ML Assignment 1

Haas Fabian, Schwarz Lea, Weingartshofer Florian

November 15, 2022

Disclaimer: some of the plots scale way better in the jupyter notebook itself.

1 Diamond Prices

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

[13]:
```
import pandas as pd
from pandas import DataFrame

df = pd.read_csv('diamonds.csv')
df.describe() # list some statistics for the number features in the dataset
```

[13]:

	carat	depth	table	price	x	\
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	
std	0.474011	1.432621	2.234491	3989.439738	1.121761	
min	0.200000	43.000000	43.000000	326.000000	0.000000	
25%	0.400000	61.000000	56.000000	950.000000	4.710000	
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	
max	5.010000	79.000000	95.000000	18823.000000	10.740000	

	y	z
count	53940.000000	53940.000000
mean	5.734526	3.538734
std	1.142135	0.705699
min	0.000000	0.000000
25%	4.720000	2.910000
50%	5.710000	3.530000
75%	6.540000	4.040000
max	58.900000	31.800000

1.1.1 How many samples and features are in the dataset?

[48]:
```
print('Number of samples: ', len(df))
print('Number of features: ', len(df.columns))
print('Features: ', df.columns)
```

```
Number of samples:  53940
Number of features:  10
Features:  Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price',
'x', 'y',
       'z'],
      dtype='object')
```

1.1.2 What are the feature data types?

[49]:
```
df.dtypes
```

[49]:
```
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")

[16]:
```
"""
Calculate the class distribution ratio of a feature by using the feature with␣
 ↪the least samples as the basis for the calculation
"""
def ratio(df: DataFrame, feature: str):
    g = df.groupby([feature]).size().to_frame(name = "samples")
    min_samples = g.iloc[:, 0].min()
    g['ratio'] = round(g.iloc[:, 0] / min_samples, 1)
    return g
```

[17]:
```
ratio(df, "color")
```

[17]:

color	samples	ratio
D	6775	2.4
E	9797	3.5
F	9542	3.4
G	11292	4.0
H	8304	3.0
I	5422	1.9

```
J         2808    1.0
```

Color has an imbalance of 1:4 from class J to F and over half the classes have at least an imbalance of 1:3 when compared to J

[18]:
```
ratio(df, "cut")
```

[18]:
```
           samples  ratio
cut
Fair          1610    1.0
Good          4906    3.0
Ideal        21551   13.4
Premium      13791    8.6
Very Good    12082    7.5
```

Cut has an even higher imbalance of 1:13.4 between Fair and Ideal. in general cut is very unbalanced

[19]:
```
ratio(df, "clarity")
```

[19]:
```
         samples  ratio
clarity
I1           741    1.0
IF          1790    2.4
SI1        13065   17.6
SI2         9194   12.4
VS1         8171   11.0
VS2        12258   16.5
VVS1        3655    4.9
VVS2        5066    6.8
```

Clarity has the worst imbalance of 1:17.6 between I1 and SI1 and apart from that is the most imbalanced feature of the three

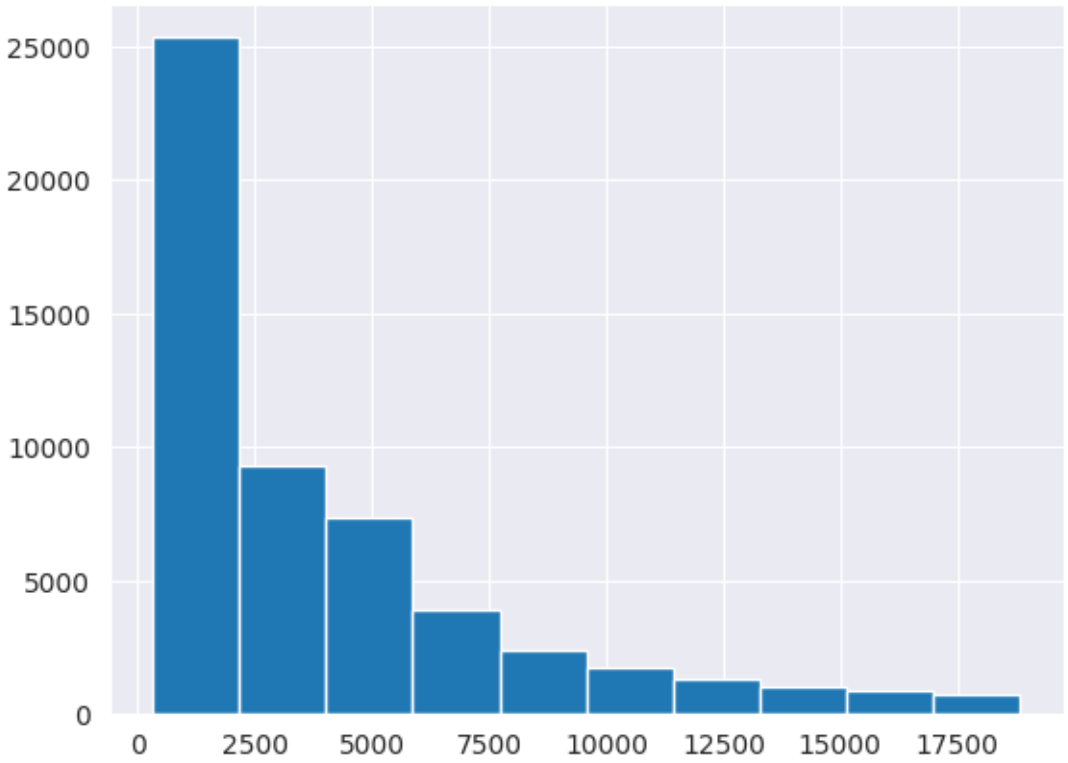
1.2 Visualize diamond prices using a histogram, boxplot and densityplot.

Answer this question:

- Is there trend visible in those plots? If yes, which is it and in whichplots can you see it?

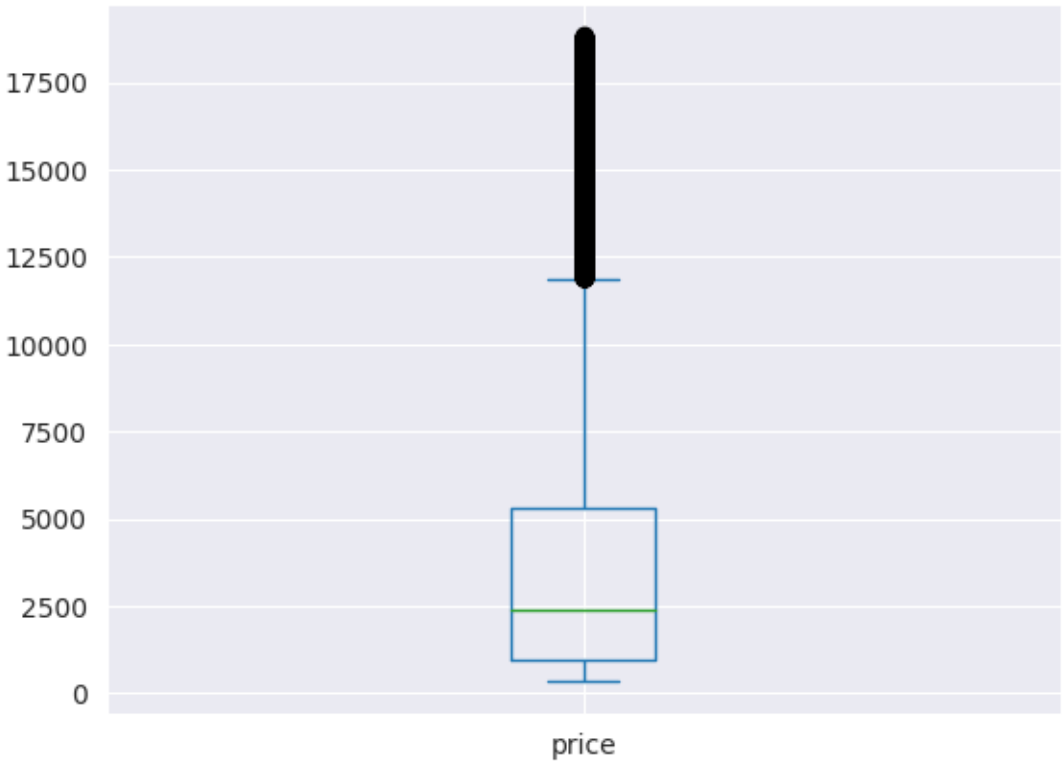
[20]:
```
prices = df["price"]

prices.hist()
```

[20]:
```
<AxesSubplot: >
```



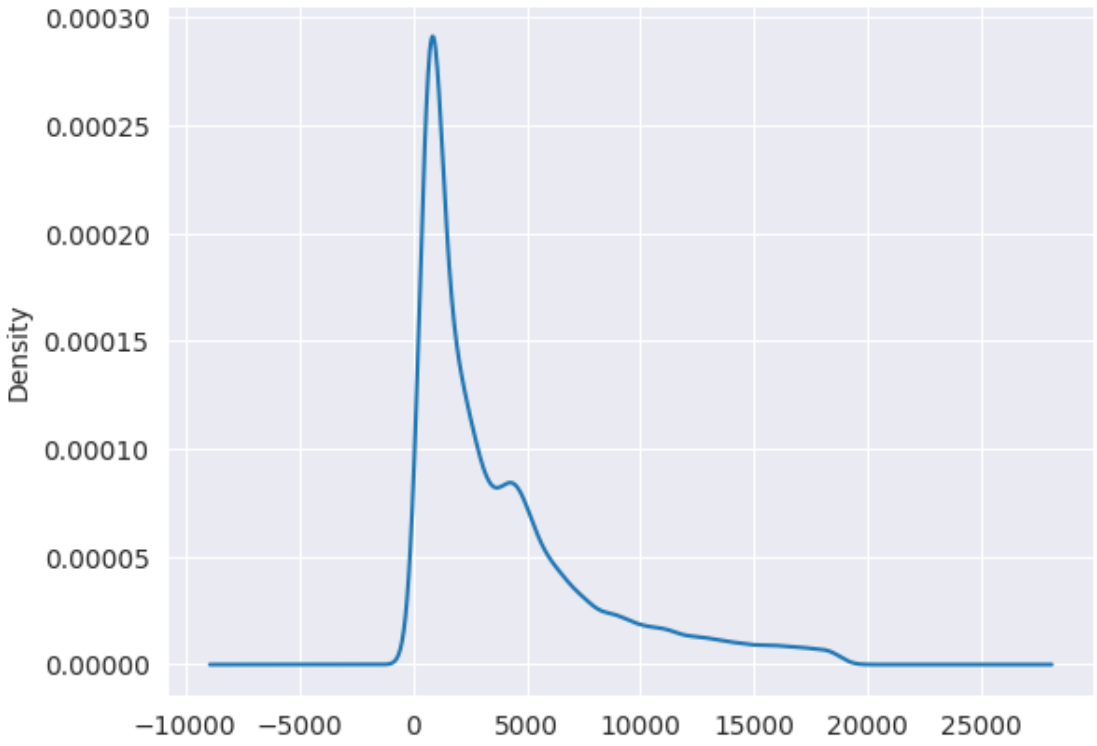
[21]:
```
prices.plot.box()
```

[21]: <AxesSubplot: >



[22]:
```
prices.plot.density()
```

[22]: <AxesSubplot: ylabel='Density'>



The median price is about 2500, 75% are below 5000. (boxplot) The other plots also show where most of the prices are. We can see, the higher the price, the lower the number of diamonds. (density, hist)

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

[23]:
```
print("Diamond prices:")

print("mean: ", round(prices.mean(), 3))
print("median: ", round(prices.median(), 3))

mad = (prices - prices.mean()).abs().mean() # FutureWarning: The 'mad' method␣
 ↪is deprecated and will be removed in a future version. To compute the same␣
 ↪result, you may do `(df - df.mean()).abs().mean()`.
print("mad: ", round(mad, 3))
print("std:", round(prices.std(), 3))

first_quantile = prices.quantile(0.25)
third_quantile = prices.quantile(0.75)
inner_quantile = third_quantile - first_quantile
print("1st quantile: ", round(first_quantile, 3))
```

```
print("3st quantile: ", round(third_quantile, 3))
print("inner quantile rang: ", round(inner_quantile, 3))
```

```
Diamond prices:
mean:  3932.8
median:  2401.0
mad:  3031.603
std: 3989.44
1st quantile:  950.0
3st quantile:  5324.25
inner quantile rang:  4374.25
```

[24]:
```
prices.describe() # or get most of the info by using describe on the price␣
↪column
```

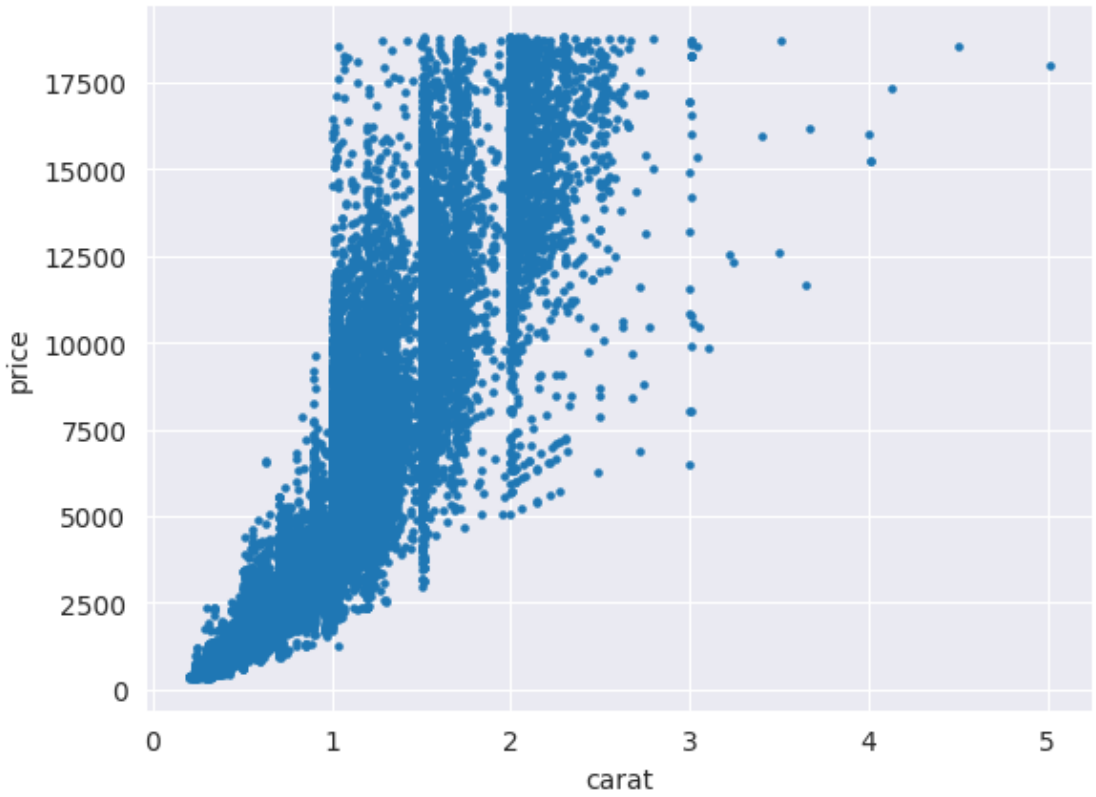
[24]:
```
count    53940.000000
mean      3932.799722
std       3989.439738
min        326.000000
25%        950.000000
50%       2401.000000
75%       5324.250000
max      18823.000000
Name: price, dtype: float64
```

1.4 Plot the diamond price against the carat values as a scatterplot.

Answer this question: * Is there a trend visible in the plot? If yes, which is it?

[25]:
```
df.plot.scatter(y="price", x="carat", marker=".")
```

[25]:
```
<AxesSubplot: xlabel='carat', ylabel='price'>
```



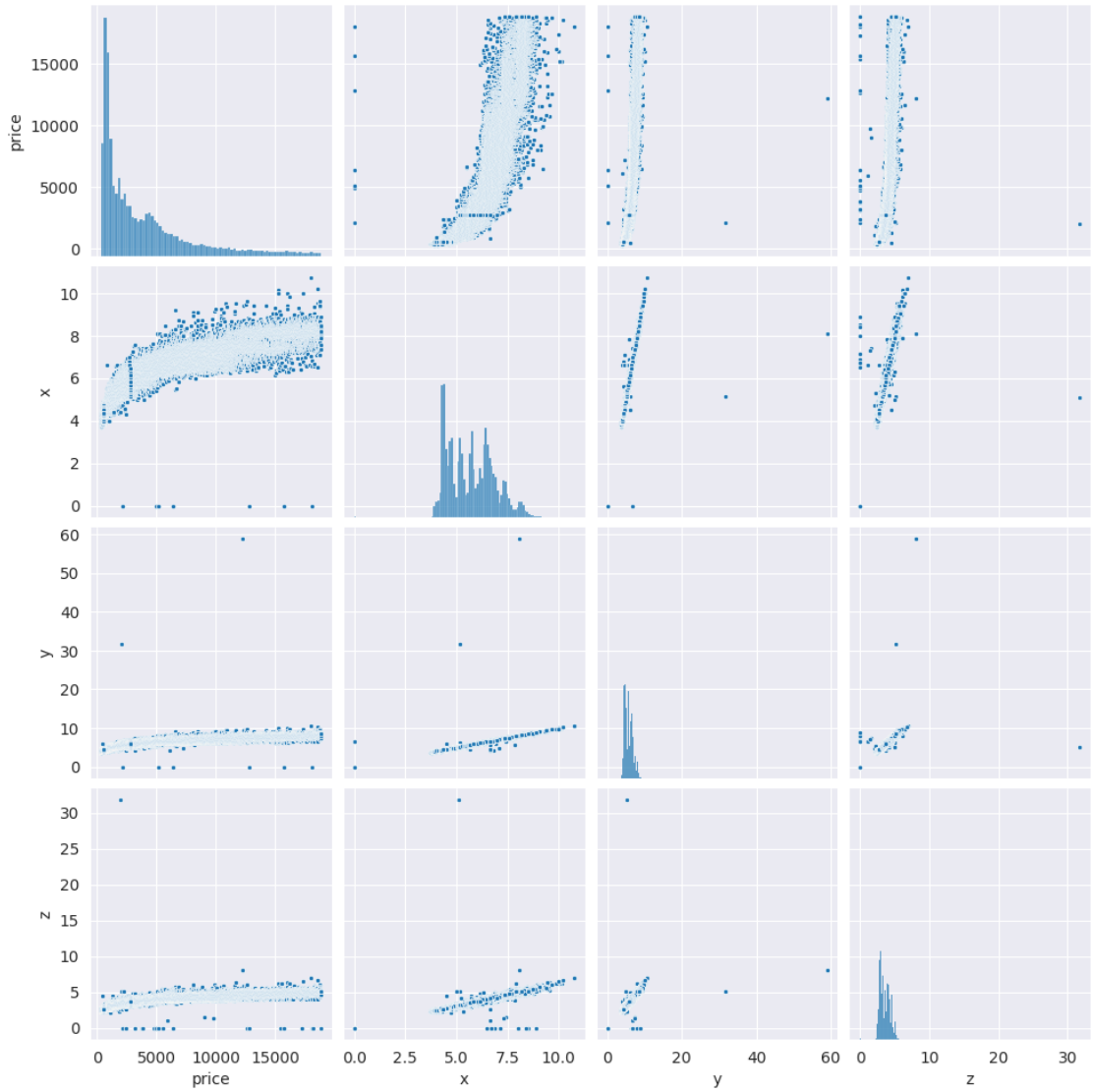
The price increases exponentially with the carat value of the diamond.

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

Answer those questions: * Create pairwise plots for these features. * Is there a trend visible between x, y, and z? If yes, which is it? * Is there a trend visible between the dimensions and the price? If yes,which is it? •

[37]:
```
import seaborn as sns
sns.pairplot(df[["price", "x", "y", "z"]], markers=".")
```

[37]:
```
<seaborn.axisgrid.PairGrid at 0x7ff5981dc7f0>
```



[28]:
```
print("Correlation:")
df[["price", "x", "y", "z"]].corr()
```

```
Correlation:
```

[28]:

	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

Is there a trend between x, y, and z? * yes, usually the bigger one dimension the bigger the other dimensions since all the dimensions have a pretty high linear correlation (> 0.95).

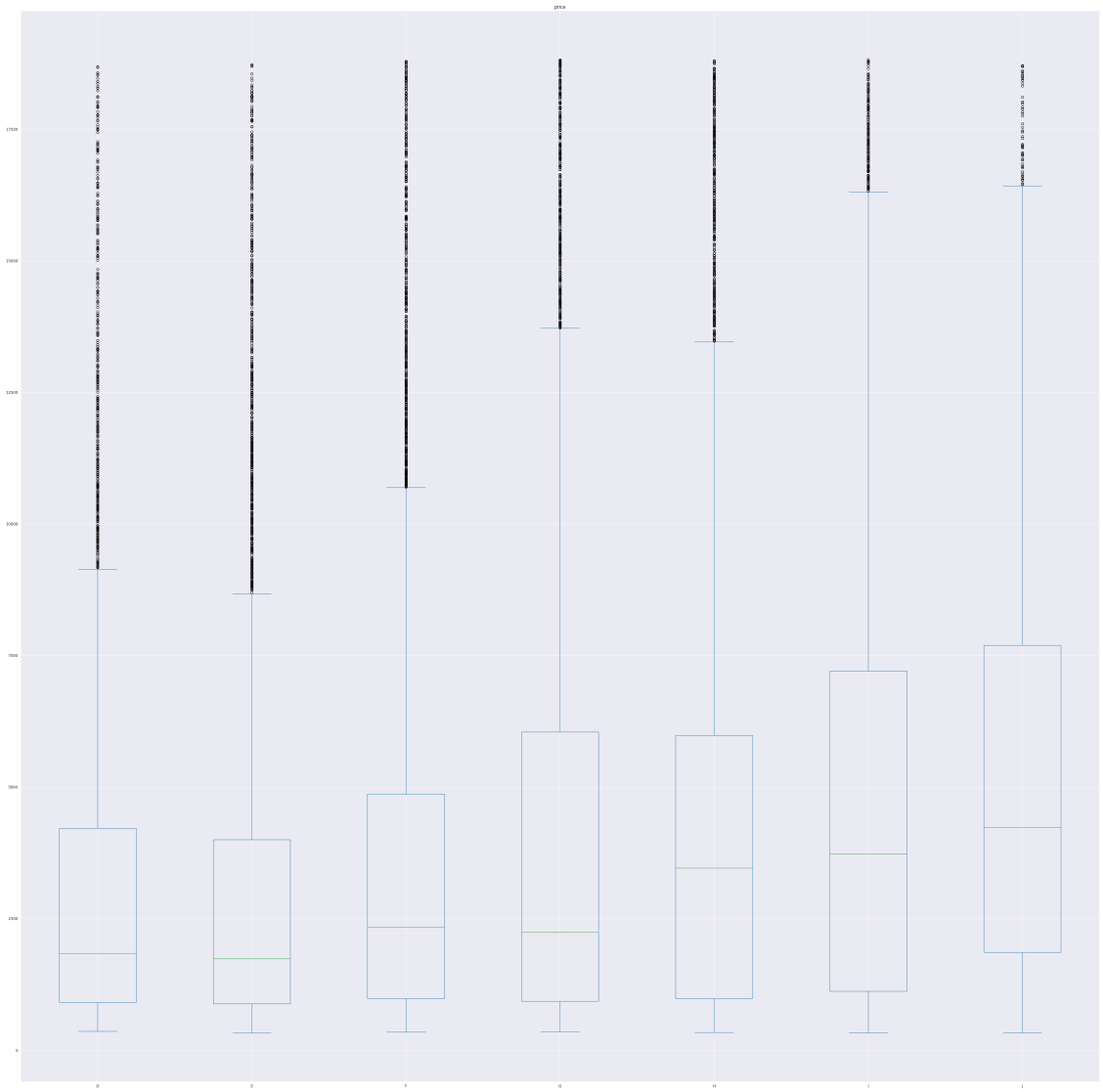
Is there a trend between the dimensions and the price? * yes, the larger the diamond, the greater the price * there is also a pretty strong linear correlation between the dimension and the price. Every dimension correlates between 0.86 - 0.88.

1.6 Analyze diamond prices per diamond color.

- Create boxplots showing diamond price boxes for each diamond color(all boxes should be in one figure).
- Create densityplots showing diamond prices for each diamond color(all densities should be in one figure).
- Answer this question: is there a trend visible? If yes, which one?

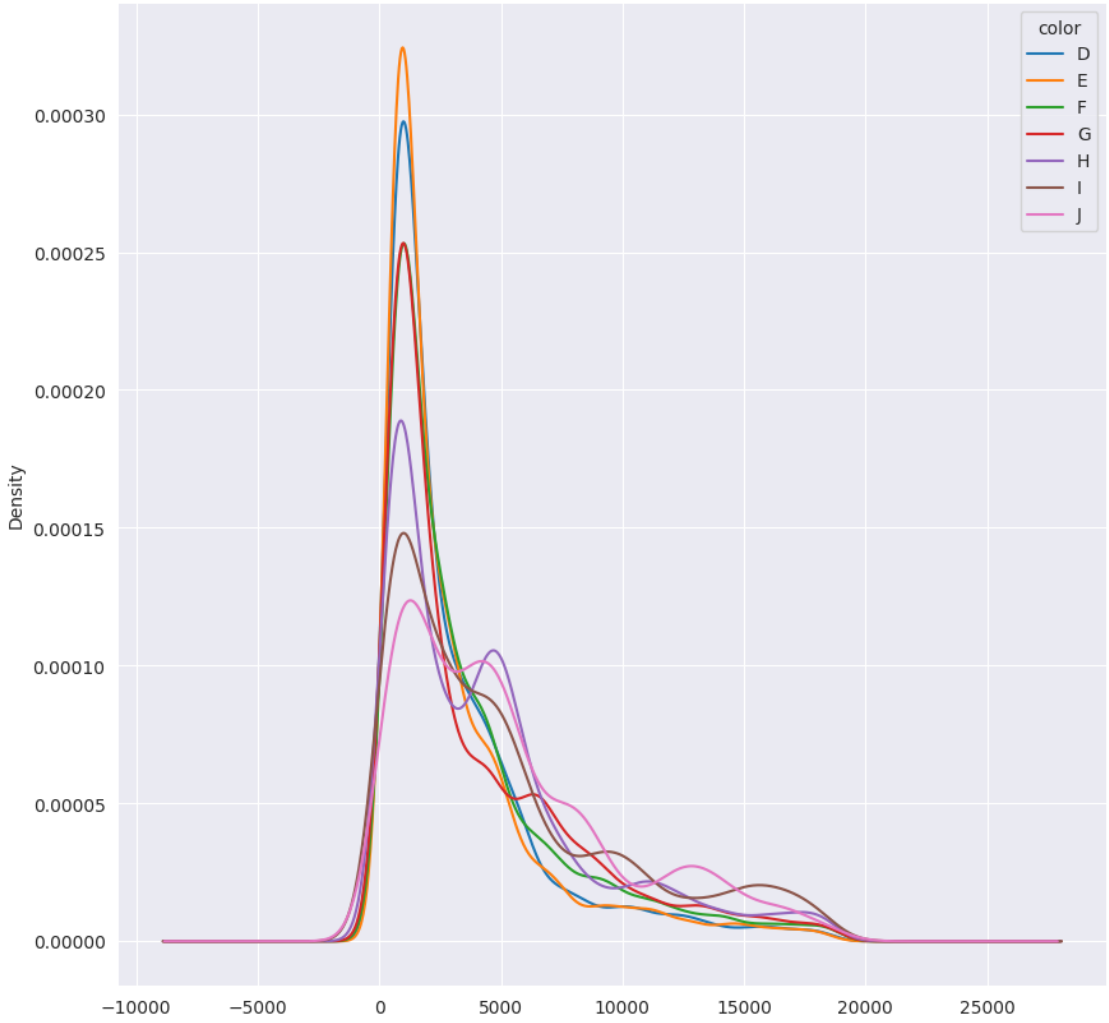
[41]:
```
df[["price", "color"]].plot.box(by="color", figsize=(50,50)) # dont know how to␣
↪make a sensible size for this one so you can still read the axis
```

[41]:
```
price    AxesSubplot(0.125,0.11;0.775x0.77)
dtype: object
```



[30]:
```
df[["price", "color"]].pivot(columns="color", values="price").plot.
↪density(figsize=(10,10))
```

[30]: <AxesSubplot: ylabel='Density'>



1.6.1 Is there a trend visible?

- Most diamonds of different colors are sold for roughly same price. (from density chart)
- The diamonds of color `D`, for example, will have a lower price than diamonds of the color `J`. (from density chart)
- The prices of a color fluctuates differently, the price of color `D` is more stable than that of color `J`. The price of `E` fluctuates the least. (can be seen in boxplots by how far the boxplot stretches)

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

- How many diamonds have a price above 9500?

[31]:
```
df[df["price"] > 9500]["price"].count()
```

[31]: 5734

- How many diamonds have a price above 9500 and have color “D”?

[32]:
```
df[(df["price"] > 9500) & (df["color"] == "D")]["price"].count()
```

[32]:
```
461
```

- What is the mean and std of the price of all color “D” diamonds with cut “Fair”?

[33]:
```
d_and_fair = df[(df["cut"] == "Fair") & (df["color"] == "D")]["price"]

print("mean: ", d_and_fair.mean())
print("std: ", d_and_fair.std())
```

```
mean:  4291.061349693252
std:  3286.1142381749964
```

- What is the median and mad of the price of all color “J” diamonds with cut “Ideal”?

[34]:
```
j_and_ideal = df[(df["cut"] == "Ideal") & (df["color"] == "J")]["price"]

print("median: ", j_and_ideal.median())
print("mad: ", (j_and_ideal - j_and_ideal.mean()).abs().mean())
```

```
median:  4096.0
mad:  3467.5866823780293
```

- 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 ($= 2$) the features in the other dataframe. What is the mean and std of the transformed features in both dataframes?

[47]:
```
import numpy as np

copy_1 = df[["price", "carat"]]
copy_2 = df[["price", "carat"]]

log_10 = np.log10(copy_1)
squared = (copy_2 ** 2)
```

log_10

- mean

[43]:
```
log_10.mean()
```

[43]:
```
price    3.381751
carat   -0.171532
dtype: float64
```

- std

[44]:
```
log_10.std()
```

[44]:
```
price    0.440657
carat    0.253987
dtype: float64
```

squared

- mean

[45]:
```
squared.mean()
```

[45]:
```
price    3.138225e+07
carat    8.613903e-01
dtype: float64
```

- std

[46]:
```
squared.std()
```

[46]:
```
price    6.049189e+07
carat    1.056506e+00
dtype: float64
```

2 Cell Data Segmentation

2.1 Which classes exist? Are they (roughly) balanced?

[64]:
```
from pandas import DataFrame
import pandas as pd

df = pd.read_csv('segmentation_data.csv')

"""
Calculate the class distribution ratio of a feature by using the feature with␣
 ↪the least samples as the basis for the calculation
"""
def ratio(df: DataFrame, feature: str):
    g = df.groupby([feature]).size().to_frame(name = "samples")
    min_samples = g.iloc[:, 0].min()
    g['ratio'] = round(g.iloc[:, 0] / min_samples, 1)
    return g

ratio(df, "Class")
```

[64]:
```
       samples  ratio
Class
```

```
PS        1300    1.8
WS         719    1.0
```

There are two different classes (PS, WS), they are not balanced with PS having 1300 instances and WS 719. (imbalance of 1:1.8 when using WS as the base)

2.2 Which noteworthy trends of features and relations between features aswell as features and Class do you see?

[65]:
```
df.describe()
```

[65]:

	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

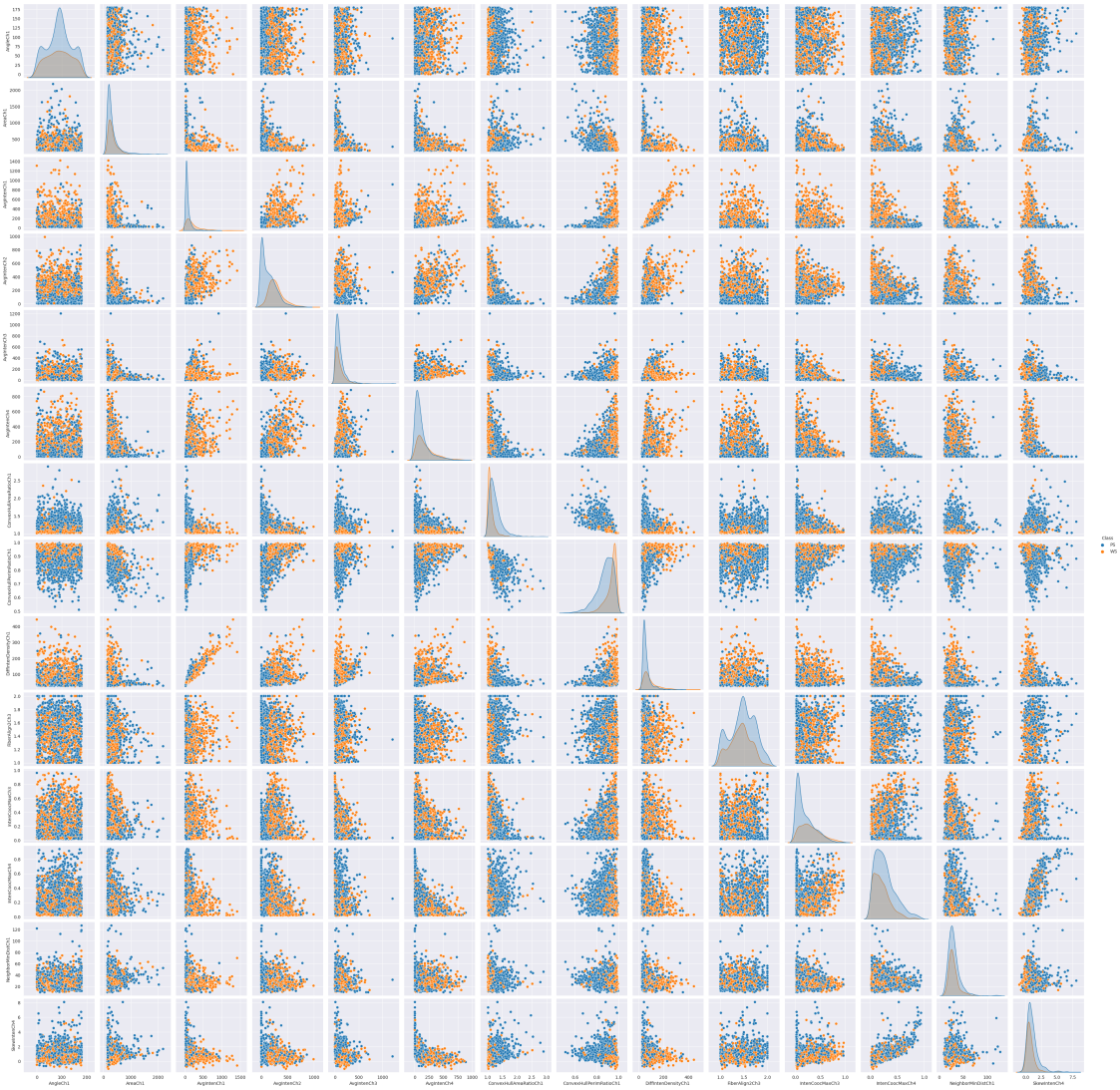
	NeighborMinDistCh1	SkewIntenCh4
count	2019.000000	2019.000000
mean	29.691933	0.932515
std	11.501550	0.885901
min	10.083350	-1.004442
25%	22.547068	0.403460

50%	27.642860	0.728311
75%	34.079173	1.225431
max	126.993700	8.069013

scatterplot of all features:

[66]:
```
import seaborn as sns
sns.pairplot(df, hue="Class")
```

[66]: <seaborn.axisgrid.PairGrid at 0x7f8b391329e0>



2.2.1 correlation matrix:

[67]:
```
df.corr(numeric_only=True)
```

[67]:

	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

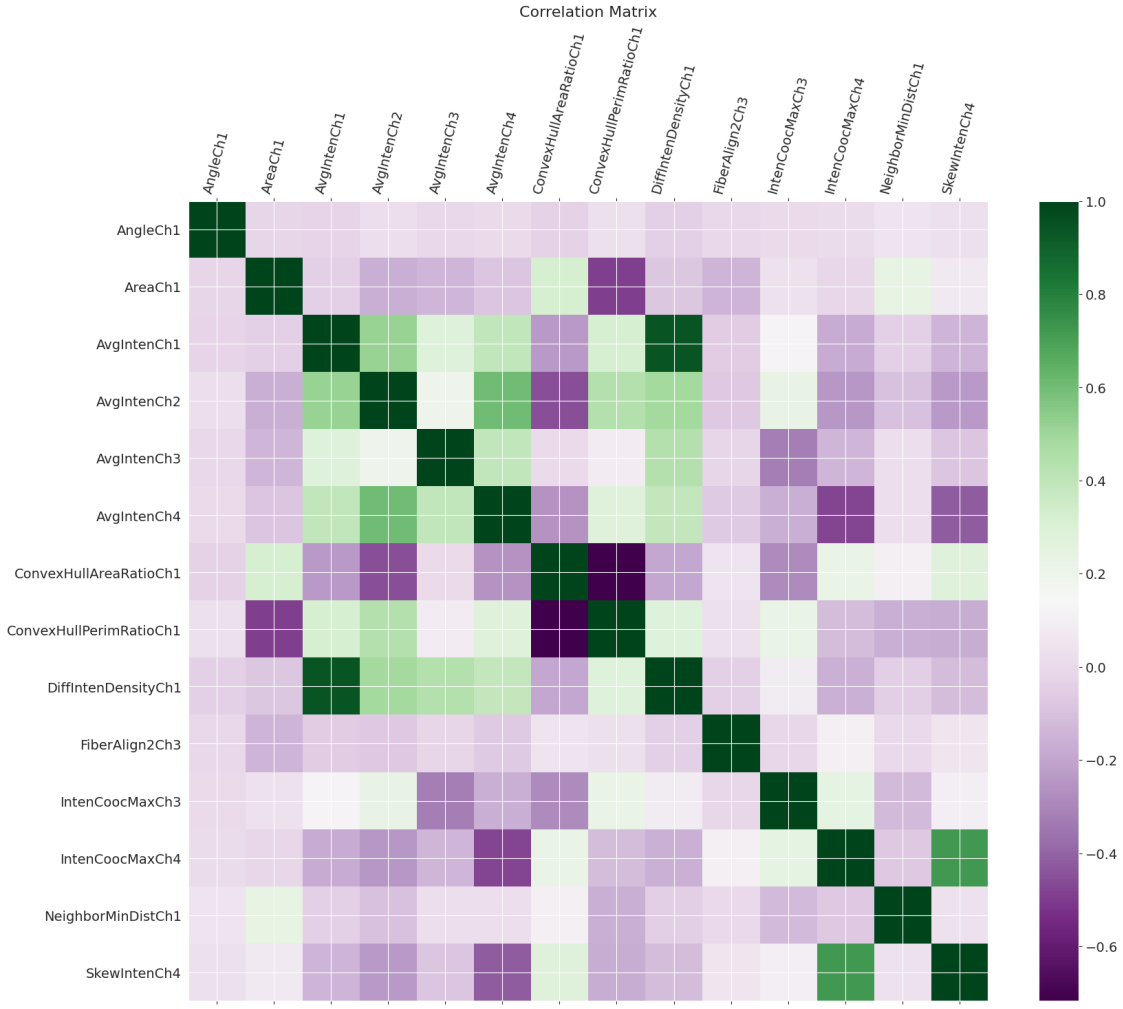
	ConvexHullPerimRatioCh1	DiffIntenDensityCh1
AngleCh1	0.032881	-0.040770
AreaCh1	-0.489944	-0.074990
AvgIntenCh1	0.315972	0.942705
AvgIntenCh2	0.438276	0.485847
AvgIntenCh3	0.089375	0.441698
AvgIntenCh4	0.274304	0.386716
ConvexHullAreaRatioCh1	-0.716921	-0.193268
ConvexHullPerimRatioCh1	1.000000	0.276235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

2.2.2 correlation heatmap:

[68]:
```
# from https://stackoverflow.com/questions/29432629/
 ↪plot-correlation-matrix-using-pandas
import matplotlib.pyplot as plt
f = plt.figure(figsize=(19, 15))
plt.matshow(df.corr(numeric_only=True), fignum=f.number, cmap="PRGn")
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.
 ↪select_dtypes(['number']).columns, fontsize=14, rotation=75)
```

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

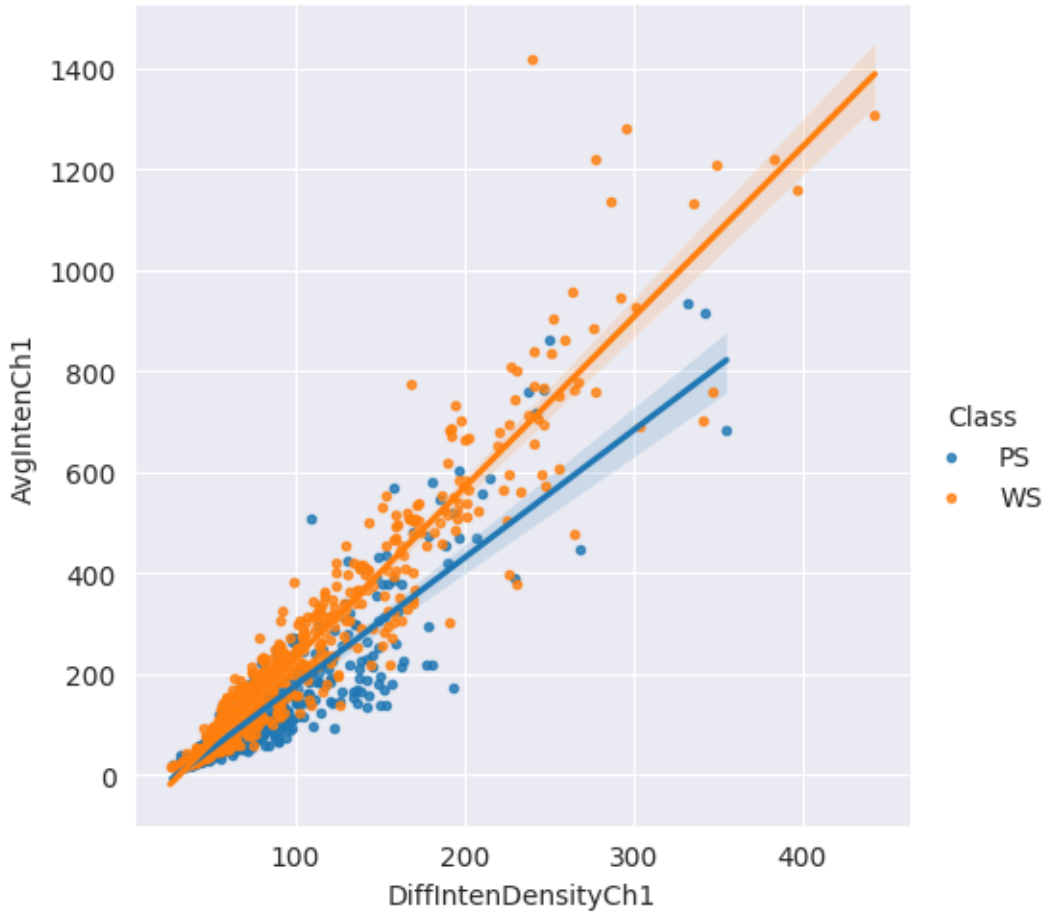


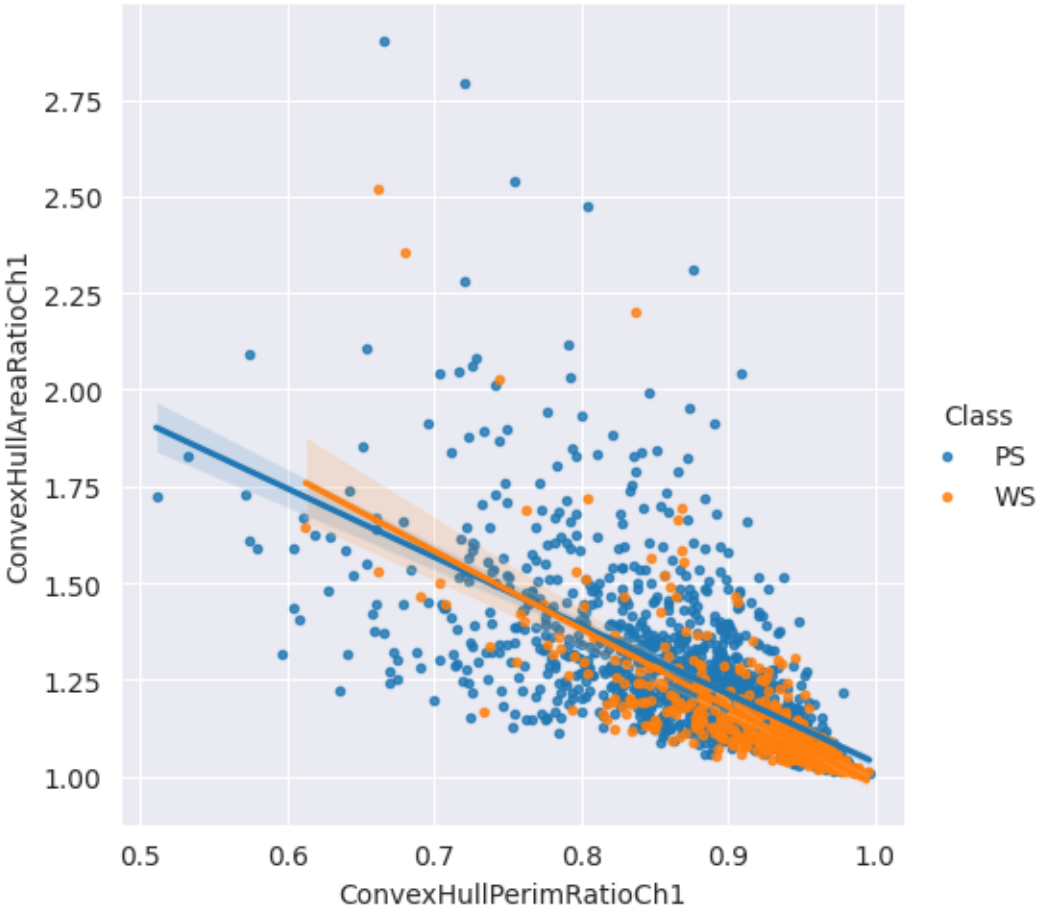
The heatmap is a great tool to visualize the correlation matrix. From it, we can easily see, that the feature `DiffIntenDensityCh1` and the `AvgIntenCh1` have a correlation above 0.9 and that `ConvexHullPerimRatioCh1` and `ConvexHullAreaRatioCh1` have a high negative correlation $<$ -0.7.

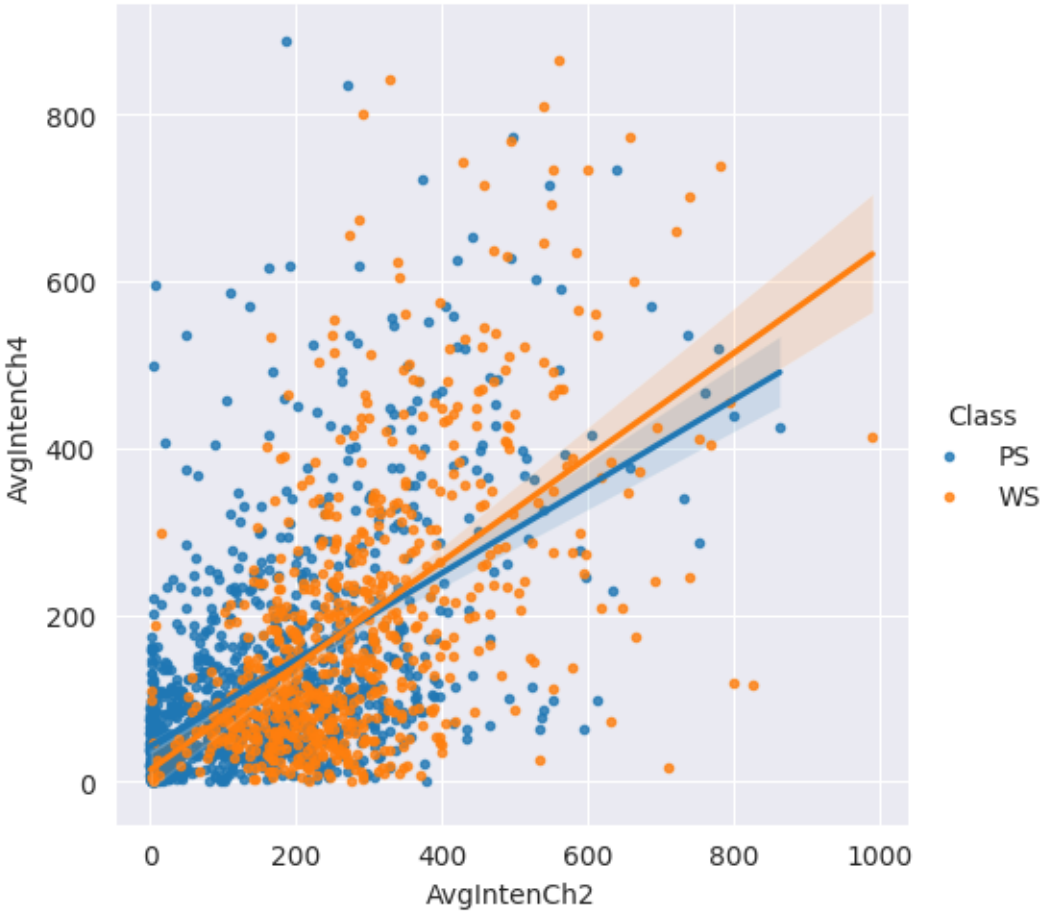
Some other features that have a decent correlation are `SkewIntenCh4` and `IntenCoocMaxCh4` with a correlation of $\sim$ 0.71, as well as `AvgIntenCh2` and `AvgIntenCh4` with a correlation of $\sim$ 0.59.

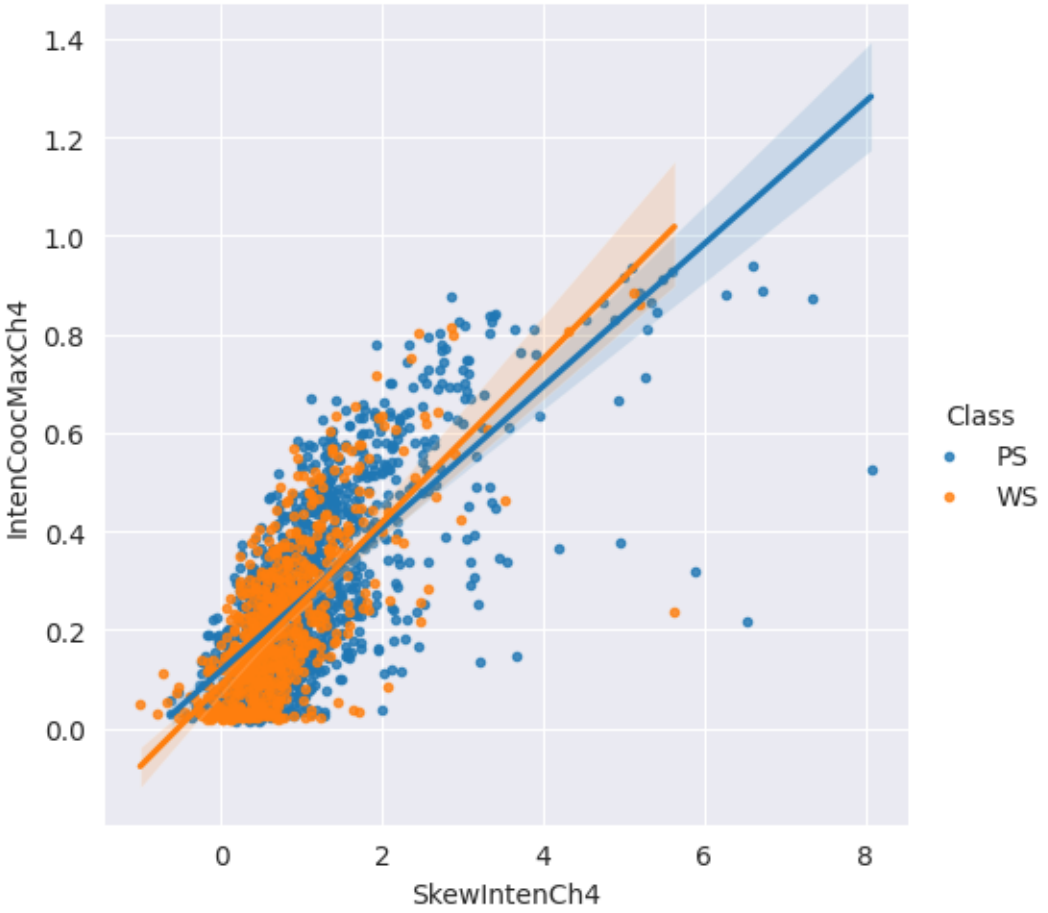
2.2.3 plot the mentioned features:

[69]:
```
def lmplot_by_class(data, x, y):
    sns.lmplot(x=x, y=y, data=df, hue="Class", markers=[".", "."], fit_reg=True)

lmplot_by_class(x="DiffIntenDensityCh1", y="AvgIntenCh1", data=df)
lmplot_by_class(x="ConvexHullPerimRatioCh1", y="ConvexHullAreaRatioCh1",␣
 ↪data=df)
lmplot_by_class(x="AvgIntenCh2", y="AvgIntenCh4", data=df)
lmplot_by_class(x="SkewIntenCh4", y="IntenCoocMaxCh4", data=df)  # did not␣
 ↪manage to place them side by side as this is not supported by lmplot
```




ConvexHullAreaRatioCh1
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.5
0.6
0.7
0.8
0.9
1.0
ConvexHullPerimRatioCh1
Class
PS
WS


800
600
400
200
0
AvgIntenCh4
0
200
400
600
800
1000
AvgIntenCh2
Class
PS
WS

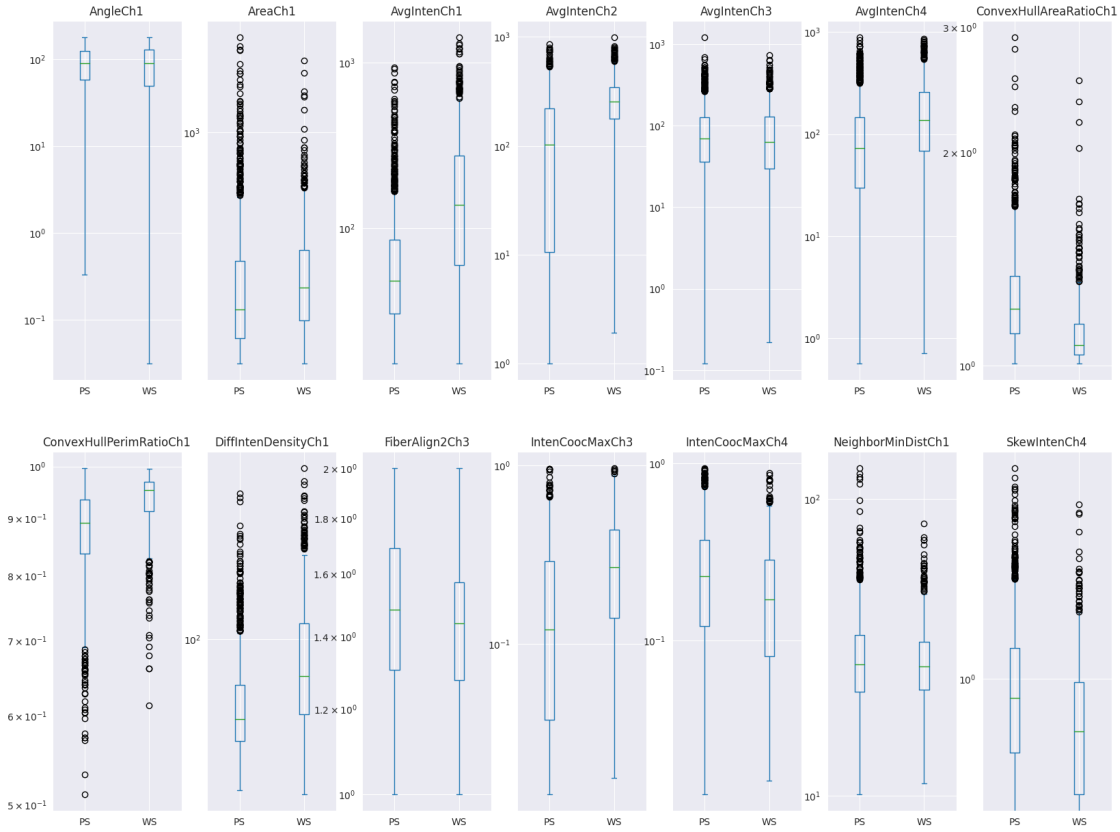


2.2.4 boxplots of the features scaled by log:

[70]:
```
df.plot.box(by='Class', figsize=(20, 15), logy=True, layout=(2, 7))
```

[70]:
```
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)
```

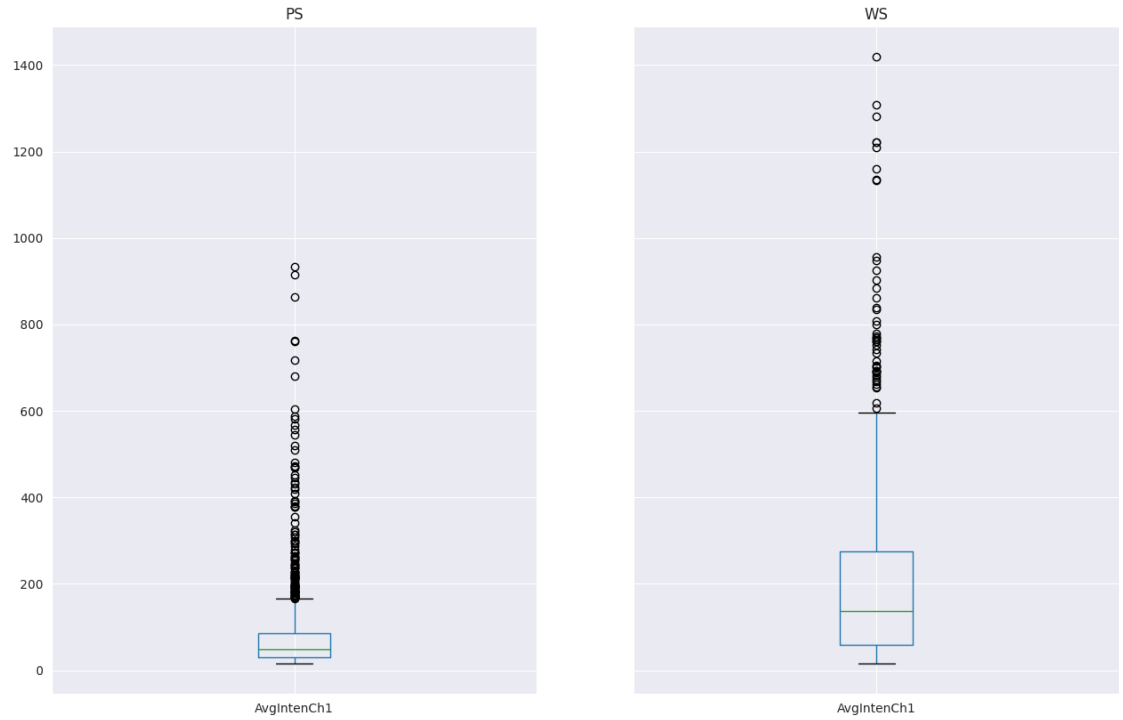
```
dtype: object
```



From these boxplots we take a closer look on the ones that have the least overlap.

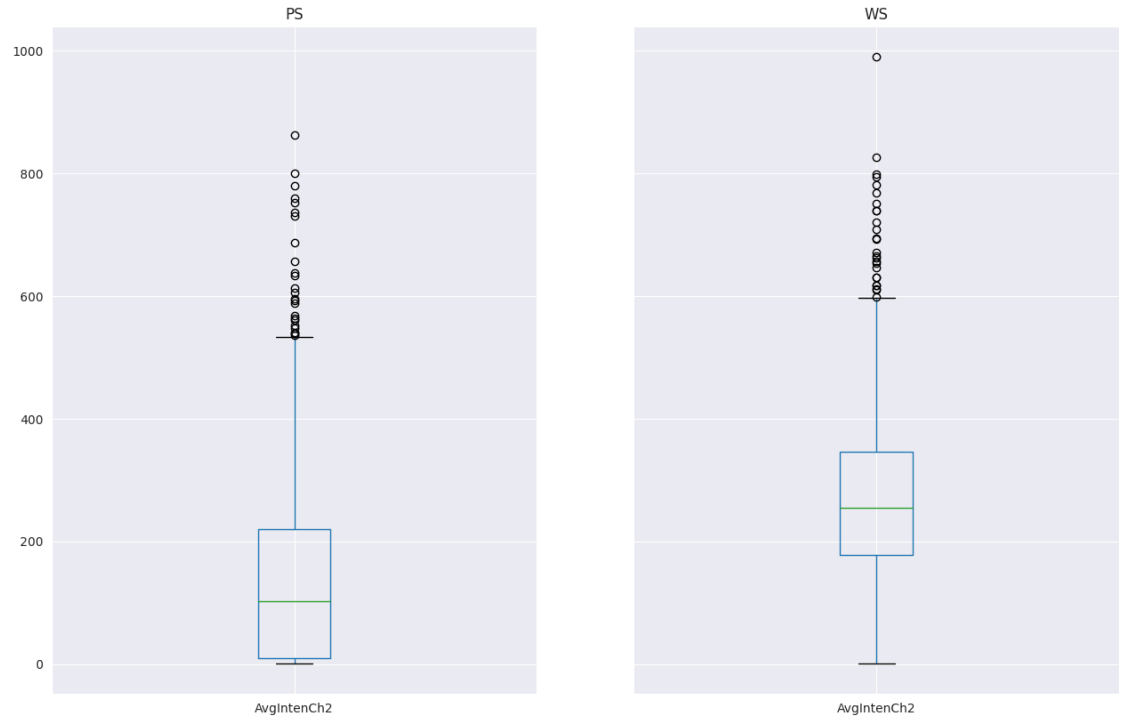
[71]:
```
g = df.groupby(by="Class")
g[['AvgIntenCh1']].boxplot(figsize=(15, 10))
```

[71]:
```
PS          AxesSubplot(0.1,0.15;0.363636x0.75)
WS    AxesSubplot(0.536364,0.15;0.363636x0.75)
dtype: object
```



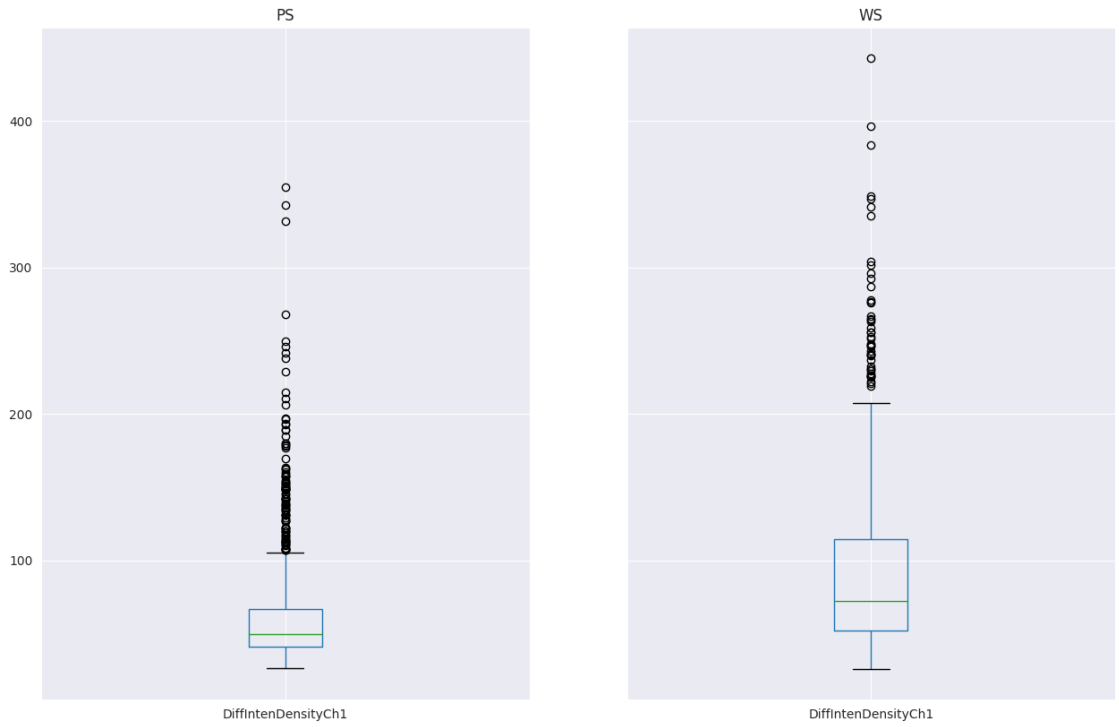
[72]:
```
g[['AvgIntenCh2']].boxplot(figsize=(15, 10))
```

[72]:
```
PS         AxesSubplot(0.1,0.15;0.363636x0.75)
WS    AxesSubplot(0.536364,0.15;0.363636x0.75)
dtype: object
```



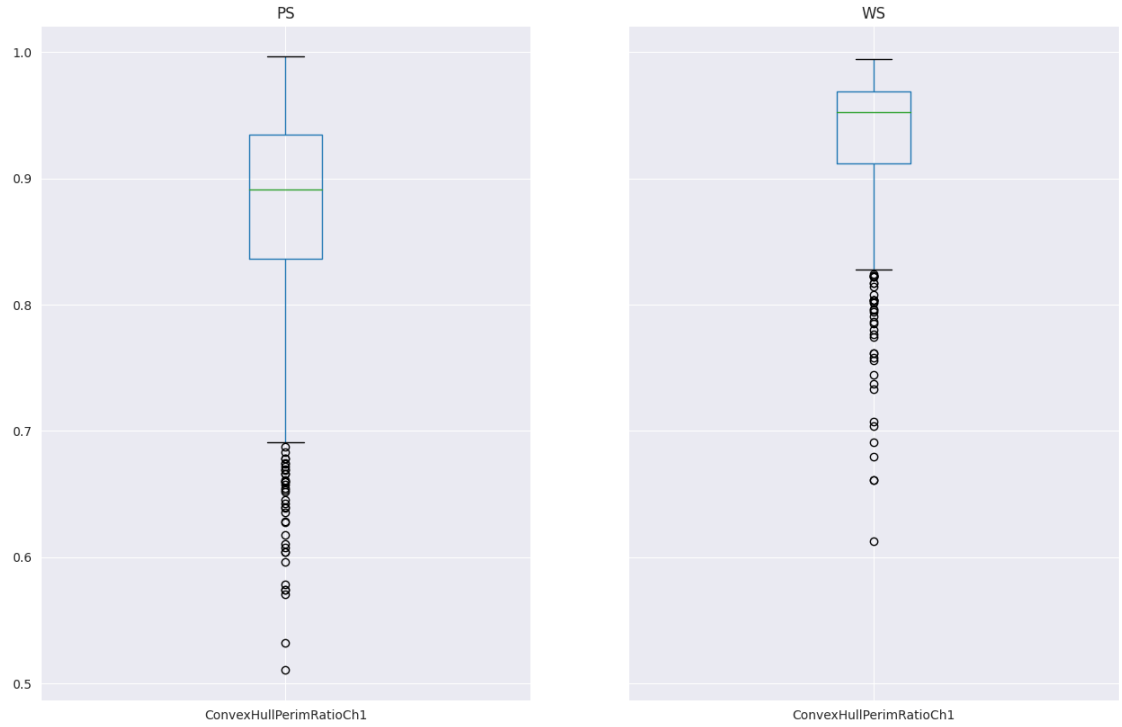
[73]:
```
g[['DiffIntenDensityCh1']].boxplot(figsize=(15, 10))
```

[73]:
```
PS         AxesSubplot(0.1,0.15;0.363636x0.75)
WS    AxesSubplot(0.536364,0.15;0.363636x0.75)
dtype: object
```



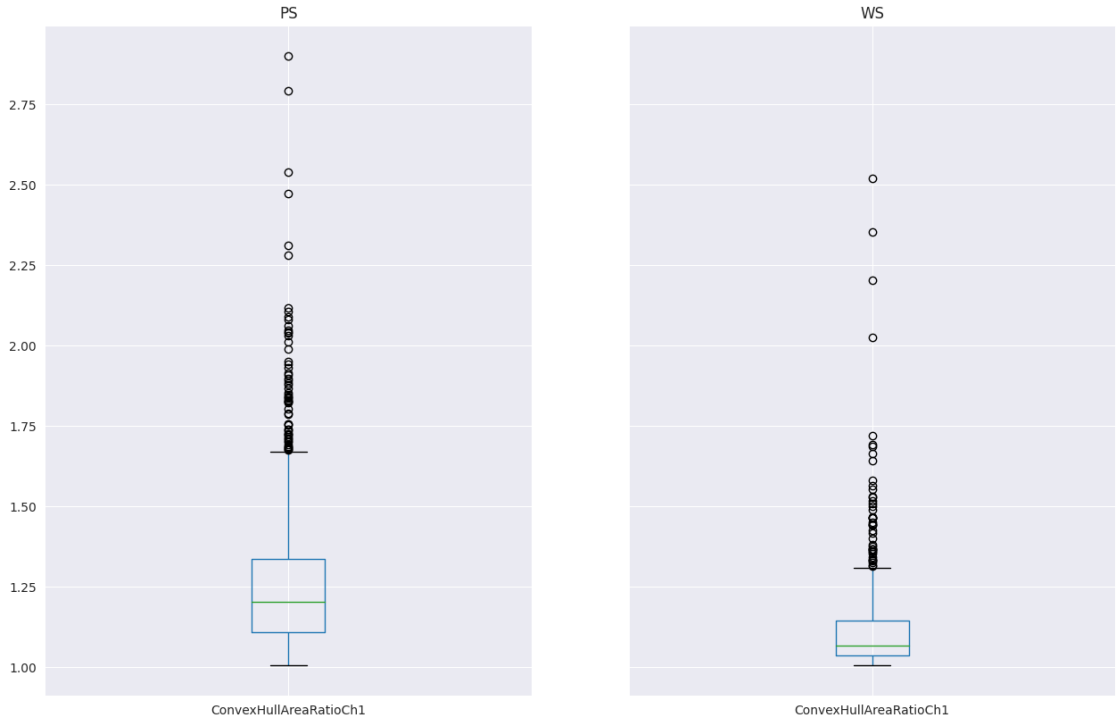
[74]:
```
g[['ConvexHullPerimRatioCh1']].boxplot(figsize=(15, 10))
```

[74]:
```
PS         AxesSubplot(0.1,0.15;0.363636x0.75)
WS    AxesSubplot(0.536364,0.15;0.363636x0.75)
dtype: object
```



[75]:
```
g[['ConvexHullAreaRatioCh1']].boxplot(figsize=(15, 10))
```

[75]:
```
PS         AxesSubplot(0.1,0.15;0.363636x0.75)
WS    AxesSubplot(0.536364,0.15;0.363636x0.75)
dtype: object
```



2.3 If you would need to distinguish the classes with those features, which features would you choose, any why?

The boxplots above show the features that can be used to differentiate between well segmented and poor segmented data entries. - AvgIntenCh1 - AvgIntenCh2 - DiffIntenDensityCh1 - ConvexHullPerimRatioCh1 - ConvexHullAreaRatioCh1

The scatterplots also show that either `ConvexHullAreaRatioCh1` or `ConvexHullPerimRatioCh1` can be used with most other Features to get a somewhat decent separation between the 2 classes.

In retrospect: we were not sure if we should normalize and scale the data or not.