Ue01_Diamonds

November 14, 2022

[ ]:
```python
import pandas as pd
import math

df = pd.read_csv('diamonds.csv')
df.describe()
```

1 Implementation Part 1 (50%): Diamond Prices

1.1 1. Give an overview of the dataset structure by answering those questions:

1.1.1 How many samples and features are in the dataset?

[2]:
```python
print('Samples:', len(df))
print('Features:',len(df.columns))
```

```
Samples: 53940
Features: 10
```

1.1.2 What are the feature data types?

[3]:
```python
df.dtypes
```

[3]:
```
carat      float64
cut         object
color       object
clarity     object
depth      float64
table      float64
price        int64
x          float64
y          float64
z          float64
dtype: object
```

1.1.3 Are diamonds balanced across color, cut and clarity? (Hint: roughly 1:1 means balanced, e.g. 1:2 is a “1:2 imbalance”)

[4]:
```
color_df = df.groupby('color').size().to_frame('count')
color_df['balance'] = color_df['count'] / color_df['count'].min()
color_df = color_df.sort_values('balance')
print(color_df)
```

```
       count   balance
color
J       2808  1.000000
I       5422  1.930912
D       6775  2.412749
H       8304  2.957265
F       9542  3.398148
E       9797  3.488960
G      11292  4.021368
```

Compared to the lowest count J, every class has at least an imbalance of 1:2 and most of them 1:3. E and F for example are balanced if viewed separately without the other colors.

[5]:
```
cut_df = df.groupby('cut').size().to_frame('count')
cut_df['balance'] = cut_df['count'] / cut_df['count'].min()
cut_df = cut_df.sort_values('balance')
print(cut_df)
```

```
           count    balance
cut
Fair        1610   1.000000
Good        4906   3.047205
Very Good  12082   7.504348
Premium    13791   8.565839
Ideal      21551  13.385714
```

Compared to the fair cut every class is imbalanced and ideal has an imbalance of 1:13

[6]:
```
clarity_df = df.groupby('clarity').size().to_frame('count')
clarity_df['balance'] = clarity_df['count'] / clarity_df['count'].min()
clarity_df = clarity_df.sort_values('balance')
print(clarity_df)
```

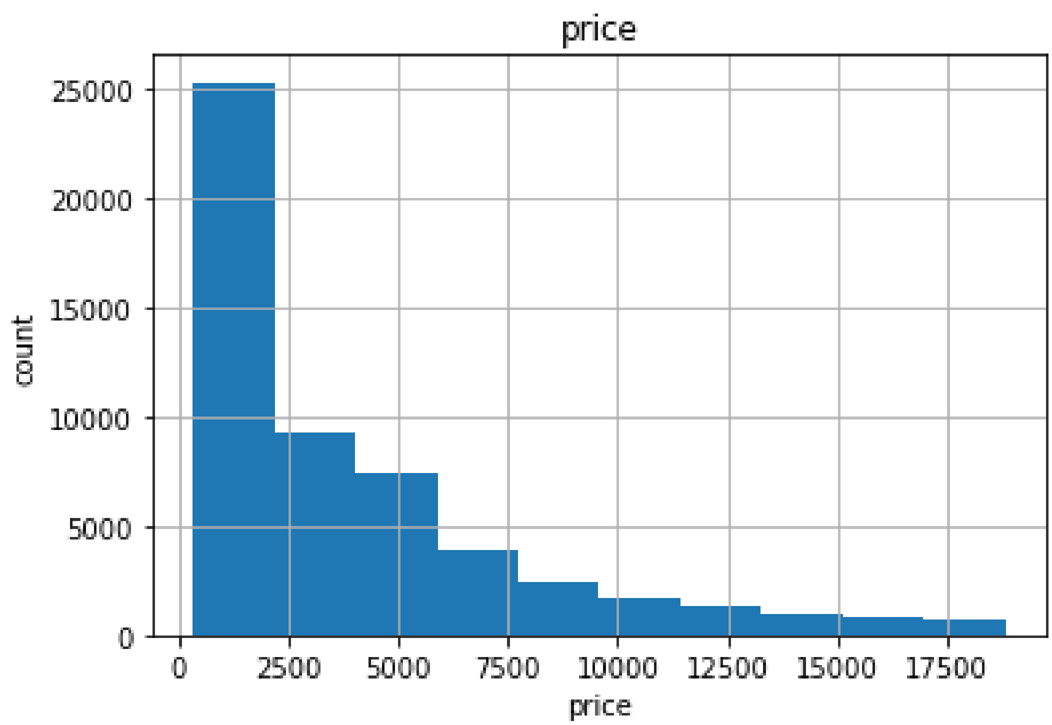
```
         count    balance
clarity
I1         741   1.000000
IF        1790   2.415655
VVS1      3655   4.932524
VVS2      5066   6.836707
VS1       8171  11.026991
SI2       9194  12.407557
VS2      12258  16.542510
```

```
SI1      13065  17.631579
```

The classes are extremely imbalanced, I1 vs VS2 has even an imbalance of 1:17.

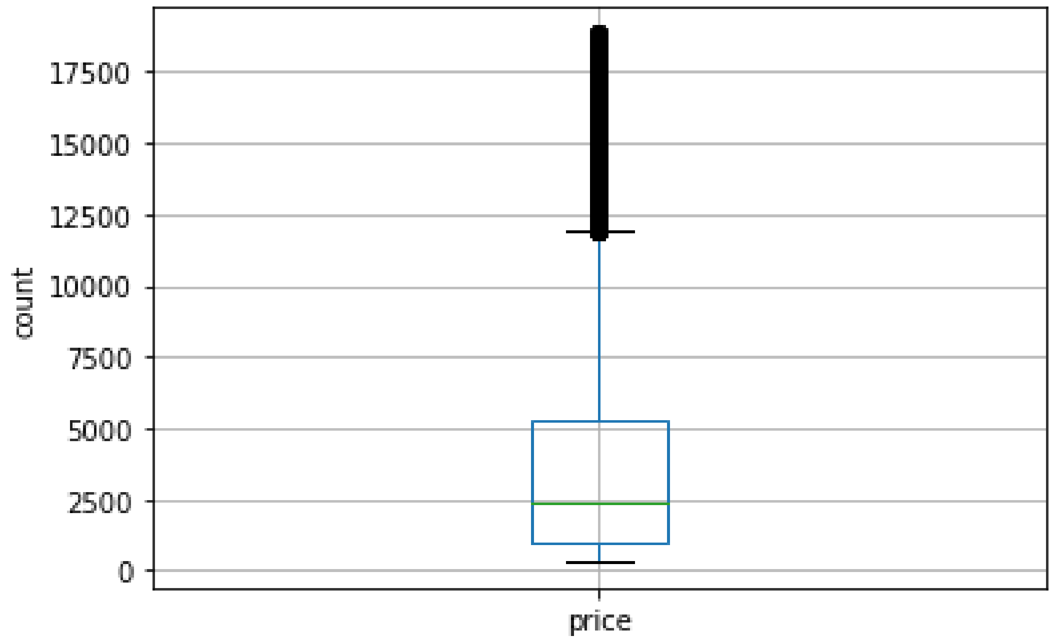
1.2 2. Visualize diamond prices using a histogram, boxplot and density plot.

[7]:
```
axarr = df.hist('price')
for ax in axarr.flatten():
    ax.set_xlabel("price")
    ax.set_ylabel("count")
```



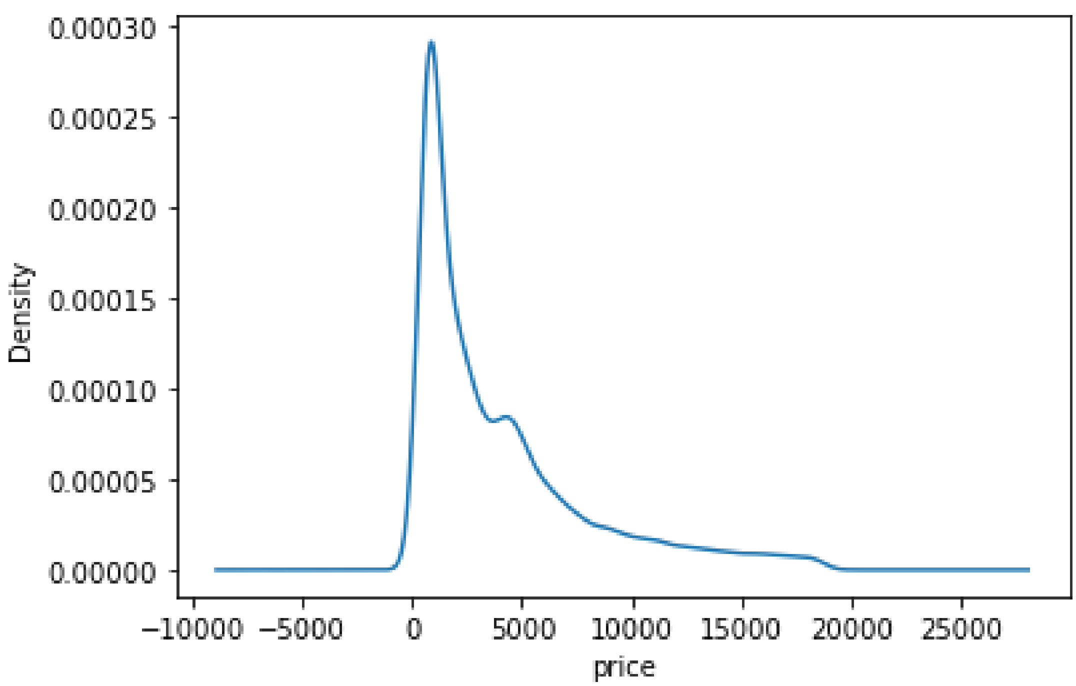
[8]:
```
axarr = df.boxplot('price')
axarr.set_ylabel("count")
```

[8]: `Text(0, 0.5, 'count')`



[9]:
```
axarr = df['price'].plot.density()
axarr.set_xlabel("price")
```

[9]: Text(0.5, 0, 'price')



1.2.1 Answer this question: Is there trend visible in those plots? If yes, which is it and in which plots can you see it?

There exist much more cheap diamonds up to 2500 and the amount decreases exponential which is very good visible in the boxplot and histogram. The boxplot also shows that there are many outliers beyond 12000. 75% of the diamonds are below 5000.

1.3 3. Calculate and state the mean, median, standard deviation, median absolute deviation (MAD), 1st and 3rd quartile (Q1 and Q3), and inner quartile range of the diamond price.

• If you are not familiar with those functions: use Google, Wikipedia, etc. • Required commands are all in the provided script.

[10]:
```
print('Mean:', round(df['price'].mean(), 2))
print('Median:', round(df['price'].median(), 2))
print('STD: ', round(df['price'].std(), 2))
# the mad() function is deprecated
print('MAD: ', round((df['price'] - df['price'].mean()).abs().mean(), 2))
print('Q1:', df['price'].quantile(0.25))
print('Q3:', df['price'].quantile(0.75))
print('Inner quartile: ', df['price'].quantile(0.75) - df['price'].quantile(0.
↪25))
```

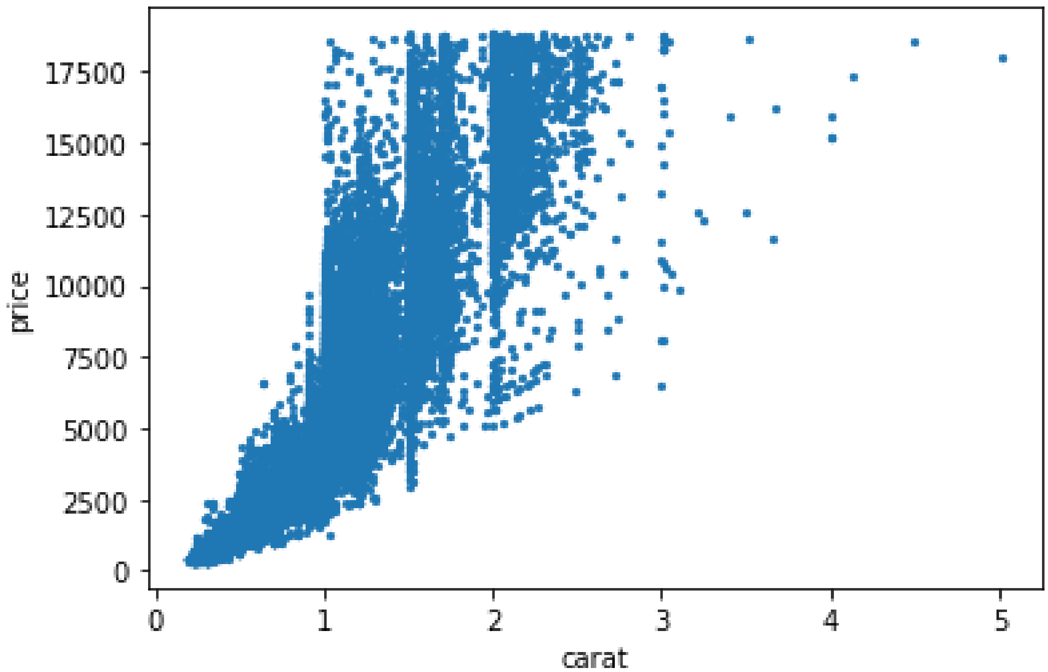
```
Mean: 3932.8
Median: 2401.0
STD:  3989.44
MAD:  3031.6
Q1: 950.0
Q3: 5324.25
Inner quartile:  4374.25
```

1.4 4. Plot the diamond price against the carat values as a scatterplot. Answer this question:

Hint: plotting many samples will be slow. Changing the plot symbol to '.' will cause a speedup.

[11]:
```
df.plot.scatter(x='carat', y='price', marker='.')
```

[11]:
```
<AxesSubplot:xlabel='carat', ylabel='price'>
```



1.4.1 Is there a trend visible in the plot? If yes, which is it?

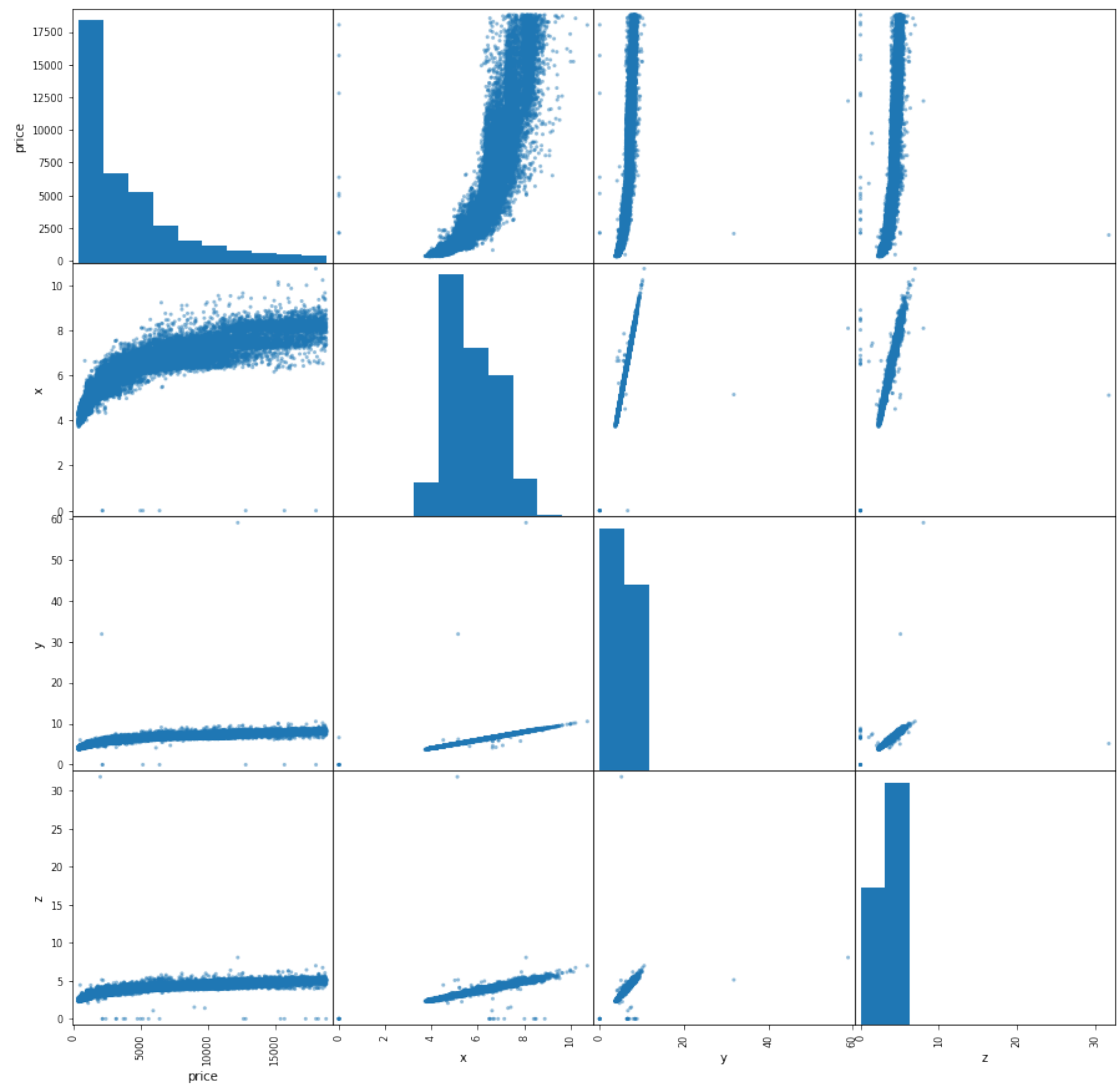
There is no direct connection between price and carat. It depends on more than the carat feature when you look on the spread of the points. Tendencial more carat leads to higher prices.

1.5 5. Analyze the correlation between diamond price and diamond x, y, and z dimensions.

1.5.1 Create pairwise plots for these features.

[12]:
```
pairwise_plot = df[['price', 'x', 'y', 'z']]
pd.plotting.scatter_matrix(pairwise_plot, figsize=(15,15))
print(pairwise_plot.corr())
```

```
          price         x         y         z
price  1.000000  0.884435  0.865421  0.861249
x      0.884435  1.000000  0.974701  0.970772
y      0.865421  0.974701  1.000000  0.952006
z      0.861249  0.970772  0.952006  1.000000
```



1.5.2 Is there a trend visible between x, y, and z? If yes, which is it?

Yes the dimensions all have a very high linear correlation, there is nearly a straight line.

1.5.3 Is there a trend visible between the dimensions and the price? If yes, which is it?

- Hint: if you don't know what a linear relation is (Google it!): – Linear correlation: feature A low –> feature B low, and feature A high –> feature B high. – (Inverse) linear correlation is also a linear correlation: feature A low –> feature B high, and feature A high –> feature B low: inverse linear correlation. Usually also just called linear correlation. – When plotting feature A against feature B and their points form a "straight line", then it's a linear relationship between A and B = linear correlation.

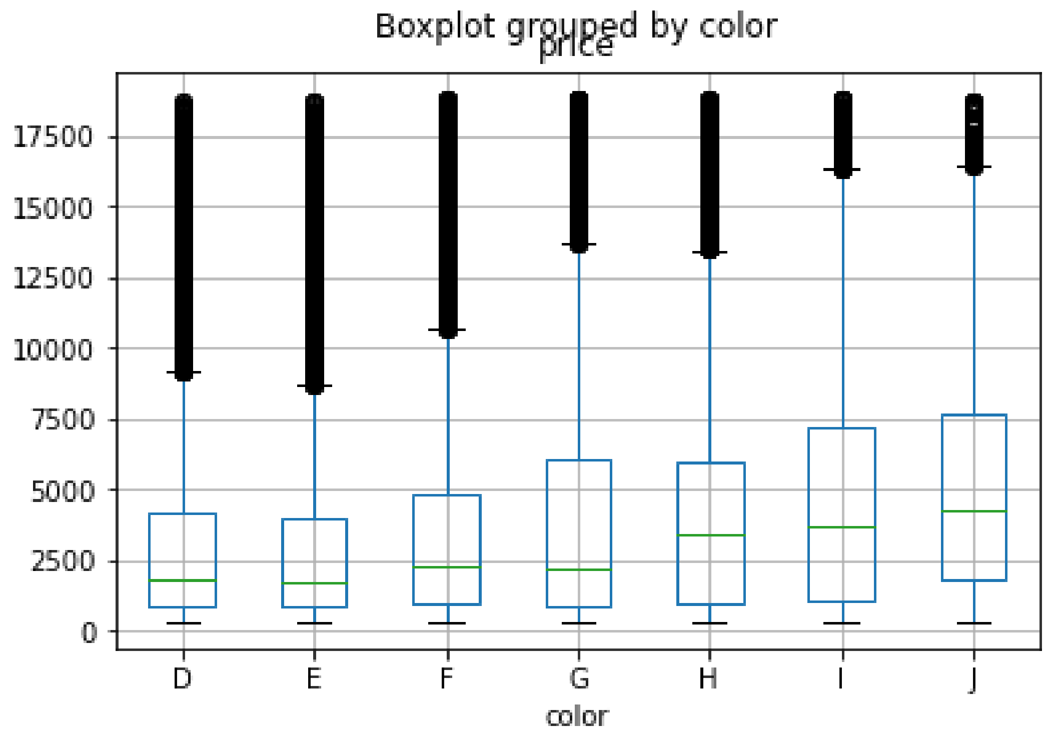
The larger the dimensions the greater the price. X spreads more than y or z. There is also a large linear correlation between price and dimension, at least 86%.

1.6 6. Analyze diamond prices per diamond color.

1.6.1 Create boxplots showing diamond price boxes for each diamond color (all boxes should be in one figure).

[13]:
```
df.boxplot('price', by='color')
```

[13]: <AxesSubplot:title={'center':'price'}, xlabel='color'>

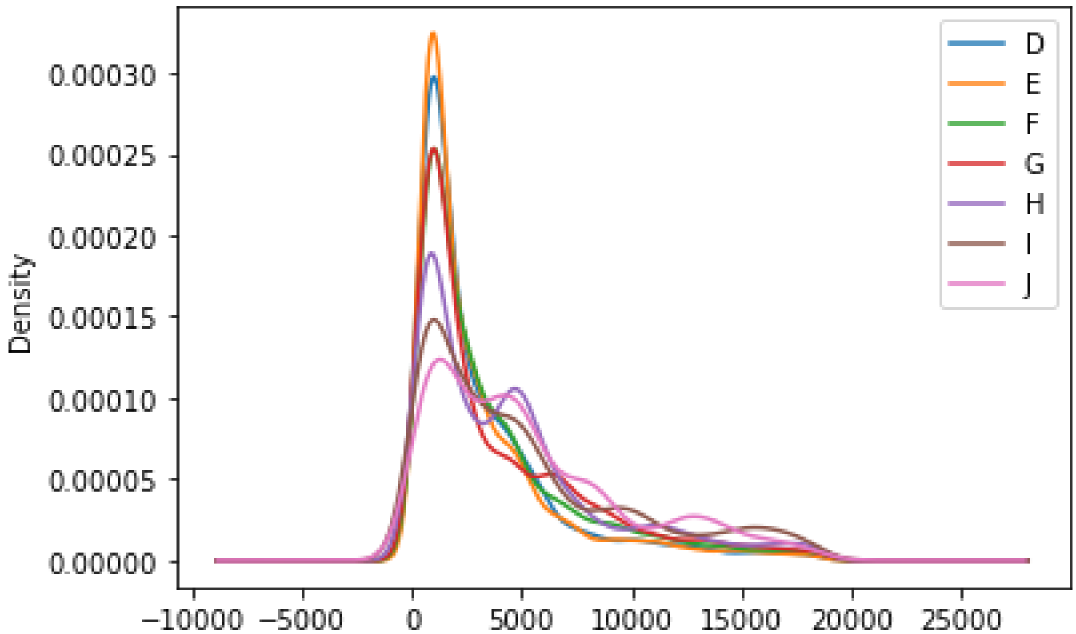


1.6.2 Create densityplots showing diamond prices for each diamond color (all densities should be in one figure).

[14]:
```
df.groupby('color')['price'].plot.density(legend=True)
```

[14]:
```
color
D    AxesSubplot(0.125,0.125;0.775x0.755)
E    AxesSubplot(0.125,0.125;0.775x0.755)
F    AxesSubplot(0.125,0.125;0.775x0.755)
G    AxesSubplot(0.125,0.125;0.775x0.755)
H    AxesSubplot(0.125,0.125;0.775x0.755)
I    AxesSubplot(0.125,0.125;0.775x0.755)
```

```
J    AxesSubplot(0.125,0.125;0.775x0.755)
Name: price, dtype: object
```



1.6.3 Answer this question: is there a trend visible? If yes, which one?

Most of the diamond colors are sold for approximately the same price. Depending on the color there exist more or less diamonds. Type E for example looks very stable and there are no peaks.

1.7 7. Use vectorized commands (= no loops!) to answer these questions:

1.7.1 How many diamonds have a price above 9500?

[15]:
```
df[df['price'] > 9500].count()
```

[15]:
```
carat       5734
cut         5734
color       5734
clarity     5734
depth       5734
table       5734
price       5734
x           5734
y           5734
z           5734
dtype: int64
```

1.7.2 How many diamonds have a price above 9500 and have color "D"?

[16]:
```
df[(df['price'] > 9500) & (df['color'] == 'D')].count()
```

[16]:
```
carat       461
cut         461
color       461
clarity     461
depth       461
table       461
price       461
x           461
y           461
z           461
dtype: int64
```

1.7.3 What is the mean and std of the price of all color "D" diamonds with cut "Fair"?

[17]:
```
df[(df['cut'] == 'Fair') & (df['color'] == 'D')]['price'].mean()
```

[17]:
```
4291.061349693252
```

[18]:
```
df[(df['cut'] == 'Fair') & (df['color'] == 'D')]['price'].std()
```

[18]:
```
3286.114238174996
```

1.7.4 What is the median and mad of the price of all color "J" diamonds with cut "Ideal"?

[19]:
```
dfIdeal = df[(df['cut'] == 'Ideal') & (df['color'] == 'J')]['price']
print('Median: ')
print(dfIdeal.median())
print('Ideal: ')
print((dfIdeal - dfIdeal.mean()).abs().mean())
```

```
Median:
4096.0
Ideal:
3467.586682378051
```

1.7.5 Create two copies of the dataframe that contains only the price and carat feature. Apply a log with base 10 to both features in one of those dataframes, and square (x' = x²) the features in the other dataframe. What is the mean and std of the transformed features in both dataframes?

[20]:

```
dfSquare = df[['carat', 'price']].applymap(func=lambda x: x ** 2)
print(dfSquare.mean())
print(dfSquare.std())
```

```
carat    8.613903e-01
price    3.138225e+07
dtype: float64
carat    1.056506e+00
price    6.049189e+07
dtype: float64
```

[21]:

```
dfLog = df[['carat', 'price']].applymap(func=math.log10)
print(dfLog.mean())
print(dfLog.std())
```

```
carat   -0.171532
price    3.381751
dtype: float64
carat    0.253987
price    0.440657
dtype: float64
```

[21]:

```

```

Ue01_CellBody

November 14, 2022

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df = pd.read_csv('segmentationData.csv')
```

1 Implementation Part 2 (50%): Cell Body Segmentation Data.

1.1 1. Which classes exist? Are they (roughly) balanced?

[2]:
```
df.describe()
```

[2]:
```
          AngleCh1     AreaCh1  AvgIntenCh1  AvgIntenCh2  AvgIntenCh3  \
count  2019.000000 2019.000000  2019.000000  2019.000000  2019.000000
mean     90.493405  320.336305   126.071679   189.052115    96.420171
std      48.760000  214.023533   165.008379   158.956105    96.666924
min       0.030876  150.000000    15.160400     1.000000     0.120000
25%      53.892205  193.000000    35.364158    44.998570    33.495693
50%      90.588770  253.000000    62.343173   173.506300    67.431250
75%     126.682013  362.500000   143.187800   279.289704   127.341651
max     179.939323 2186.000000  1418.634831   989.509800  1205.512000

       AvgIntenCh4  ConvexHullAreaRatioCh1  ConvexHullPerimRatioCh1  \
count  2019.000000             2019.000000              2019.000000
mean    140.701585                1.205859                 0.895764
std     146.634665                0.202522                 0.076108
min       0.563265                1.005831                 0.510623
25%      40.679740                1.065236                 0.856972
50%      90.250000                1.148620                 0.913262
75%     191.170410                1.280514                 0.955606
max     886.837500                2.900320                 0.996499

       DiffIntenDensityCh1  FiberAlign2Ch3  IntenCoocMaxCh3  IntenCoocMaxCh4  \
count          2019.000000     2019.000000      2019.000000      2019.000000
mean             72.660125        1.454076         0.231957         0.246709
std              49.028338        0.252347         0.204030         0.183398
min              25.760355        1.000000         0.014286         0.013423
```

```
25%            43.532759      1.290022      0.051171      0.107596
50%            55.810304      1.469231      0.179775      0.211886
75%            79.909902      1.647809      0.353311      0.337116
max           442.773196      2.000000      0.968326      0.940367

       NeighborMinDistCh1  SkewIntenCh4
count         2019.000000   2019.000000
mean            29.691933      0.932515
std             11.501550      0.885901
min             10.083350     -1.004442
25%             22.547068      0.403460
50%             27.642860      0.728311
75%             34.079173      1.225431
max            126.993700      8.069013
```

[3]:
```
df.dtypes
```

[3]:
```
Class                       object
AngleCh1                   float64
AreaCh1                      int64
AvgIntenCh1                float64
AvgIntenCh2                float64
AvgIntenCh3                float64
AvgIntenCh4                float64
ConvexHullAreaRatioCh1     float64
ConvexHullPerimRatioCh1    float64
DiffIntenDensityCh1        float64
FiberAlign2Ch3             float64
IntenCoocMaxCh3            float64
IntenCoocMaxCh4            float64
NeighborMinDistCh1         float64
SkewIntenCh4               float64
dtype: object
```

[4]:
```
color_df = df.groupby('Class').size().to_frame('count')
color_df['balance'] = color_df['count'] / color_df['count'].min()
color_df = color_df.sort_values('balance')
print(color_df)
```

```
       count   balance
Class
WS       719  1.000000
PS      1300  1.808067
```

The two classes WS and PS are not really balanced (imbalanced) with a ration of 1 : 1.8

1.2 2. Which noteworthy trends of features and relations between features as well as features and Class do you see?

[5]:
```python
df.loc[:, df.columns!='Class'].corr()
```

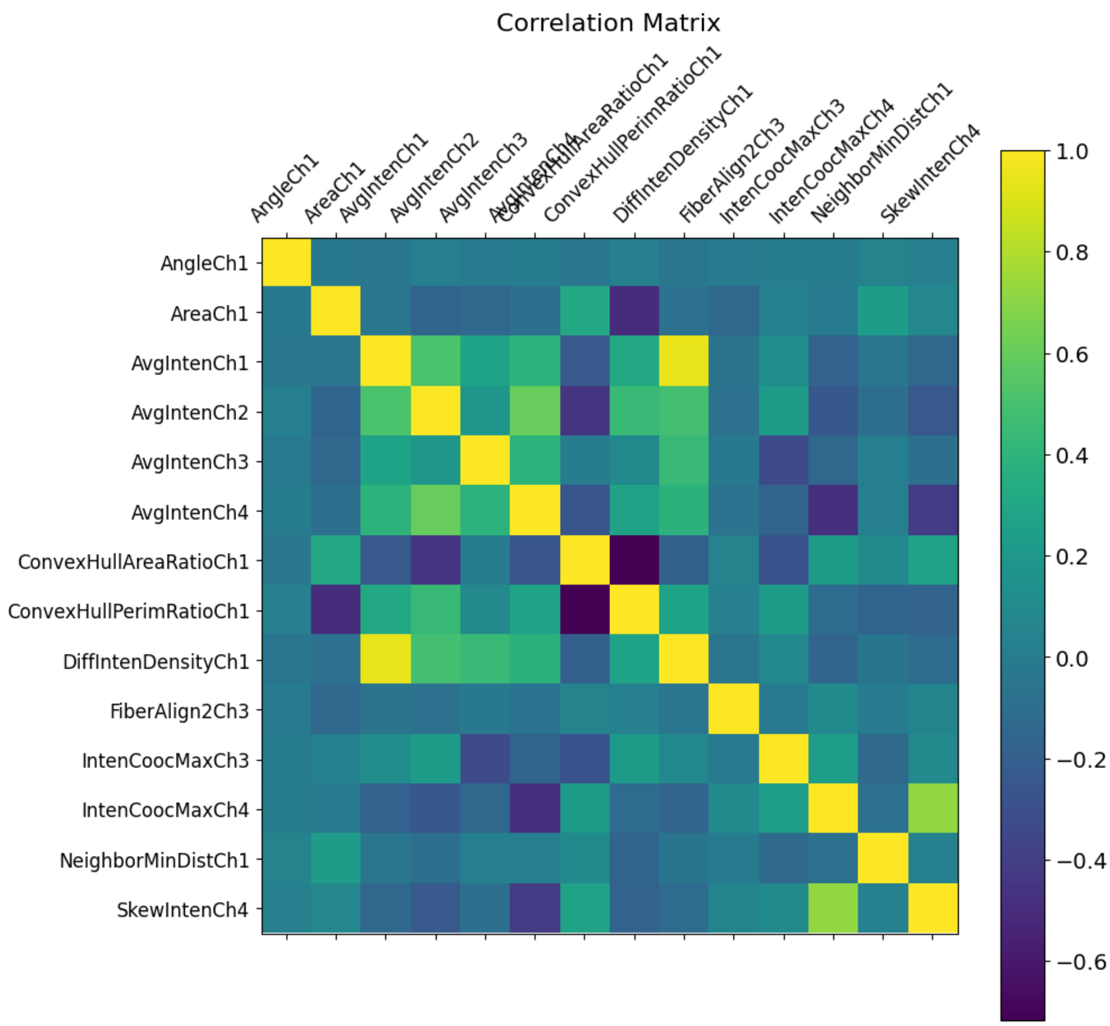
[5]:
```
                         AngleCh1   AreaCh1  AvgIntenCh1  AvgIntenCh2  \
AngleCh1                 1.000000 -0.025281    -0.026470     0.022270
AreaCh1                 -0.025281  1.000000    -0.039965    -0.163522
AvgIntenCh1             -0.026470 -0.039965     1.000000     0.516892
AvgIntenCh2              0.022270 -0.163522     0.516892     1.000000
AvgIntenCh3             -0.008911 -0.139592     0.276232     0.191390
AvgIntenCh4              0.006931 -0.084072     0.394118     0.599178
ConvexHullAreaRatioCh1  -0.039384  0.320712    -0.238587    -0.448929
ConvexHullPerimRatioCh1  0.032881 -0.489944     0.315972     0.438276
DiffIntenDensityCh1     -0.040770 -0.074990     0.942705     0.485847
FiberAlign2Ch3          -0.011776 -0.143195    -0.054437    -0.066706
IntenCoocMaxCh3          0.005589  0.038674     0.124660     0.226547
IntenCoocMaxCh4          0.009855 -0.015260    -0.176615    -0.242853
NeighborMinDistCh1       0.054098  0.231252    -0.043701    -0.095988
SkewIntenCh4             0.028063  0.075870    -0.141429    -0.236903

                         AvgIntenCh3  AvgIntenCh4  ConvexHullAreaRatioCh1  \
AngleCh1                   -0.008911     0.006931               -0.039384
AreaCh1                    -0.139592    -0.084072                0.320712
AvgIntenCh1                 0.276232     0.394118               -0.238587
AvgIntenCh2                 0.191390     0.599178               -0.448929
AvgIntenCh3                 1.000000     0.390760                0.007011
AvgIntenCh4                 0.390760     1.000000               -0.259174
ConvexHullAreaRatioCh1      0.007011    -0.259174                1.000000
ConvexHullPerimRatioCh1     0.089375     0.274304               -0.716921
DiffIntenDensityCh1         0.441698     0.386716               -0.193268
FiberAlign2Ch3             -0.020619    -0.062946                0.050550
IntenCoocMaxCh3            -0.326386    -0.165345               -0.283276
IntenCoocMaxCh4            -0.134638    -0.477101                0.216181
NeighborMinDistCh1          0.022199     0.024899                0.103121
SkewIntenCh4               -0.079956    -0.420889                0.274447

                         ConvexHullPerimRatioCh1  DiffIntenDensityCh1  \
AngleCh1                                0.032881            -0.040770
AreaCh1                                -0.489944            -0.074990
AvgIntenCh1                             0.315972             0.942705
AvgIntenCh2                             0.438276             0.485847
AvgIntenCh3                             0.089375             0.441698
AvgIntenCh4                             0.274304             0.386716
ConvexHullAreaRatioCh1                 -0.716921            -0.193268
ConvexHullPerimRatioCh1                 1.000000             0.276235
DiffIntenDensityCh1                     0.276235             1.000000
```

```
FiberAlign2Ch3                          0.027547            -0.046196
IntenCoocMaxCh3                         0.216211             0.087012
IntenCoocMaxCh4                        -0.118948            -0.160157
NeighborMinDistCh1                     -0.163567            -0.051174
SkewIntenCh4                           -0.169147            -0.114899

                         FiberAlign2Ch3  IntenCoocMaxCh3  IntenCoocMaxCh4  \
AngleCh1                      -0.011776         0.005589         0.009855
AreaCh1                       -0.143195         0.038674        -0.015260
AvgIntenCh1                   -0.054437         0.124660        -0.176615
AvgIntenCh2                   -0.066706         0.226547        -0.242853
AvgIntenCh3                   -0.020619        -0.326386        -0.134638
AvgIntenCh4                   -0.062946        -0.165345        -0.477101
ConvexHullAreaRatioCh1         0.050550        -0.283276         0.216181
ConvexHullPerimRatioCh1        0.027547         0.216211        -0.118948
DiffIntenDensityCh1           -0.046196         0.087012        -0.160157
FiberAlign2Ch3                 1.000000        -0.019200         0.107769
IntenCoocMaxCh3               -0.019200         1.000000         0.239047
IntenCoocMaxCh4                0.107769         0.239047         1.000000
NeighborMinDistCh1            -0.003185        -0.121486        -0.070880
SkewIntenCh4                   0.060125         0.096885         0.724893

                         NeighborMinDistCh1  SkewIntenCh4
AngleCh1                           0.054098      0.028063
AreaCh1                            0.231252      0.075870
AvgIntenCh1                       -0.043701     -0.141429
AvgIntenCh2                       -0.095988     -0.236903
AvgIntenCh3                        0.022199     -0.079956
AvgIntenCh4                        0.024899     -0.420889
ConvexHullAreaRatioCh1             0.103121      0.274447
ConvexHullPerimRatioCh1           -0.163567     -0.169147
DiffIntenDensityCh1               -0.051174     -0.114899
FiberAlign2Ch3                    -0.003185      0.060125
IntenCoocMaxCh3                   -0.121486      0.096885
IntenCoocMaxCh4                   -0.070880      0.724893
NeighborMinDistCh1                 1.000000      0.037793
SkewIntenCh4                       0.037793      1.000000
```

For example AvgIntenCh1 and DiffIntenDensityCh1 seem to have a high correlation.

[6]:
```
# https://stackoverflow.com/questions/29432629/
 ↪plot-correlation-matrix-using-pandas
f = plt.figure(figsize=(10, 10))
plt.matshow(df.loc[:, df.columns!='Class'].corr(), fignum=f.number)
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.
 ↪select_dtypes(['number']).columns, fontsize=12, rotation=45)
```

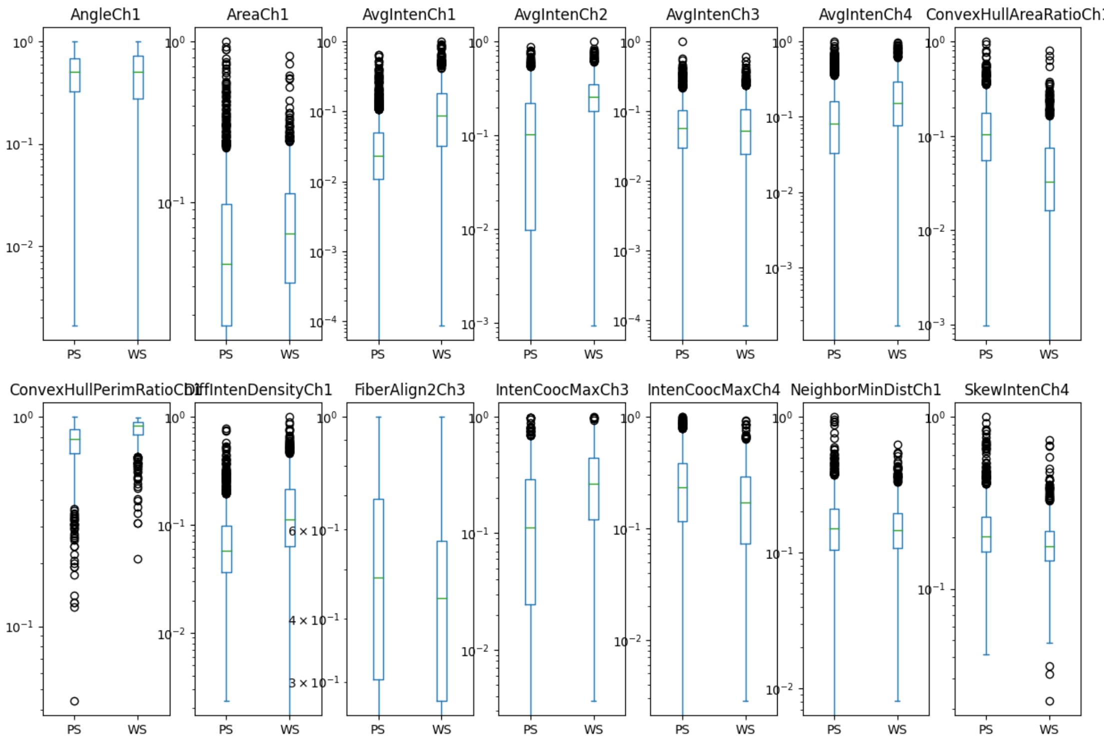
```
plt.yticks(range(df.select_dtypes(['number']).shape[1]), df.
 ↪select_dtypes(['number']).columns, fontsize=12)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16);
```



[7]:
```
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()
normalized_df = df.copy()
```

```
normalized_df[["AngleCh1", "AreaCh1", "AvgIntenCh1", "AvgIntenCh2",␣
↪"AvgIntenCh3", "AvgIntenCh4", "ConvexHullAreaRatioCh1",␣
↪"ConvexHullPerimRatioCh1", "DiffIntenDensityCh1", "FiberAlign2Ch3",␣
↪"IntenCoocMaxCh3", "IntenCoocMaxCh4", "NeighborMinDistCh1", "SkewIntenCh4"]]␣
↪= min_max_scaler.fit_transform(df[["AngleCh1", "AreaCh1", "AvgIntenCh1",␣
↪"AvgIntenCh2", "AvgIntenCh3", "AvgIntenCh4", "ConvexHullAreaRatioCh1",␣
↪"ConvexHullPerimRatioCh1", "DiffIntenDensityCh1", "FiberAlign2Ch3",␣
↪"IntenCoocMaxCh3", "IntenCoocMaxCh4", "NeighborMinDistCh1", "SkewIntenCh4"]])

normalized_df.plot.box(by="Class", figsize=(15,10), logy=True, layout=(2,7))
```

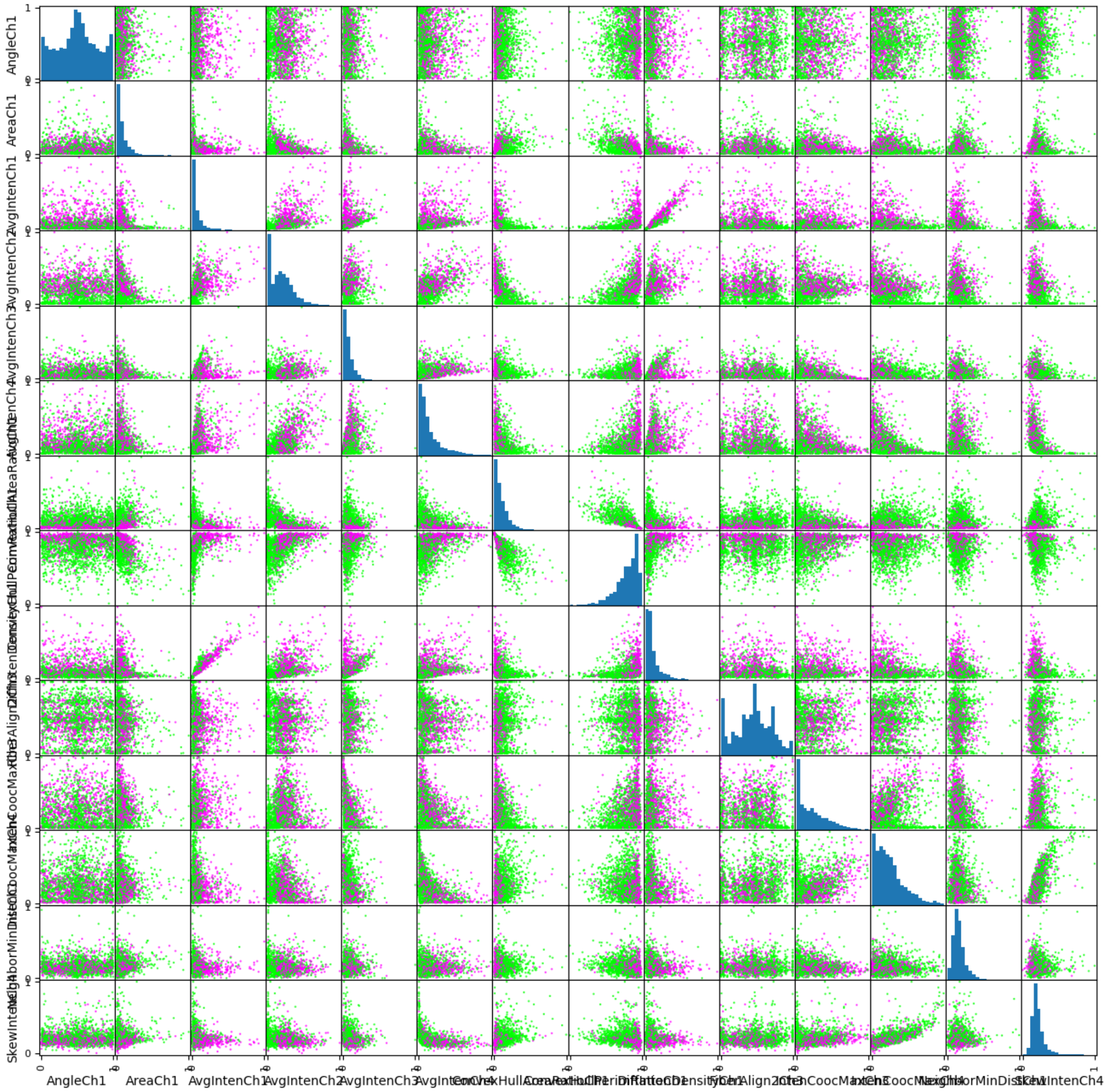
[7]:
```
AngleCh1                      AxesSubplot(0.125,0.53;0.0945122x0.35)
AreaCh1                    AxesSubplot(0.238415,0.53;0.0945122x0.35)
AvgIntenCh1                AxesSubplot(0.351829,0.53;0.0945122x0.35)
AvgIntenCh2                AxesSubplot(0.465244,0.53;0.0945122x0.35)
AvgIntenCh3                AxesSubplot(0.578659,0.53;0.0945122x0.35)
AvgIntenCh4                AxesSubplot(0.692073,0.53;0.0945122x0.35)
ConvexHullAreaRatioCh1     AxesSubplot(0.805488,0.53;0.0945122x0.35)
ConvexHullPerimRatioCh1       AxesSubplot(0.125,0.11;0.0945122x0.35)
DiffIntenDensityCh1        AxesSubplot(0.238415,0.11;0.0945122x0.35)
FiberAlign2Ch3             AxesSubplot(0.351829,0.11;0.0945122x0.35)
IntenCoocMaxCh3            AxesSubplot(0.465244,0.11;0.0945122x0.35)
IntenCoocMaxCh4            AxesSubplot(0.578659,0.11;0.0945122x0.35)
NeighborMinDistCh1         AxesSubplot(0.692073,0.11;0.0945122x0.35)
SkewIntenCh4               AxesSubplot(0.805488,0.11;0.0945122x0.35)
dtype: object
```



[8]:
```
df.groupby("Class").mean()
```

[8]:
```
        AngleCh1     AreaCh1  AvgIntenCh1  AvgIntenCh2  AvgIntenCh3  \
Class
PS     90.619486  314.339231    78.342220   138.852531    96.309678
WS     90.265441  331.179416   212.369728   279.816315    96.619950

       AvgIntenCh4  ConvexHullAreaRatioCh1  ConvexHullPerimRatioCh1  \
Class
PS      114.191873                1.255922                 0.875728
WS      188.632915                1.115343                 0.931988

       DiffIntenDensityCh1  FiberAlign2Ch3  IntenCoocMaxCh3  IntenCoocMaxCh4  \
Class
PS               60.748834        1.470619         0.189937         0.269191
WS               94.196536        1.424166         0.307932         0.206060

       NeighborMinDistCh1  SkewIntenCh4
Class
PS              30.063084      1.054383
WS              29.020866      0.712169
```

[9]:
```
columns = ["AngleCh1", "AreaCh1", "AvgIntenCh1", "AvgIntenCh2", "AvgIntenCh3",␣
 ↪"AvgIntenCh4", "ConvexHullAreaRatioCh1", "ConvexHullPerimRatioCh1",␣
 ↪"DiffIntenDensityCh1", "FiberAlign2Ch3", "IntenCoocMaxCh3",␣
 ↪"IntenCoocMaxCh4", "NeighborMinDistCh1", "SkewIntenCh4"]

test = lambda x: '#0f0' if x == "PS" else "#f0f"
vfunc = np.vectorize(test)
grr = pd.plotting.scatter_matrix(
    normalized_df[columns], c=vfunc(df.Class), figsize=(15,15), marker='.',
    hist_kwds={'bins':20}, s=10, alpha=.8)
```



[10]:
```
print('Mean:', round(df['AreaCh1'].mean(), 2))
print('Median:', round(df['AreaCh1'].median(), 2))
print('STD: ', round(df['AreaCh1'].std(), 2))
# the mad() function is deprecated
print('MAD: ', round((df['AreaCh1'] - df['AreaCh1'].mean()).abs().mean(), 2))
print('Q1:', df['AreaCh1'].quantile(0.25))
print('Q3:', df['AreaCh1'].quantile(0.75))
print('Inner quartile: ', df['AreaCh1'].quantile(0.75) - df['AreaCh1'].
 ↪quantile(0.25))
```

```
Mean: 320.34
Median: 253.0
STD:  214.02
```

```
MAD:  137.58
Q1: 193.0
Q3: 362.5
Inner quartile:  169.5
```

Some features like Angle1, AvgIntenCh3 or NeighborMinDistCh1 don't help to distinguish between the two classes PS and WS. AvgIntenCh2 for example has a different median and a different value range which could work very well to seperate the two classes.

1.3 3. If you would need to distinguish the classes with those features, which features would you choose, and why?

- AvgIntenCh1
- AvgIntenCh2
- DiffIntenDensityCh1
- ConvexHullAreaRatioCh1
- ConvexHullPerimRatioCh1

When you look at the median and inner quartile range of the boxplot for these features, the classes are easily separable.

```
[ ]:
```