IAF604 Assignment - 01

- Importing Pandas library for reading the file, data manipulation and handling data.
- Importing Matplot library for visualization of the datasets presented.
- Random library to get random list or values to get randomised.

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import random
```

1. Download carpet.csv and hardwood.csv data sets from the following website at the end of chapter three and describe them http://www.uncg.edu/cmp/downloads/.

Downloaded file from http://www.uncg.edu/cmp/downloads/.
Reading File carpet.csv and hardwood.csv which is said to be used for this assignment.
Used relative path to retrive the file.

In [2]:

```
carpet = pd.read_csv("..\CH3\Files\carpet.csv",header=None)
```

Displaying header of carpet.csv data set.

In [3]:

```
carpet.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60	
0	170.39	167.28	143.44	124.67	139.01	125.83	144.33	151.26	175.51	171.31	...	172.96	169.67	157.51	161.06	133.23	124.41	138.44	142
1	169.75	190.96	175.53	138.27	137.47	139.23	133.23	130.25	147.73	163.93	...	139.58	141.58	153.39	141.00	148.43	168.12	169.90	165
2	153.69	153.68	144.02	158.73	178.87	157.04	152.92	147.52	142.87	165.26	...	155.19	170.51	155.37	167.11	146.89	141.01	159.43	169
3	131.69	151.56	151.05	134.00	151.18	175.53	171.34	159.77	151.95	146.10	...	164.25	155.82	157.83	152.43	150.82	146.58	128.85	140
4	162.85	158.88	132.27	138.41	143.98	159.30	177.26	180.58	159.34	164.66	...	132.80	130.96	135.74	167.31	188.21	179.52	146.20	153

5 rows × 64 columns

Here we can understand about the null values present in each column and datatype (float64) of that column and there are 64 columns.

In [4]:

```
carpet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 64 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       1024 non-null   float64
 1   1       1024 non-null   float64
 2   2       1024 non-null   float64
 3   3       1024 non-null   float64
 4   4       1024 non-null   float64
 5   5       1024 non-null   float64
 6   6       1024 non-null   float64
 7   7       1024 non-null   float64
 8   8       1024 non-null   float64
 9   9       1024 non-null   float64
 10  10      1024 non-null   float64
 11  11      1024 non-null   float64
 12  12      1024 non-null   float64
 13  13      1024 non-null   float64
 14  14      1024 non-null   float64
 15  15      1024 non-null   float64
 16  16      1024 non-null   float64
 17  17      1024 non-null   float64
 18  18      1024 non-null   float64
 19  19      1024 non-null   float64
 20  20      1024 non-null   float64
 21  21      1024 non-null   float64
 22  22      1024 non-null   float64
 23  23      1024 non-null   float64
 24  24      1024 non-null   float64
 25  25      1024 non-null   float64
 26  26      1024 non-null   float64
 27  27      1024 non-null   float64
```

```
 28  28      1024 non-null   float64
 29  29      1024 non-null   float64
 30  30      1024 non-null   float64
 31  31      1024 non-null   float64
 32  32      1024 non-null   float64
 33  33      1024 non-null   float64
 34  34      1024 non-null   float64
 35  35      1024 non-null   float64
 36  36      1024 non-null   float64
 37  37      1024 non-null   float64
 38  38      1024 non-null   float64
 39  39      1024 non-null   float64
 40  40      1024 non-null   float64
 41  41      1024 non-null   float64
 42  42      1024 non-null   float64
 43  43      1024 non-null   float64
 44  44      1024 non-null   float64
 45  45      1024 non-null   float64
 46  46      1024 non-null   float64
 47  47      1024 non-null   float64
 48  48      1024 non-null   float64
 49  49      1024 non-null   float64
 50  50      1024 non-null   float64
 51  51      1024 non-null   float64
 52  52      1024 non-null   float64
 53  53      1024 non-null   float64
 54  54      1024 non-null   float64
 55  55      1024 non-null   float64
 56  56      1024 non-null   float64
 57  57      1024 non-null   float64
 58  58      1024 non-null   float64
 59  59      1024 non-null   float64
 60  60      1024 non-null   float64
 61  61      1024 non-null   float64
 62  62      1024 non-null   float64
 63  63      1024 non-null   float64
dtypes: float64(64)
memory usage: 512.1 KB
```

In [5]:

```
hardwood = pd.read_csv("..\CH3\Files\hardwood.csv", header=None)
```

In [6]:

```
hardwood.head()
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60	
0	93.593	89.581	86.892	89.289	87.814	87.369	85.607	85.630	83.339	84.683	...	82.271	77.157	57.394	65.553	68.725	69.740	70.054	69.
1	62.800	68.942	70.733	72.270	74.104	70.765	70.433	73.389	83.640	83.944	...	80.844	85.389	90.223	91.711	93.813	92.941	92.318	91.
2	91.456	95.562	95.546	97.105	95.005	95.161	93.941	93.656	93.530	95.806	...	93.733	96.668	88.511	88.927	87.496	87.760	92.894	90.
3	88.069	85.126	87.511	88.397	91.063	91.295	87.670	91.243	94.734	89.150	...	91.443	93.115	90.032	91.643	91.100	88.701	86.289	85.
4	91.156	89.904	88.336	87.195	86.341	90.781	92.560	93.496	94.155	95.442	...	88.820	93.671	92.162	91.778	95.059	92.023	90.437	94.

5 rows × 64 columns

In [7]:

```
hardwood.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1024 entries, 0 to 1023
Data columns (total 64 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       1024 non-null   float64
 1   1       1024 non-null   float64
 2   2       1024 non-null   float64
 3   3       1024 non-null   float64
 4   4       1024 non-null   float64
 5   5       1024 non-null   float64
 6   6       1024 non-null   float64
 7   7       1024 non-null   float64
 8   8       1024 non-null   float64
 9   9       1024 non-null   float64
 10  10      1024 non-null   float64
 11  11      1024 non-null   float64
 12  12      1024 non-null   float64
 13  13      1024 non-null   float64
 14  14      1024 non-null   float64
 15  15      1024 non-null   float64
 16  16      1024 non-null   float64
 17  17      1024 non-null   float64
```

```
 18  18      1024 non-null   float64
 19  19      1024 non-null   float64
 20  20      1024 non-null   float64
 21  21      1024 non-null   float64
 22  22      1024 non-null   float64
 23  23      1024 non-null   float64
 24  24      1024 non-null   float64
 25  25      1024 non-null   float64
 26  26      1024 non-null   float64
 27  27      1024 non-null   float64
 28  28      1024 non-null   float64
 29  29      1024 non-null   float64
 30  30      1024 non-null   float64
 31  31      1024 non-null   float64
 32  32      1024 non-null   float64
 33  33      1024 non-null   float64
 34  34      1024 non-null   float64
 35  35      1024 non-null   float64
 36  36      1024 non-null   float64
 37  37      1024 non-null   float64
 38  38      1024 non-null   float64
 39  39      1024 non-null   float64
 40  40      1024 non-null   float64
 41  41      1024 non-null   float64
 42  42      1024 non-null   float64
 43  43      1024 non-null   float64
 44  44      1024 non-null   float64
 45  45      1024 non-null   float64
 46  46      1024 non-null   float64
 47  47      1024 non-null   float64
 48  48      1024 non-null   float64
 49  49      1024 non-null   float64
 50  50      1024 non-null   float64
 51  51      1024 non-null   float64
 52  52      1024 non-null   float64
 53  53      1024 non-null   float64
 54  54      1024 non-null   float64
 55  55      1024 non-null   float64
 56  56      1024 non-null   float64
 57  57      1024 non-null   float64
 58  58      1024 non-null   float64
 59  59      1024 non-null   float64
 60  60      1024 non-null   float64
 61  61      1024 non-null   float64
 62  62      1024 non-null   float64
 63  63      1024 non-null   float64
dtypes: float64(64)
memory usage: 512.1 KB
```

2. Extract statistical information (e.g. number of observations, dimension of the data, mean of each feature, etc.) from these datasets. Also present visual representations (e.g. histogram, scatter plot, etc.) of the data. Is the dataset imbalanced, inaccurate or incomplete? Is it a trivial data or possibly a big data? Does it have scalability problem? Are they high dimensional? You need to write programs to read the data and do this.

Carpet data statistic analysis and plotting Histogram and scatterplots.

Here is some of the statistical information about the data set carpet.csv, we can see the total count of rows, mean of individual column, standard deviation, Maximun and minimum values .., under each column.

In [8]:

```
carpet.describe()
```

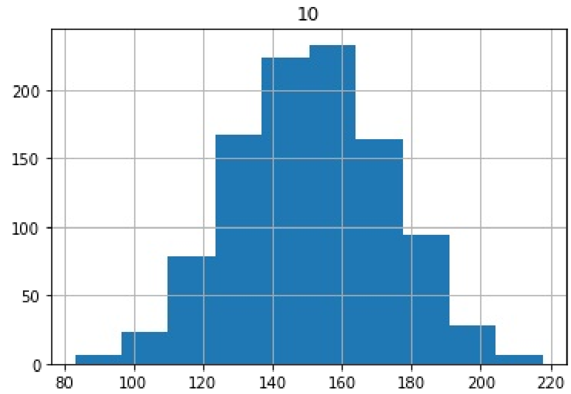
Out[8]:

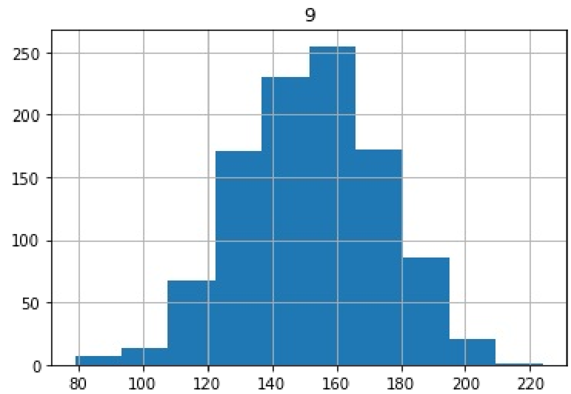
	0	1	2	3	4	5	6	7	8	9	...
count	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	...
mean	151.850416	151.281300	151.283510	151.919732	152.599011	152.829199	152.293690	151.960526	152.123718	152.248084	...
std	22.671128	22.043466	21.642348	21.715601	22.467180	22.336789	22.028949	22.660049	22.858322	22.513211	...
min	85.590000	81.564000	83.886000	81.334000	83.447000	80.529000	75.796000	66.143000	75.157000	78.858000	...
25%	135.980000	135.897500	136.445000	137.265000	137.447500	136.402500	136.672500	136.095000	137.337500	136.700000	...
50%	153.085000	151.255000	152.520000	152.365000	152.685000	153.180000	153.235000	152.100000	152.970000	152.215000	...
75%	167.650000	165.772500	166.545000	166.892500	168.750000	168.285000	168.680000	168.567500	168.370000	167.330000	...
max	210.650000	210.200000	212.930000	211.000000	213.100000	215.900000	218.090000	215.430000	223.880000	224.050000	...

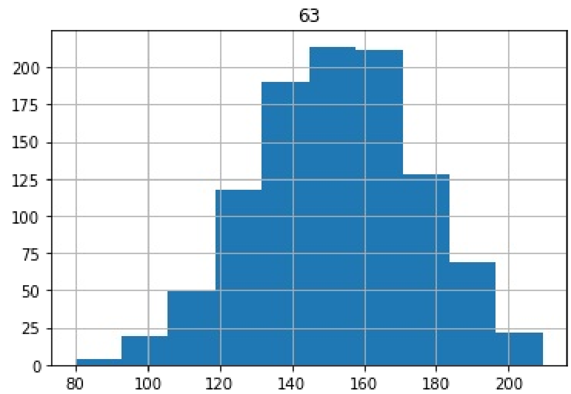
8 rows × 64 columns

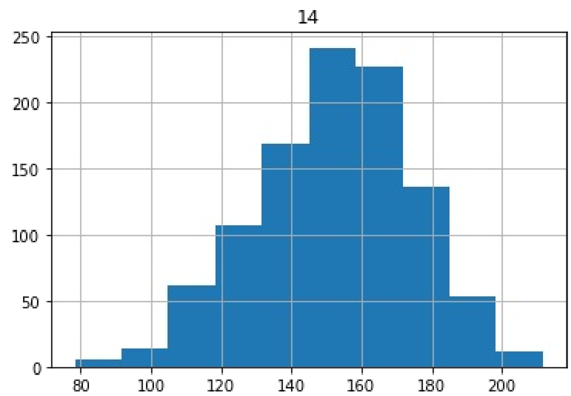
Plotting random features to understand more about the carpet data. There is a scatterplot to show distribution for few features.

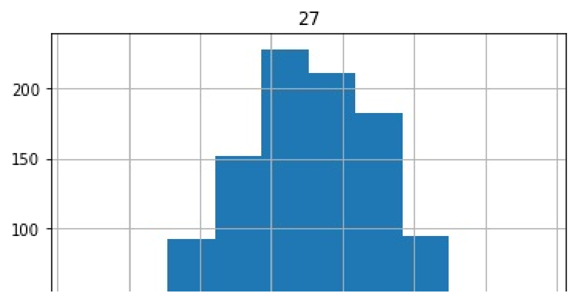
In [9]:
```python
for i in random.sample(range(len(carpet.columns)), 5):
    carpet.hist(column=i)
```

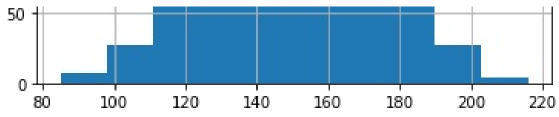






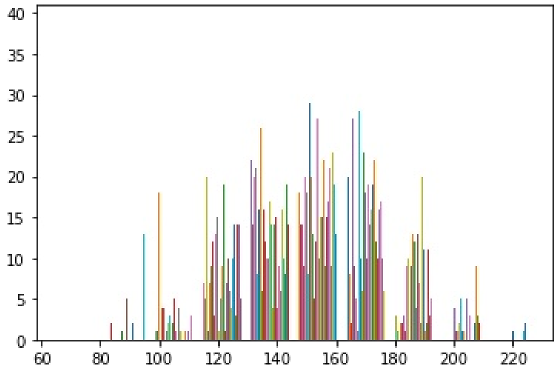






In [10]:

```
plt.hist(carpet)
plt.show()
```

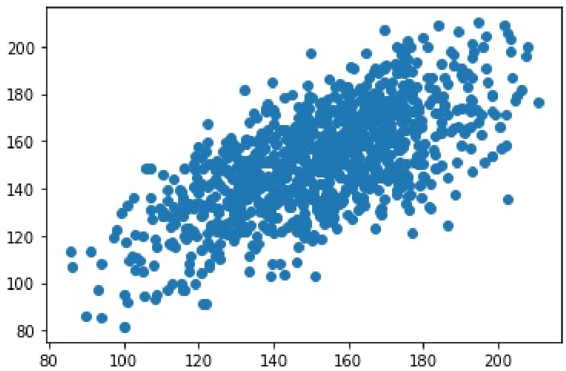


In [11]:

```
plt.scatter(carpet[0],carpet[1])
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0x26080349e80>
```



Hardwood data statistic analysis and plotting Histogram and scatterplots.

In [12]:

```
hardwood.describe()
```

Out[12]:

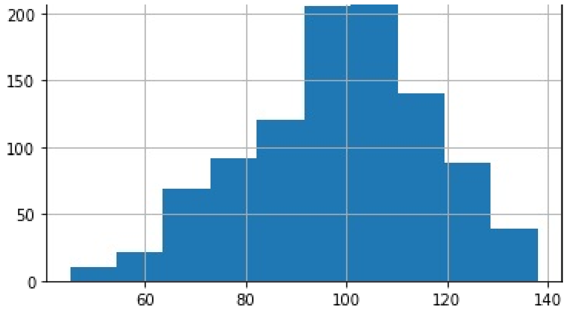
	0	1	2	3	4	5	6	7	8	9	...
count	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	1024.000000	...
mean	99.288047	99.432313	99.560572	99.518933	99.411100	99.339731	99.476555	99.607140	99.285813	99.468638	...
std	17.921041	18.007256	18.023408	18.131449	18.124113	18.074399	18.077122	17.951452	18.229546	18.208710	...
min	47.124000	47.262000	48.485000	49.323000	47.077000	47.365000	47.063000	47.546000	49.302000	48.393000	...
25%	87.321750	86.846000	87.349250	86.916250	87.390000	87.200250	87.633500	87.332000	86.926000	87.410000	...
50%	100.120000	99.810000	100.410000	100.125000	99.511000	99.235000	99.462000	100.355000	99.479000	99.837500	...
75%	110.722500	111.195000	111.452500	111.812500	111.435000	111.795000	112.065000	111.777500	110.952500	111.267500	...
max	139.610000	139.600000	139.900000	141.780000	142.470000	143.340000	137.640000	138.830000	138.020000	141.270000	...

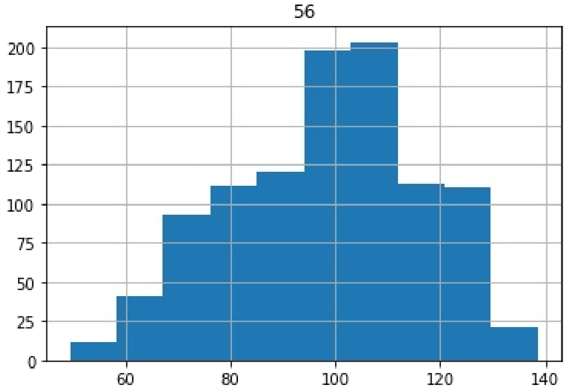
8 rows × 64 columns

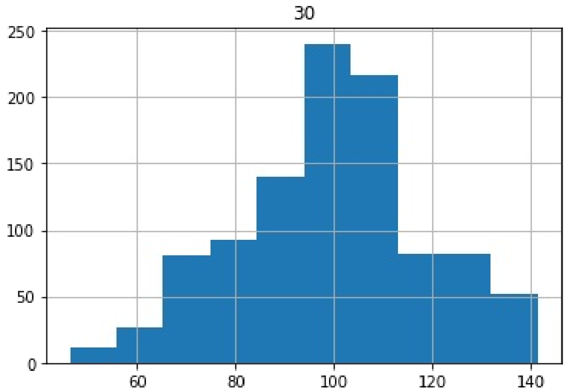
In [13]:

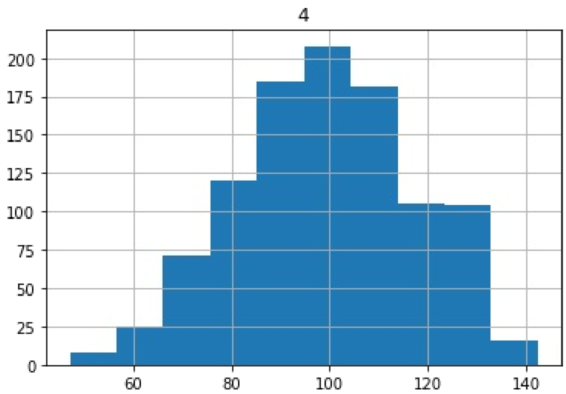
```
for i in random.sample(range(len(hardwood.columns)), 5):
    hardwood.hist(column=i)
```

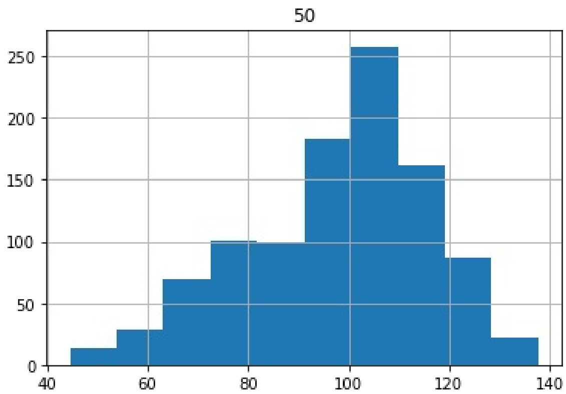
35





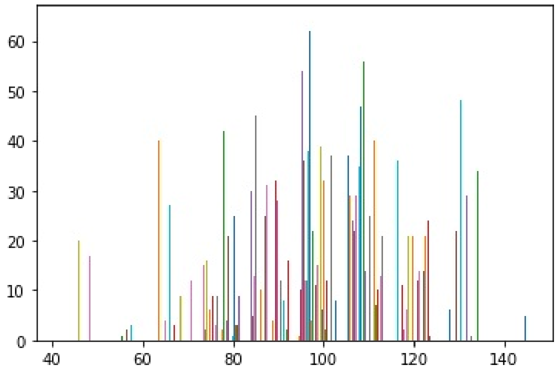






In [14]:

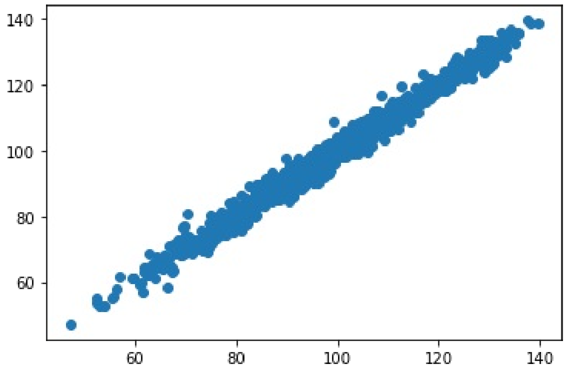
```
plt.hist(hardwood)
plt.show()
```



In [15]:

```
plt.scatter(hardwood[0],hardwood[1])
```

Out[15]: <matplotlib.collections.PathCollection at 0x26084b3e8d0>



Observations made:

carpet dataset:

- It is a big data.
- It is imbalanced data, as features has distributed or varied values.
- Not High dimensional, as dataset in which the number of features p is less than the number of observations N.
- It is not incomplete data as there are no null values.
- If observations are Incorrect it is inaccurate, the dataset is accurate, also standard deviations of each column are comparatively similar.
- No, scalability problem as there are no new additions to the features. when there is unstable growth in the features which can only possible in the high dimensional data then exists scalabilty problem. Since it is a low dimensional data, there is no scalability problem and also no features to add.

hardwood dataset:

- It is a big data.
- It is imbalanced data, as features has distributed or varied values.
- Not High dimensional, as dataset in which the number of features p is less than the number of observations N.
- It is not incomplete data as there are no null values.
- If observations are Incorrect it is inaccurate, the dataset is accurate, also standard deviations of each column are comparatively similar.
- No, scalability problem as there are no new additions to the features. when there is unstable growth in the features which can only possible in the high dimensional data then exists scalabilty problem. Since it is a low dimensional data, there is no scalability problem and also no features to add.

  If both files are merged still no issue because, same features for both datasets.

3. Merge carpet.csv and hardwood.csv and create a new csv file called carwood.csv in which you insert a new column with label 0 for carpet observations and label 1 for hardwood observations. Now shuffle the observations randomly and create a new file called randcarwood.csv. Then divide this file into 80:20 and name the files with Trainrandcarwood80.csv and Testrandcarwood20.csv respectively. You must write a program to do these processes using a programming language of your choice. You can use Python. Include first and last three observations of each file (instead of all data, too long) in the text so that we know what the data samples look like in the files. Include

code/commands and results of showing how many records in each file.

Here created/added label-0 for carpet and label-1 for hardwood. Later combined the datasets and shuffled.

- writing mixed data to Trainrandcarwood.csv file and saving to ch3/files folder.
- displaying Trainrandcarwood80.csv head and tail as train dataframe.
- displaying Trainrandcarwood20.csv head and tail as test dataframe.

In [16]:

```
carpet['label']=0
```

In [17]:

```
hardwood['label']=1
```

In [18]:

```
carwood=carpet.append(hardwood,ignore_index=True)
```

In [19]:

```
r_carwood = carwood.sample(frac=1)
```

In [20]:

```
Trainrandcarwood80 = r_carwood.iloc[:round(len(r_carwood)*0.8),:]
Trainrandcarwood80.to_csv('..\CH3\Files\Trainrandcarwood80.csv')
```

In [21]:

```
Trainrandcarwood20 = r_carwood.iloc[round(len(r_carwood)*0.8):,:]
Trainrandcarwood20.to_csv('..\CH3\Files\Trainrandcarwood20.csv')
```

Displaying First three records of combined data of train dataframe.

In [22]:

```
train = pd.read_csv('..\CH3\Files\Trainrandcarwood80.csv')
train.head(3)
```

Out[22]:

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	55	56	57	58	59	
0	1305	93.825	92.443	92.16	91.013	90.99	89.481	90.018	89.747	90.046	...	98.719	96.944	95.915	94.254	97.385	9
1	1102	121.340	120.540	123.55	122.800	123.23	124.050	124.440	125.540	114.030	...	119.620	122.370	121.290	122.750	123.800	11
2	1833	120.330	118.930	117.89	119.650	113.59	117.410	118.830	118.720	113.320	...	104.150	98.760	97.295	101.080	103.750	10

3 rows × 66 columns

Displaying last three records of combined data of test dataframe.

In [23]:

```
train.tail(3)
```

Out[23]:

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	55	56	57	58	59
1635	1081	91.732	90.746	89.813	90.90	92.078	95.617	93.595	93.518	92.506	...	89.24	93.512	94.025	94.807	91.998
1636	1130	113.200	112.330	114.070	113.72	114.220	115.150	115.570	116.910	105.010	...	114.21	109.250	111.010	108.930	111.280
1637	765	156.880	136.070	135.400	143.73	121.400	119.100	139.650	156.790	185.990	...	126.82	106.230	114.040	142.020	146.850

3 rows × 66 columns

Displaying last frist records of combined data of train dataframe.

In [24]:

```
test = pd.read_csv('..\CH3\Files\Trainrandcarwood20.csv')
test.head(3)
```

Out[24]:

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	55	56	57	58	59	60	61
0	670	195.76	199.64	179.10	145.30	119.63	144.67	166.04	181.52	188.57	...	123.07	131.25	110.01	133.37	147.53	145.35	156.31
1	873	162.60	136.45	155.03	178.39	180.37	164.63	134.05	156.66	170.75	...	202.48	115.29	122.51	149.90	182.74	160.67	134.62
2	878	156.54	154.10	156.31	156.30	148.55	159.10	175.17	195.88	170.19	...	189.96	168.31	156.17	160.45	146.09	120.53	135.04

3 rows × 66 columns

Displaying last three records of combined data of train dataframe.

In [25]:
```
test.tail(3)
```

Out[25]:

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	55	56	57	58	59	60	6
407	283	146.64	134.96	130.16	126.79	122.14	128.01	139.19	138.09	131.40	...	145.58	154.40	185.49	190.92	137.69	116.60	130.2
408	943	155.35	163.74	192.18	201.65	165.42	165.24	165.13	146.75	168.42	...	143.78	169.98	185.04	186.84	160.26	149.51	148.5
409	136	155.84	147.72	162.07	165.83	158.35	169.60	180.84	184.56	131.08	...	172.22	171.15	158.71	157.68	156.81	149.67	156.3

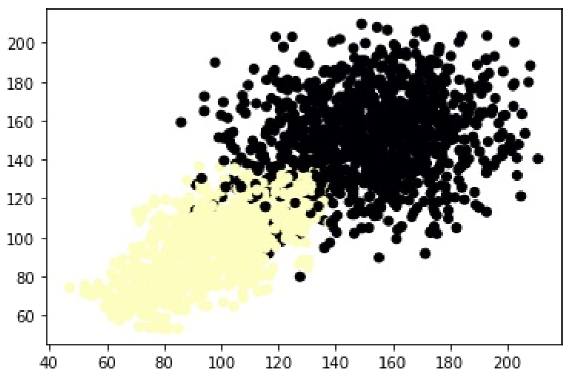
3 rows × 66 columns

Plotting first and last feature with respect to 'label' column to show carpet and hardwood data distributions.

In [26]:
```
plt.scatter(r_carwood[0], r_carwood[63], c = r_carwood['label'], cmap = 'magma')
```

Out[26]:

```
<matplotlib.collections.PathCollection at 0x26085064198>
```



In [27]:
```
r_carwood.describe()
```

Out[27]:

	0	1	2	3	4	5	6	7	8	9	...
count	2048.000000	2048.000000	2048.000000	2048.000000	2048.000000	2048.000000	2048.000000	2048.000000	2048.000000	2048.000000	...
mean	125.569231	125.356807	125.422041	125.719333	126.005055	126.084465	125.885123	125.783833	125.704766	125.858361	...
std	33.292731	32.822212	32.643005	32.966031	33.526247	33.589233	33.220224	33.214697	33.548514	33.402901	...
min	47.124000	47.262000	48.485000	49.323000	47.077000	47.365000	47.063000	47.546000	49.302000	48.393000	...
25%	99.490000	99.095500	100.217500	99.784750	99.094250	98.990500	99.144750	99.745750	98.876750	99.449500	...
50%	123.430000	124.160000	123.970000	124.460000	123.735000	124.275000	124.465000	124.390000	123.425000	124.595000	...
75%	153.017500	151.252500	152.505000	152.347500	152.677500	153.165000	153.202500	152.085000	152.965000	152.192500	...
max	210.650000	210.200000	212.930000	211.000000	213.100000	215.900000	218.090000	215.430000	223.880000	224.050000	...

8 rows × 65 columns