:
```
print("Best Hyperparameters:")
print(grid_search_rf.best_params_)
```

```
Best Hyperparameters:
{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators':
50}
```

I used grid search and found hyperparameters with 'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 5 are the best ones.

Training, Validation and Testing Performance have been reported

[ ]:
```
# Use the best model from the grid search
best_model_rf = grid_search_rf.best_estimator_

# Analyze feature importance
feature_importance = best_model_rf.feature_importances_

# Create a DataFrame to display feature importance
feature_importance_df = pd.DataFrame({'Feature': features.columns, 'Importance':
 ↪ feature_importance})
feature_importance_df = feature_importance_df.sort_values(by='Importance',␣
 ↪ascending=False)

# Print the feature importance
print("Feature Importance:")
print(feature_importance_df)

# Report performance metrics for training, validation, and testing
def report_metrics(X, y, label):
    y_pred = best_model_rf.predict(X)
    accuracy = accuracy_score(y, y_pred)
    precision = precision_score(y, y_pred, average='weighted')
    recall = recall_score(y, y_pred, average='weighted')
```

```
    f1 = f1_score(y, y_pred, average='weighted')
    print(f"{label} Performance Metrics:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
report_metrics(X_train, y_train, "Training")
report_metrics(X_validation, y_validation, "Validation")
report_metrics(X_test, y_test, "Testing")
```

```
Feature Importance:
                       Feature  Importance
0            Presence of holes    0.126270
2           Conchoidal fracture   0.125784
10                Sandy texture   0.112595
1            Salient green hue    0.104394
7              Splotchy texture   0.091667
11     Fragments (disjunctive)    0.084664
12       Stripes (disjunctive)    0.068407
13      Crystals (disjunctive)    0.067988
6        Oily/shimmery texture    0.057314
3            Angular fragments    0.053178
4            Rounded fragments    0.035871
5               Curved stripes    0.033687
9       Multiple cubic crystals   0.019805
8    Single translucent crystal   0.018376
Training Performance Metrics:
Accuracy: 0.9721
Precision: 0.9724
Recall: 0.9721
F1 Score: 0.9721
Validation Performance Metrics:
Accuracy: 0.6667
Precision: 0.7091
Recall: 0.6667
F1 Score: 0.6696
Testing Performance Metrics:
Accuracy: 0.7593
Precision: 0.7780
Recall: 0.7593
F1 Score: 0.7580
```

5. Combine your classifiers into an ensemble and try to outperform each individual classifier on the validation set (try to get above 80% accuracy). Once you have found a good one, try it on the test set. Describe and discuss your findings.

Ensemble classifier has been implemented via all the models with the best hyperparameters and Accuracy of the ensemble is greater than all the individual classifiers

Voting classifier(hard)

```
[ ]: from sklearn.ensemble import VotingClassifier

# Create a Voting Classifier with the best models
ensemble_model = VotingClassifier(estimators=[
    ('logistic', best_model_logistic),
    ('svm', best_model_svm),
    ('random_forest', best_model_rf)
], voting='hard')  # Use 'hard' voting to make decisions based on majority votes

# Fit the ensemble model to the training data
ensemble_model.fit(X_train, y_train.values.ravel())

# Evaluate the ensemble model on the validation set
y_val_pred = ensemble_model.predict(X_validation)

# Calculate the accuracy on the validation set
validation_accuracy = accuracy_score(y_validation, y_val_pred)

# Print the validation accuracy
print(f"Validation Accuracy: {validation_accuracy:.2f}")
```

```
Validation Accuracy: 0.69
```

Gradient Boosting Classifier

```
[ ]: from sklearn.ensemble import GradientBoostingClassifier

# Create a Gradient Boosting ensemble with the best models

ensemble = GradientBoostingClassifier(
    n_estimators=100,  # Adjust the number of estimators as needed
    learning_rate=0.05,  # Adjust the learning rate as needed
    random_state=42  # Set a random seed for reproducibility
)

# Fit the ensemble model on the training data
ensemble.fit(X_train, y_train.values.ravel())

# Evaluate the ensemble model on the validation set
y_val_pred1 = ensemble.predict(X_validation)

# Calculate the accuracy on the validation set
validation_accuracy = accuracy_score(y_validation, y_val_pred1)

# Print the validation accuracy
print(f"Validation Accuracy: {validation_accuracy:.2f}")
```

Validation Accuracy: 0.74

AdaBoost Classifier

[ ]:
```
from sklearn.ensemble import AdaBoostClassifier

# Create an AdaBoost ensemble with the best models
ensemble = AdaBoostClassifier(
    base_estimator=best_model_rf,  # Specify a base estimator (e.g., the best␣
↪Logistic Regression model)
    n_estimators=100,  # Adjust the number of estimators as needed
    learning_rate=0.05,# Adjust the learning rate as needed
    random_state=42,  # Set a random seed for reproducibility
)

# Fit the ensemble model on the training data
ensemble.fit(X_train, y_train.values.ravel())

# Evaluate the ensemble model on the validation set
y_val_pred = ensemble.predict(X_validation)

# Calculate the accuracy on the validation set
validation_accuracy = accuracy_score(y_validation, y_val_pred)

# Print the validation accuracy
print(f"Validation Accuracy: {validation_accuracy:.2f}")
```

Validation Accuracy: 0.70

Bagging Classifier

[ ]:
```
from sklearn.ensemble import BaggingClassifier

# Create a Bagging ensemble with the best models
ensemble = BaggingClassifier(
    base_estimator=best_model_svm,  # Replace with the best models
    n_estimators=10,  # Adjust the number of base estimators as needed
    random_state=42  # Set a random seed for reproducibility
)

# Fit the ensemble model on the training data
ensemble.fit(X_train, y_train.values.ravel())

# Evaluate the ensemble model on the validation set
y_val_pred = ensemble.predict(X_validation)

# Calculate the accuracy on the validation set
validation_accuracy = accuracy_score(y_validation, y_val_pred)
```

```
# Print the validation accuracy
print(f"Validation Accuracy: {validation_accuracy:.2f}")
```

```
Validation Accuracy: 0.76
```

Test set Accuracy

```
[ ]: y_test_pred = ensemble.predict(X_test)

# Calculate the accuracy on the test set
test_accuracy = accuracy_score(y_test, y_test_pred)

# Print the test accuracy
print(f"Test Accuracy: {test_accuracy:.2f}")
```

```
Test Accuracy: 0.74
```

Discussion on Findings

We found that ensemble method with bagging classifier gave best accuracy when compared with all other classifiers

we created a Bagging ensemble using the best SVM model as the base estimator. The Bagging ensemble has 10 base estimators. we reported the accuracy of this ensemble on the validation set and the test set. Bagging is a technique that combines multiple base models to reduce variance and enhance model stability.