FUNDAMENTALS OF MACHINE LEARNING IN DATA SCIENCE

CSIS 3290

WORK MORE ON DATASETS AND STATISTICS IN SCIPY.STATS

FATEMEH AHMADI

SciPy Package

- **SciPy** is a free and open-source python library and stands for **scientific Python** is a package contains many sub packages for helping with complex scientific calculations.
- **SciPy** package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or MATLAB's.
- Easy to use and understand as well as fast computational power.
- SciPy is built in top of the NumPy. SciPy module in Python is a fully-featured version of Linear Algebra while NumPy contains only a few features.
- Most new Data Science features are available in **SciPy** rather than NumPy.

https://www.guru99.com/scipy-tutorial.html

3

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sb
```

```
In [3]: data1=pd.read_csv('F:/00-Douglas College/1- Semester 1/3- Machine Learning in Data Science(3290)/Slides/Weather1.csv')
```

```
In [4]: print(data1)
```

```
    Summary  Temperature (C)  Apparent Temperature (C)  Humidity  \
0         1         9.472222                  7.388889      0.89
1         1         9.355556                  7.227778      0.86
2         2         9.377778                  9.377778      0.89
3         1         8.288889                  5.944444      0.83
4         2         8.755556                  6.977778      0.83
..      ...              ...                       ...       ...
94        2         7.827778                  5.405556      0.72
95        3         7.855556                  6.122222      0.72
96        2         7.316667                  6.211111      0.75
97        3         7.244444                  6.005556      0.75
98        1         5.438889                  5.438889      0.88

    Wind Speed (km/h)  Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
0             14.1197                     251          15.8263           0
1             14.2646                     259          15.8263           0
2              3.9284                     204          14.9569           0
3             14.1036                     269          15.8263           0
4             11.0446                     259          15.8263           0
..                ...                     ...              ...         ...
94            13.8943                      28          15.8263           0
95             9.8049                      11          15.0052           0
96             6.6654                     326          15.8746           0
97             7.1162                     309          15.8746           0
98             3.7191                     193           9.9820           0

    Pressure (millibars)
0                1015.13
1                1015.63
2                1015.94
```

Accessing Features


```
In [17]: np.mean(data1.Humidity)
Out[17]: 0.7425252525252525

In [18]: np.std(data1.Humidity)
Out[18]: 0.15873453752323097

In [ ]:
```

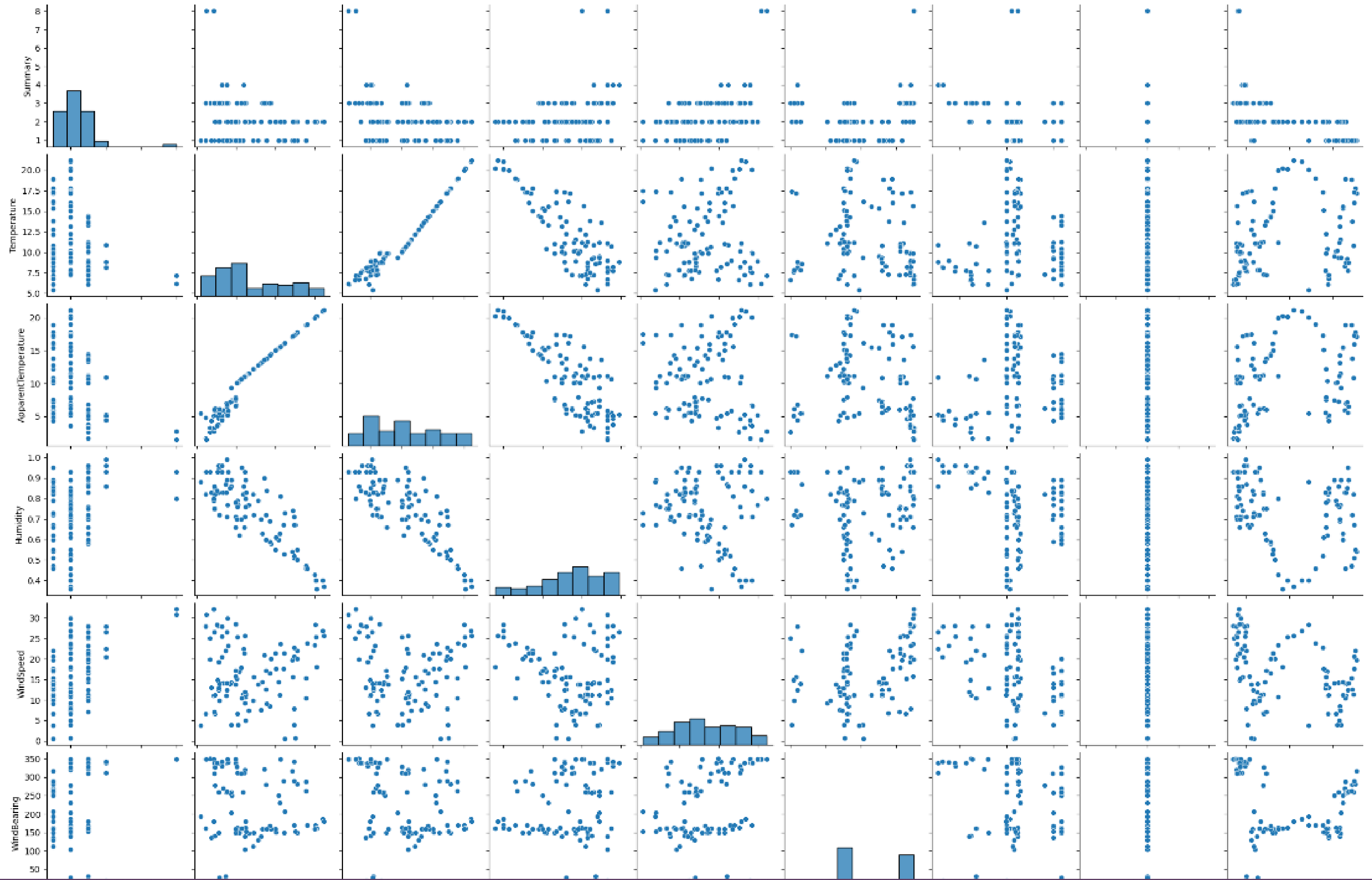
```
In [9]: print(data1.Humidity)

0     0.89
1     0.86
2     0.89
3     0.83
4     0.83
      ...
94    0.72
95    0.72
96    0.75
97    0.75
98    0.88
Name: Humidity, Length: 99, dtype: float64

In [10]: print(data1.Humidity.value_counts())

Humidity
0.93    10
0.83     5
0.71     5
0.89     4
0.82     4
0.72     4
0.96     4
0.77     3
0.40     3
0.86     3
0.66     3
0.79     3
0.70     3
0.67     3
0.60     2
0.55     2
0.85     2
0.84     2
0.95     2
0.75     2
0.63     2
0.80     2
```

Pairplot

Gives us a pretty good view of some existing patterns

5

In [25]:
```
sb.pairplot(data1)
plt.show()
```



Correlation with Pearson

```
In [27]: from scipy.stats import pearsonr

In [28]: cor1=pearsonr(data1.Temperature,data1.Humidity)

In [29]: cor1

Out[29]: PearsonRResult(statistic=-0.8255712699564425, pvalue=7.673071181466846e-26)

In [ ]:
```

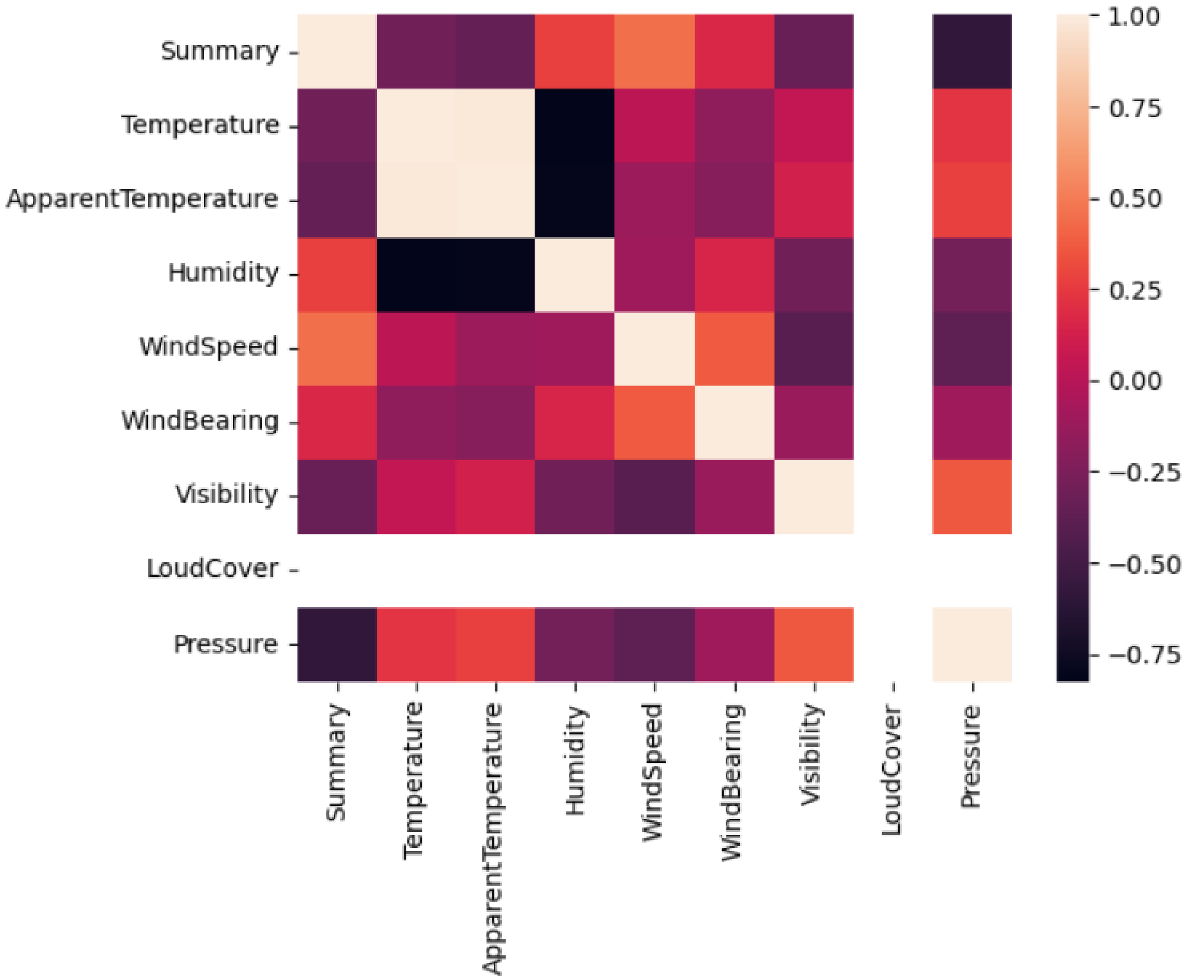
```
In [42]: cor2=data1.corr()

In [43]: cor2

Out[43]:
```

	Summary	Temperature	ApparentTemperature	Humidity	WindSpeed	WindBearing	Visibility	LoudCover	Pressure
Summary	1.000000	-0.292924	-0.350795	0.270933	0.448656	0.167242	-0.335614	NaN	-0.584766
Temperature	-0.292924	1.000000	0.978764	-0.825571	0.026648	-0.163438	0.048935	NaN	0.224109
ApparentTemperature	-0.350795	0.978764	1.000000	-0.805435	-0.112981	-0.205719	0.125471	NaN	0.269278
Humidity	0.270933	-0.825571	-0.805435	1.000000	-0.099384	0.155319	-0.304906	NaN	-0.289034
WindSpeed	0.448656	0.026648	-0.112981	-0.099384	1.000000	0.365903	-0.401903	NaN	-0.377355
WindBearing	0.167242	-0.163438	-0.205719	0.155319	0.365903	1.000000	-0.121699	NaN	-0.102996
Visibility	-0.335614	0.048935	0.125471	-0.304906	-0.401903	-0.121699	1.000000	NaN	0.362449
LoudCover	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Pressure	-0.584766	0.224109	0.269278	-0.289034	-0.377355	-0.102996	0.362449	NaN	1.000000

```
In [ ]:
```

Heatmap in Seaborn

In [44]:
```
sb.heatmap(cor2)
plt.show()
```



Positive relationship

Negative relationship

Correlation with Spearman

```
In [45]: from scipy.stats import spearmanr

In [46]: sp1=spearmanr(data1.WindBearing,data1.WindSpeed)
         sp1

Out[46]: SignificanceResult(statistic=0.41494787001808503, pvalue=1.9510082761668254e-05)

In [ ]: 
```

- ✓ **Pearson** is suitable for features with normal distribution while **Spearman** is suitable for features that don’t follow the normal distribution.
- ✓ **Pearson** is useful for interval values while **Spearman** is mostly useful for ordinal values.

9

Chi-Square

In [50]:
```
data2=pd.read_csv('F:/00-Douglas College/1- Semester 1/3- Machine Learning in Data Science(3290)/Slides/smartphone.csv')
```

In [51]:
```
data2.head()
```

Out[51]:

	Product Name	Product URL	Brand	Sale Price	Mrp	Discount Percentage	Number Of Ratings	Number Of Reviews	Upc	Star Rating	Ram
0	XOLO T1000 (Black, 4 GB)	https://www.flipkart.com/xolo-t1000-black-4-gb...	XOLO	14153	14153	0	333	130	MOBDMKDAKQGCYZ6D	3.8	1 GB
1	GIONEE Pioneer P3 (White, 4 GB)	https://www.flipkart.com/gionee-pioneer-p3-whi...	GIONEE	6500	6500	0	437	78	MOBDRKHTA3UXHAVD	3.6	512 MB
2	KARBONN Titanium S4 (Black, 4 GB)	https://www.flipkart.com/karbonn-titanium-s4-b...	KARBONN	13298	13298	0	28	7	MOBDRYWHA3ZU9BRT	3.3	1 GB
3	KARBONN Titanium S4 (White, 4 GB)	https://www.flipkart.com/karbonn-titanium-s4-w...	KARBONN	14990	14990	0	28	7	MOBDRYWHFVVQHQVZ	3.3	1 GB
4	Micromax Bolt A71 (Black, 165 MB)	https://www.flipkart.com/micromax-bolt-a71-bla...	Micromax	6499	7499	13	61	8	MOBDSMAJ5UUJUDDE	3.1	512 MB

In [ ]:

Chi-Square

In [52]:
```
from scipy.stats import chi2_contingency
```

In [53]:
```
table1=pd.crosstab(data2.Brand,data2.Ram)
```

In [54]:
```
table1
```

Out[54]:

Ram Brand	1 GB	1.5 GB	12 GB	2 GB	256 MB	3 GB	4 GB	512 MB	6 GB	8 GB
ASUS	4	0	1	2	0	4	2	0	1	1
Alcatel	2	0	0	2	0	6	0	0	0	0
Apple	0	0	0	13	0	1	29	0	19	0
BlackZone	1	0	0	6	0	3	0	0	0	0
Bluboo	0	0	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
Zoom	0	0	0	0	0	2	0	0	0	0
iball	1	0	0	0	0	0	0	0	0	0
mobiistar	0	0	0	0	0	2	0	0	0	0
realme	0	0	6	3	0	17	21	0	17	25
ringme	0	0	0	19	0	0	0	0	0	0

67 rows × 10 columns

In [ ]:

Chi-Square


```
In [57]: chi2,p_value,dof,expected=chi2_contingency(table1.values)

In [58]: chi2

Out[58]: 3287.182999615207

In [59]: p_value

Out[59]: 0.0

In [60]: dof

Out[60]: 594

In [61]: expected

Out[61]: array([[8.03040317e-01, 9.91407799e-03, 1.88367482e-01, 3.53932584e+00,
                 9.91407799e-03, 2.77594184e+00, 3.76734964e+00, 5.94844679e-02,
                 2.26040978e+00, 1.58625248e+00],
                [5.35360212e-01, 6.60938533e-03, 1.25578321e-01, 2.35955056e+00,
                 6.60938533e-03, 1.85062789e+00, 2.51156642e+00, 3.96563120e-02,
                 1.50693985e+00, 1.05750165e+00],
                [3.31923331e+00, 4.09781890e-02, 7.78585592e-01, 1.46292135e+01,
                 4.09781890e-02, 1.14738929e+01, 1.55717118e+01, 2.45869134e-01,
                 9.34302710e+00, 6.55651024e+00],
                [5.35360212e-01, 6.60938533e-03, 1.25578321e-01, 2.35955056e+00,
                 6.60938533e-03, 1.85062789e+00, 2.51156642e+00, 3.96563120e-02,
                 1.50693985e+00, 1.05750165e+00],
                [5.35360212e-02, 6.60938533e-04, 1.25578321e-02, 2.35955056e-01,
                 6.60938533e-04, 1.85062789e-01, 2.51156642e-01, 3.96563120e-03,
                 1.50693985e-01, 1.05750165e-01],
                [1.07072042e-01, 1.32187707e-03, 2.51156642e-02, 4.71910112e-01,
                 1.32187707e-03, 3.70125578e-01, 5.02313285e-01, 7.93126239e-03,
                 3.01387971e-01, 2.11500330e-01],
                [1.60608063e-01, 1.98281560e-03, 3.76734964e-02, 7.07865169e-01,
```