

1. Task - ExplorationAnalysisAndClustering

January 19, 2024

1 1. Task - Exploration data analysis and clustering

The goal of this task is to explore (analyze) the dataset using explorative and statistical techniques to get more familiar with the dataset and its characteristics, specifically:

- Attribute value data types
- Attributes distribution
- Correlation between other attributes and its covariance

And also perform Clustering on this dataset

1.1 Dataset - Weather in Australia

- Link to dataset - <https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package/>
- The goal of this dataset is to: **Predict next-day rain by training classification models on the target variable RainTomorrow.**
- This dataset contains about 10 years of daily weather observations from numerous Australian weather stations.

1.2 Import packages

```
[ ]: import pandas as pd # dataframes
import numpy as np # matrices and linear algebra
import matplotlib.pyplot as plt # plotting
import seaborn as sns # another matplotlib interface - styled and easier to use

pd.set_option('display.max_columns', None) # Show all columns
```

1.3 Load dataset

```
[ ]: pathToDataset = "..\WeatherAUS_Data\weatherAUS.csv"
origDataframe = pd.read_csv(pathToDataset)
```

1.4 Examine our dataset

1.4.1 Quick Overview

- Firstly, quick overview of our dataset

origDataframe							
	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	\
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	
...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	
\							
	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm		\
0	NaN	W	44.0	W	WNW		
1	NaN	NNW	44.0	NNW	WSW		
2	NaN	WSW	46.0	W	WSW		
3	NaN	NE	24.0	SE	E		
4	NaN	W	41.0	ENE	NW		
...	
145455	NaN	E	31.0	SE	ENE		
145456	NaN	NNW	22.0	SE	N		
145457	NaN	N	37.0	SE	WNW		
145458	NaN	SE	28.0	SSE	N		
145459	NaN	NaN	NaN	ESE	ESE		
\							
	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am		\
0	20.0	24.0	71.0	22.0	1007.7		
1	4.0	22.0	44.0	25.0	1010.6		
2	19.0	26.0	38.0	30.0	1007.6		
3	11.0	9.0	45.0	16.0	1017.6		
4	7.0	20.0	82.0	33.0	1010.8		
...	
145455	13.0	11.0	51.0	24.0	1024.6		
145456	13.0	9.0	56.0	21.0	1023.5		
145457	9.0	9.0	53.0	24.0	1021.0		
145458	13.0	7.0	51.0	24.0	1019.4		
145459	17.0	17.0	62.0	36.0	1020.2		
\							
	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	\
0	1007.1	8.0	NaN	16.9	21.8	No	
1	1007.8	NaN	NaN	17.2	24.3	No	
2	1008.7	NaN	2.0	21.0	23.2	No	
3	1012.8	NaN	NaN	18.1	26.5	No	
4	1006.0	7.0	8.0	17.8	29.7	No	

```

...
145455      1020.3      NaN      NaN    10.1    22.4     No
145456      1019.1      NaN      NaN    10.9    24.5     No
145457      1016.8      NaN      NaN    12.5    26.1     No
145458      1016.5      3.0      2.0    15.1    26.0     No
145459      1017.9      8.0      8.0    15.0    20.9     No

      RainTomorrow
0            No
1            No
2            No
3            No
4            No
...
145455      No
145456      No
145457      No
145458      No
145459      NaN

```

[145460 rows x 23 columns]

We can see that our dataset has 145 460 rows and 23 columns (22 attributes + 1 target value). Some columns are categorical (WindGustDir, WindDir9am, RainToday, TrainTomorrow), others are represented by string (Date, Location) and also we have some real values (MinTemp, MaxTemp, Rainfall, ...)

- Show some useful characteristics of real valued columns

[]: origDataframe.describe()

```

[ ]:      MinTemp      MaxTemp      Rainfall      Evaporation \
count  143975.000000  144199.000000  142199.000000  82670.000000
mean    12.194034    23.221348    2.360918     5.468232
std     6.398495    7.119049    8.478060     4.193704
min    -8.500000   -4.800000    0.000000     0.000000
25%    7.600000    17.900000    0.000000     2.600000
50%    12.000000    22.600000    0.000000     4.800000
75%    16.900000    28.200000    0.800000     7.400000
max    33.900000    48.100000   371.000000   145.000000

      Sunshine      WindGustSpeed      WindSpeed9am      WindSpeed3pm \
count  75625.000000  135197.000000  143693.000000  142398.000000
mean    7.611178    40.035230    14.043426     18.662657
std     3.785483    13.607062    8.915375     8.809800
min    0.000000     6.000000    0.000000     0.000000
25%    4.800000    31.000000    7.000000    13.000000
50%    8.400000    39.000000   13.000000    19.000000

```

75%	10.600000	48.000000	19.000000	24.000000		\
max	14.500000	135.000000	130.000000	87.000000		
	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm		\
count	142806.000000	140953.000000	130395.000000	130432.000000		
mean	68.880831	51.539116	1017.64994	1015.255889		
std	19.029164	20.795902	7.10653	7.037414		
min	0.000000	0.000000	980.50000	977.100000		
25%	57.000000	37.000000	1012.90000	1010.400000		
50%	70.000000	52.000000	1017.60000	1015.200000		
75%	83.000000	66.000000	1022.40000	1020.000000		
max	100.000000	100.000000	1041.00000	1039.600000		
	Cloud9am	Cloud3pm	Temp9am	Temp3pm		
count	89572.000000	86102.000000	143693.000000	141851.00000		
mean	4.447461	4.509930	16.990631	21.68339		
std	2.887159	2.720357	6.488753	6.93665		
min	0.000000	0.000000	-7.200000	-5.40000		
25%	1.000000	2.000000	12.300000	16.60000		
50%	5.000000	5.000000	16.700000	21.10000		
75%	7.000000	7.000000	21.600000	26.40000		
max	9.000000	9.000000	40.200000	46.70000		

By looking at this table, we can see some useful characteristic about these attributes:

- The count of attributes does not correspond with the count of rows, so some rows does not have a value for that specific attribute
- Some attributes goes from 0 to 100 (Humidity attributes), others goes to negative values (Temp), others has small range such as from 0 to 9 (Cloud) and Pressures goes above 1000
- From previous examination, we get to the conclusion that these attributes has different scales, so we need them scale down/up so all attributes has the same scale

Now, let's take a look at the categorical attributes

```
[ ]: origDataframe.describe(include='all').drop(columns=origDataframe.describe().columns)
```

	Date	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday	\
count	145460	145460	135134	134894	141232	142199	
unique	3436	49	16	16	16	2	
top	2013-11-12	Canberra	W	N	SE	No	
freq	49	3436	9915	11758	10838	110319	
mean	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	NaN	

```

max           NaN      NaN      NaN      NaN      NaN      NaN
RainTomorrow
count        142193
unique         2
top            No
freq        110316
mean          NaN
std            NaN
min            NaN
25%           NaN
50%           NaN
75%           NaN
max           NaN

```

We can see unique values, as well as the most occurring value with its frequency. For example, attribute WindDir9am has 16 unique categorical values with value ‘N’ (North) the most occurring one and its frequency is 11758 out of 134894.

1.4.2 Attribute distributions

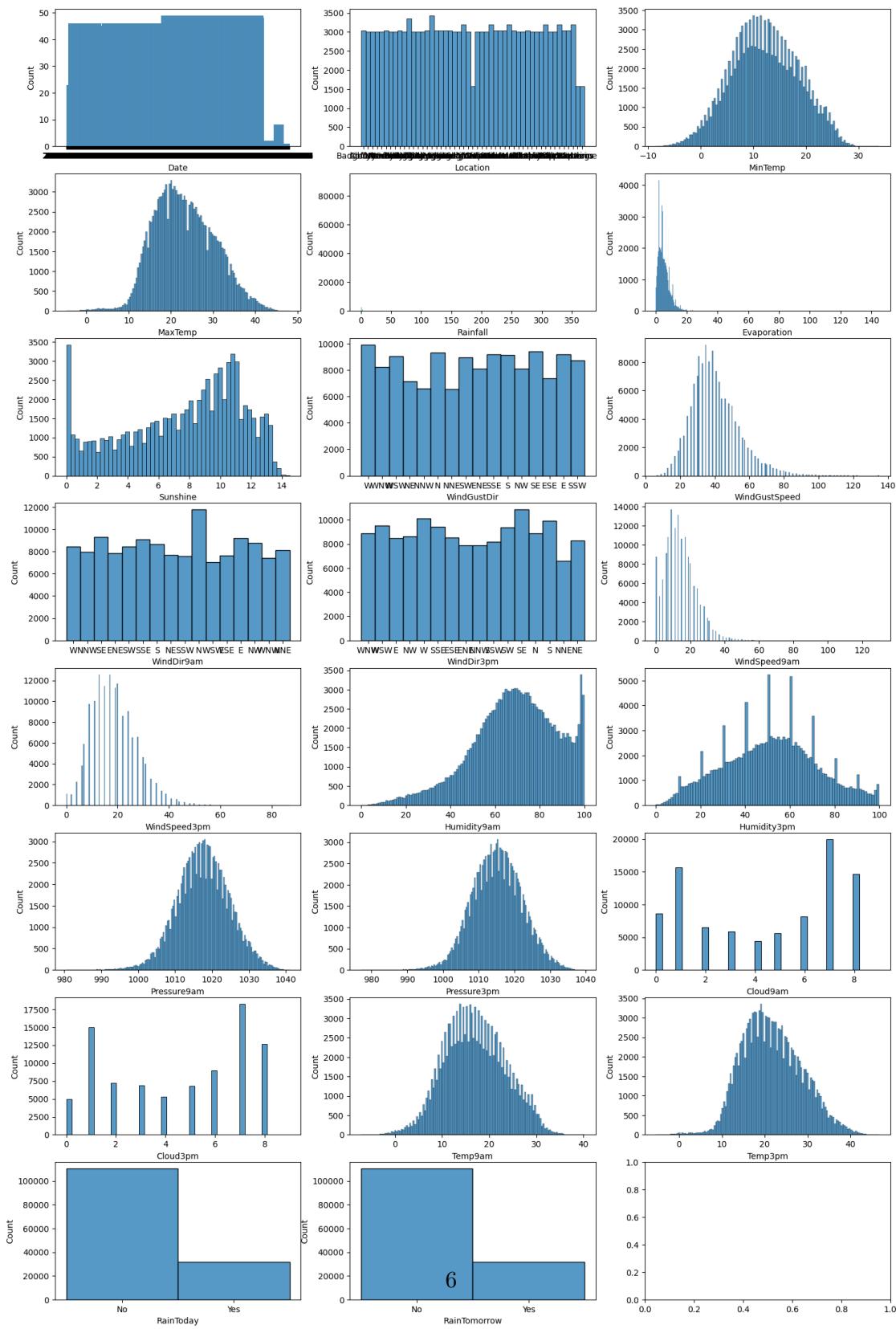
- Now let’s take a look at distribution of attributes. Histograms will do the job.

```
[ ]: fig, axes = plt.subplots(8, 3, figsize=(18, 28))
fig.suptitle('Attribute distributions')

rowIndex = 0
columnIndex = 0
for column in origDataframe.columns:
    sns.histplot(ax=axes[rowIndex, columnIndex], data=origDataframe[column])

    if columnIndex == 2:
        rowIndex += 1
        columnIndex = 0
    else:
        columnIndex += 1
```

Attribute distributions



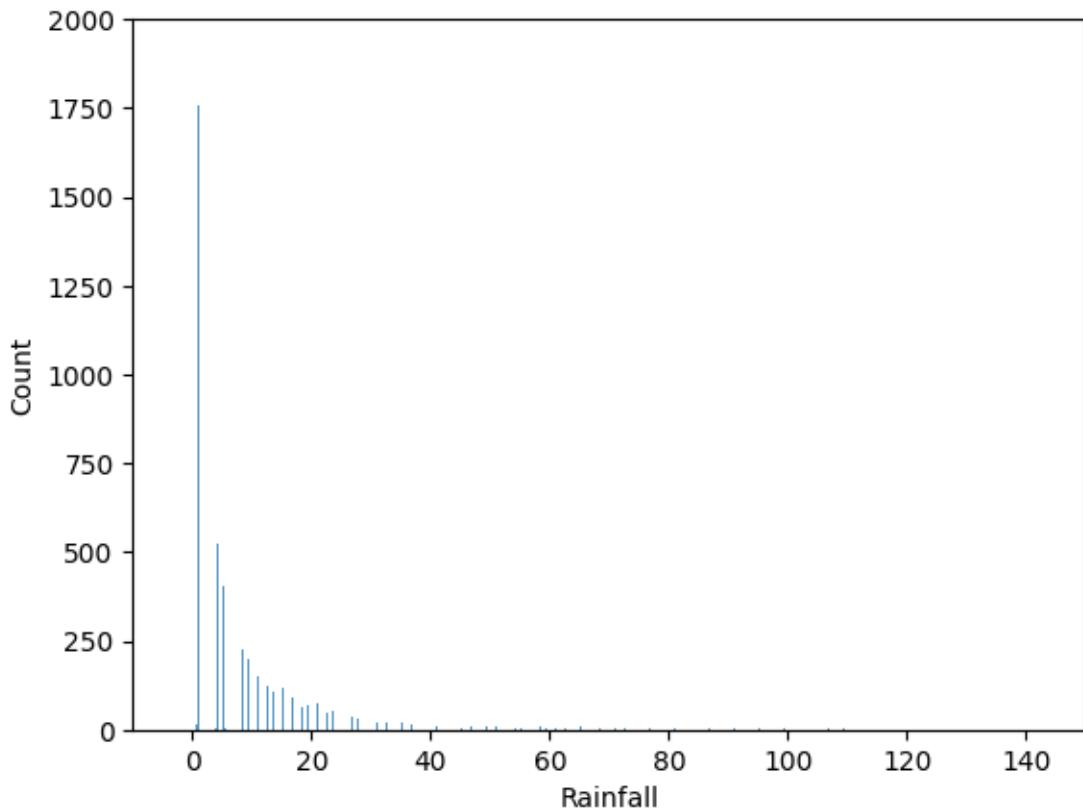
I've plotted histograms for all attributes.

- Attributes ‘Date’, ‘Location’ has uniform distribution with a few outliers
- Attributes ‘WindGustDir’, ‘WindDir9am’, ‘WindDir3pm’ has distribution close to uniform, ‘WindDir9am’ can also be considered as Gaussian (normal) distribution
- Other than mentioned attributes and ‘RainToday’, ‘RainTomorrow’, we notice that every attribute has Gaussian (normal) distribution, which is related to the fact that distributions gathered from environment tends to have Gaussian (normal) distribution.
- ‘Cloud9am’ and ‘Cloud3pm’ attributes have 2 peaks of Gaussian (normal) distribution
- Distribution for ‘RainToday’ and ‘RainTomorrow’ looks almost identical

Let's zoom a little bit to the ‘Rainfall’ column to see the distribution

```
[ ]: sns.histplot(origDataframe['Rainfall'])
plt.ylim(0, 2000)
plt.xlim(-10, 150)
```

```
[ ]: (-10.0, 150.0)
```



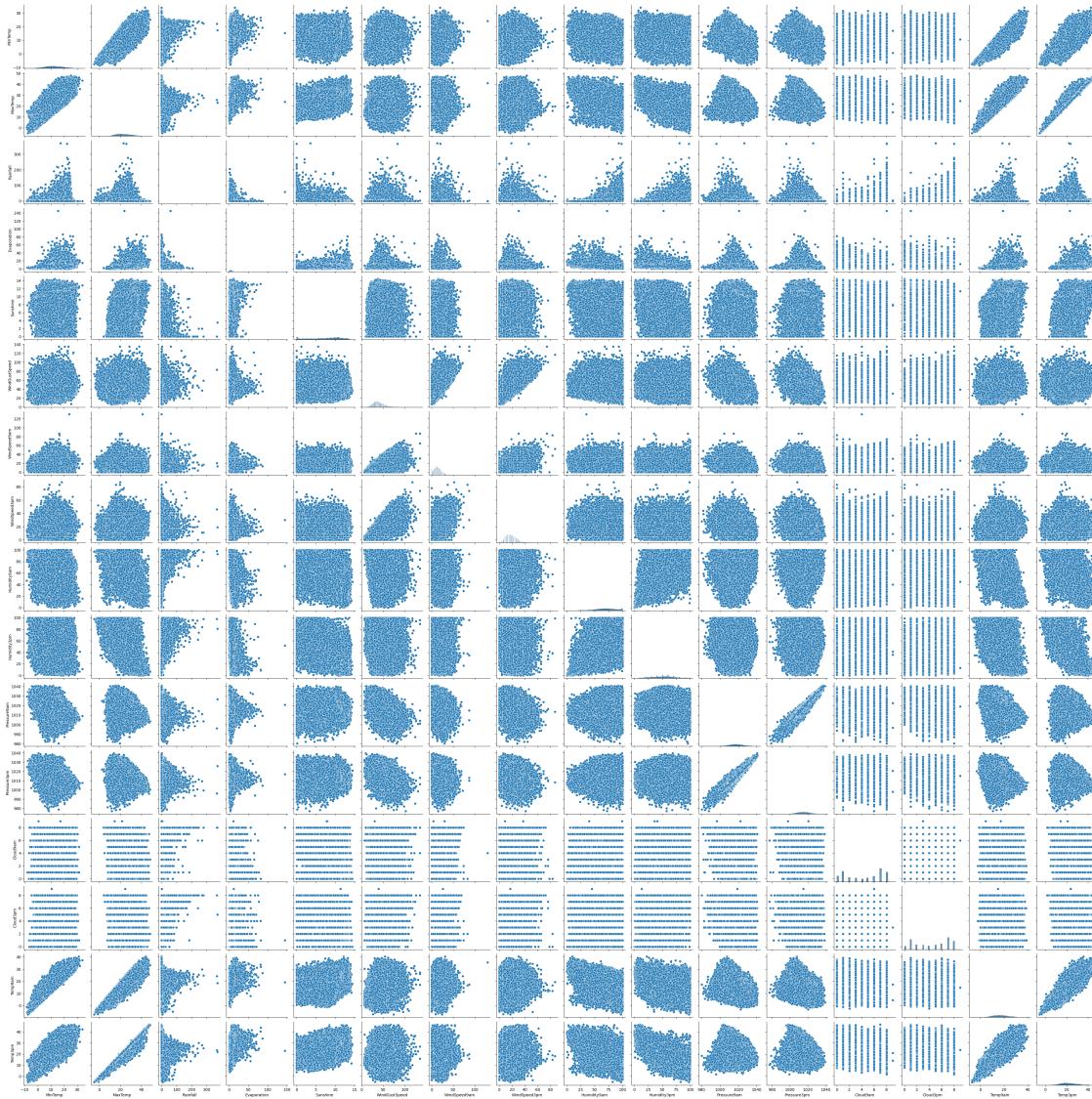
1.4.3 Attributes covariance

Recall that attributes has weak covariance between each other when their ‘scatter’ plot looks like a circle and has stronger covariance in case of ellipse.

- I’ve used seaborn’s pairplot to plot all pairs of attributes, so we can take a look at those covariances.

```
[ ]: sns.pairplot(origDataframe)
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x26bbb859270>
```

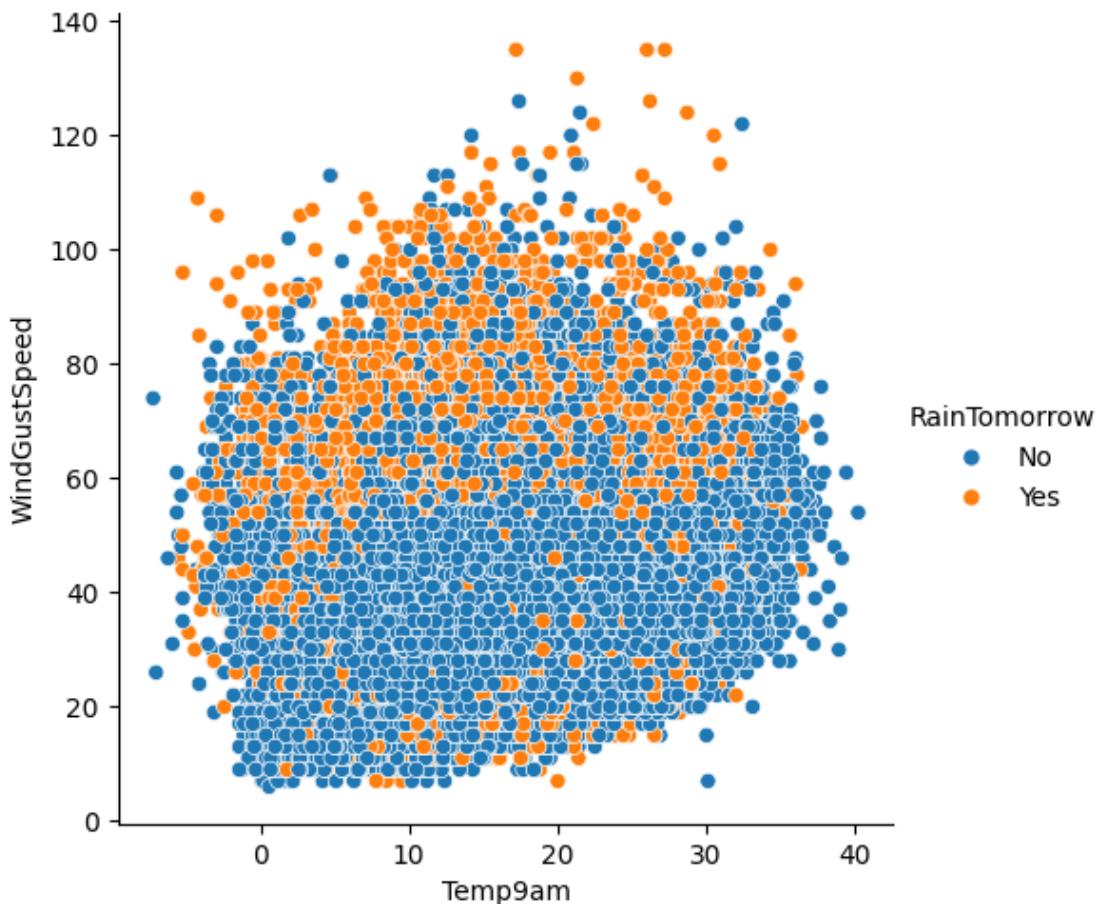


1.4.4 Closer look on the covariance

- Below I've plotted one of the most weak covariance in the dataset. The attributes 'Temp9am' and 'WindGustSpeed' does not relate between each other very much.

```
[ ]: sns.relplot(origDataframe, x='Temp9am', y='WindGustSpeed', hue='RainTomorrow')
```

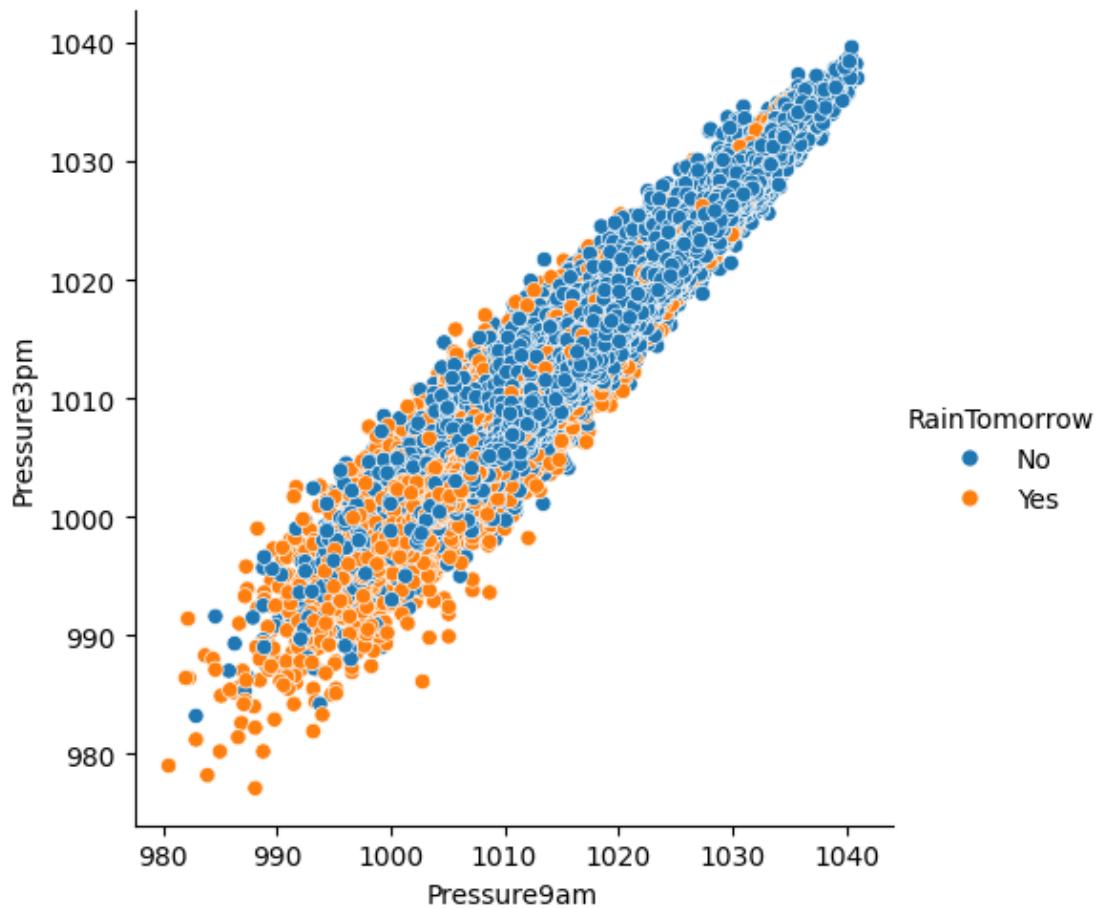
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c09b6d930>
```



- Now let's take a look at strong positive covariance

```
[ ]: sns.relplot(origDataframe, x='Pressure9am', y='Pressure3pm', hue='RainTomorrow')
```

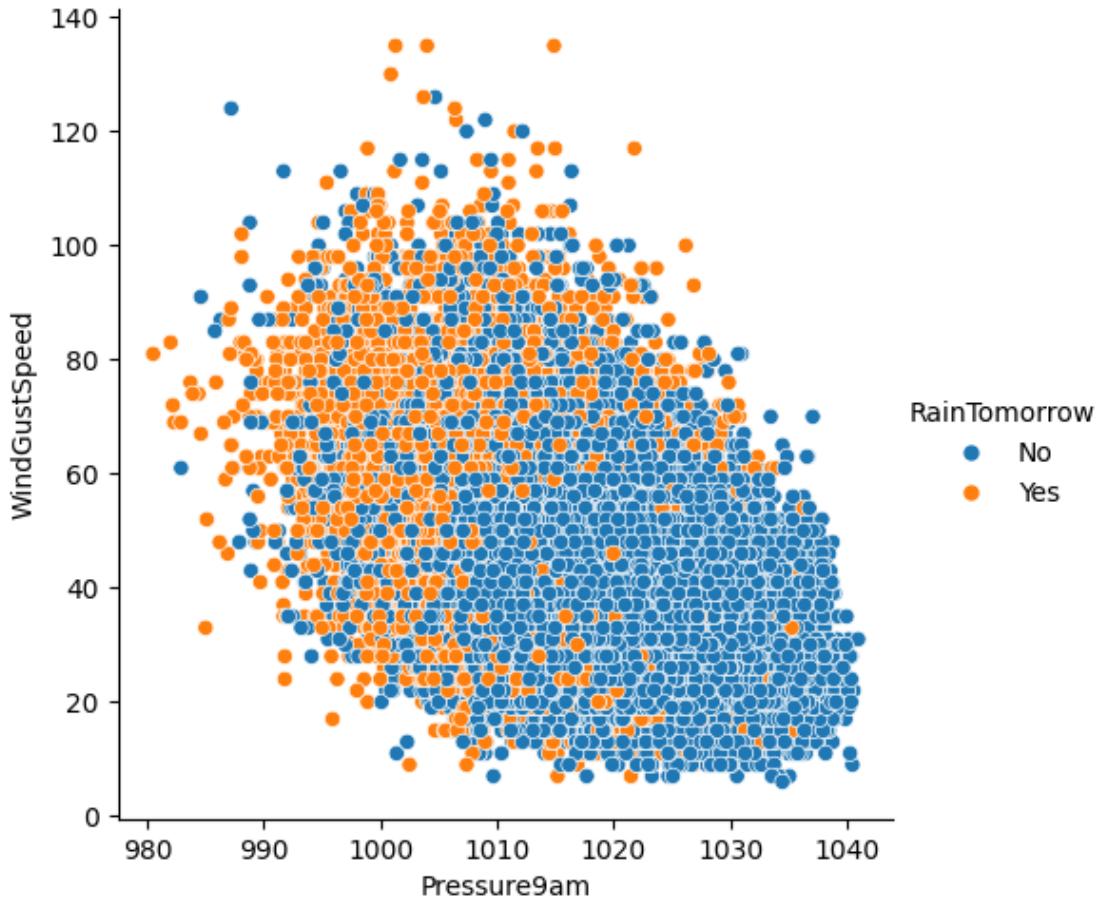
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c28dbfe80>
```



- Example of negative covariance, although not as strong as the previous one

```
[ ]: sns.relplot(origDataframe, x='Pressure9am', y='WindGustSpeed',  
    hue='RainTomorrow')
```

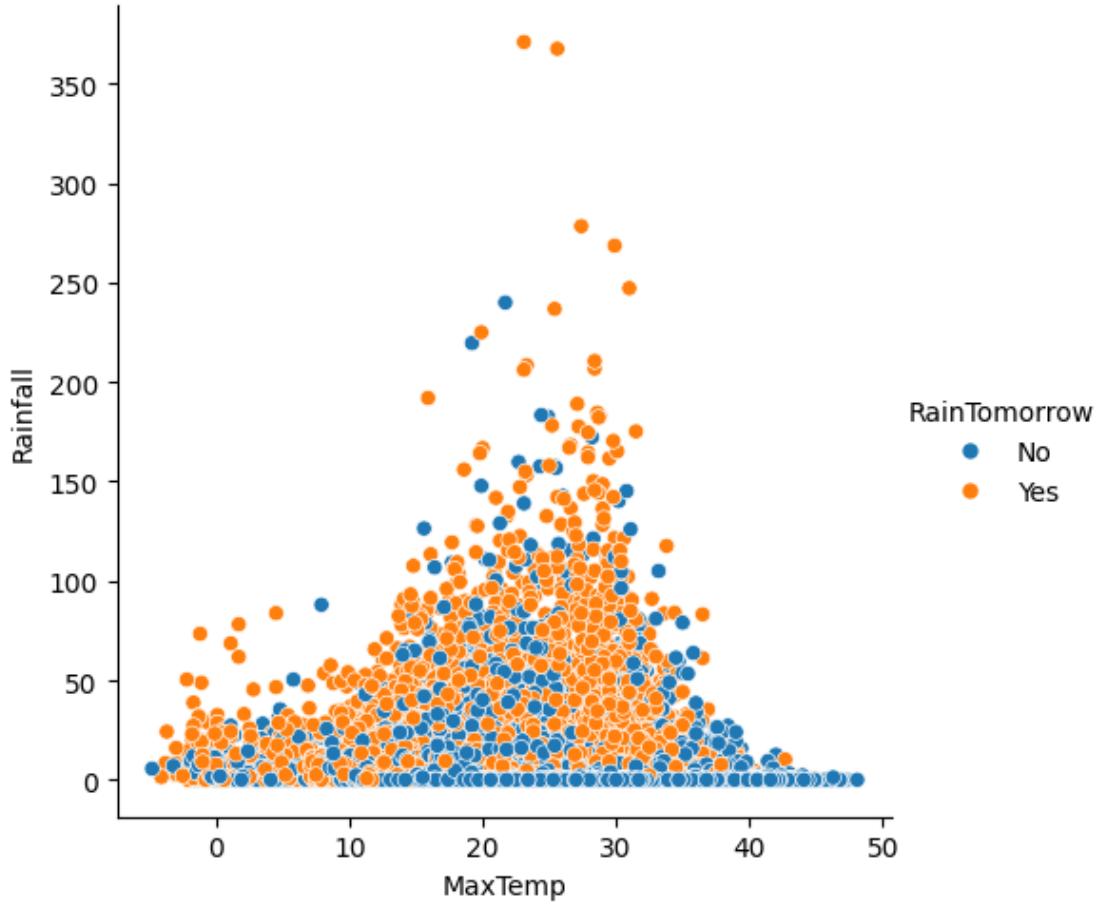
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c28dd5720>
```



- Some plots are interesting such this one. We can clearly observe that the higher rainfalls tends to have temperature between 15 to 30 degrees.

```
[ ]: sns.relplot(origDataframe, x='MaxTemp', y='Rainfall', hue='RainTomorrow')
```

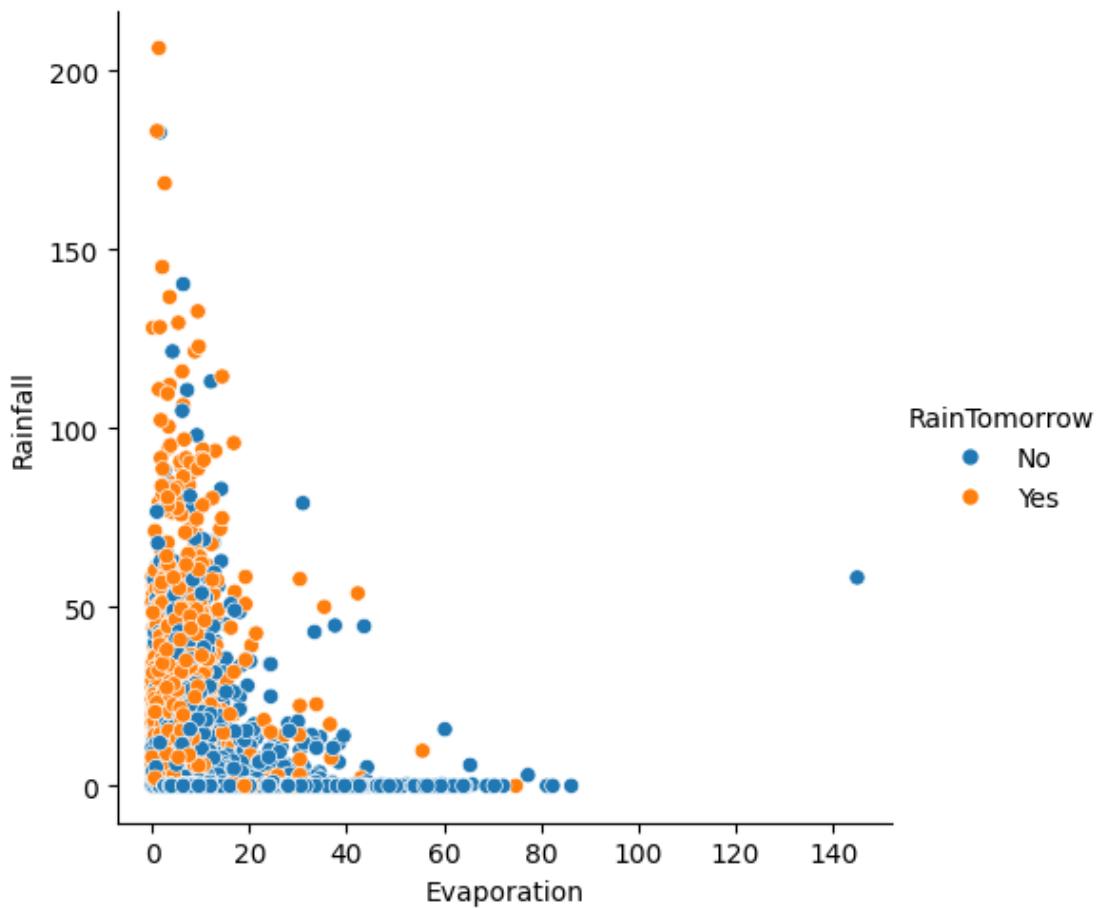
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c39ebca90>
```



- Also during rainfall, there's less evaporation

```
[ ]: sns.relplot(origDataframe, x='Evaporation', y='Rainfall', hue='RainTomorrow')
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c411d0c40>
```



1.4.5 Attribute correlation

Now let's take a look at the correlation.

- Correlation give us the measure of how strong or weak is the relation between attributes.

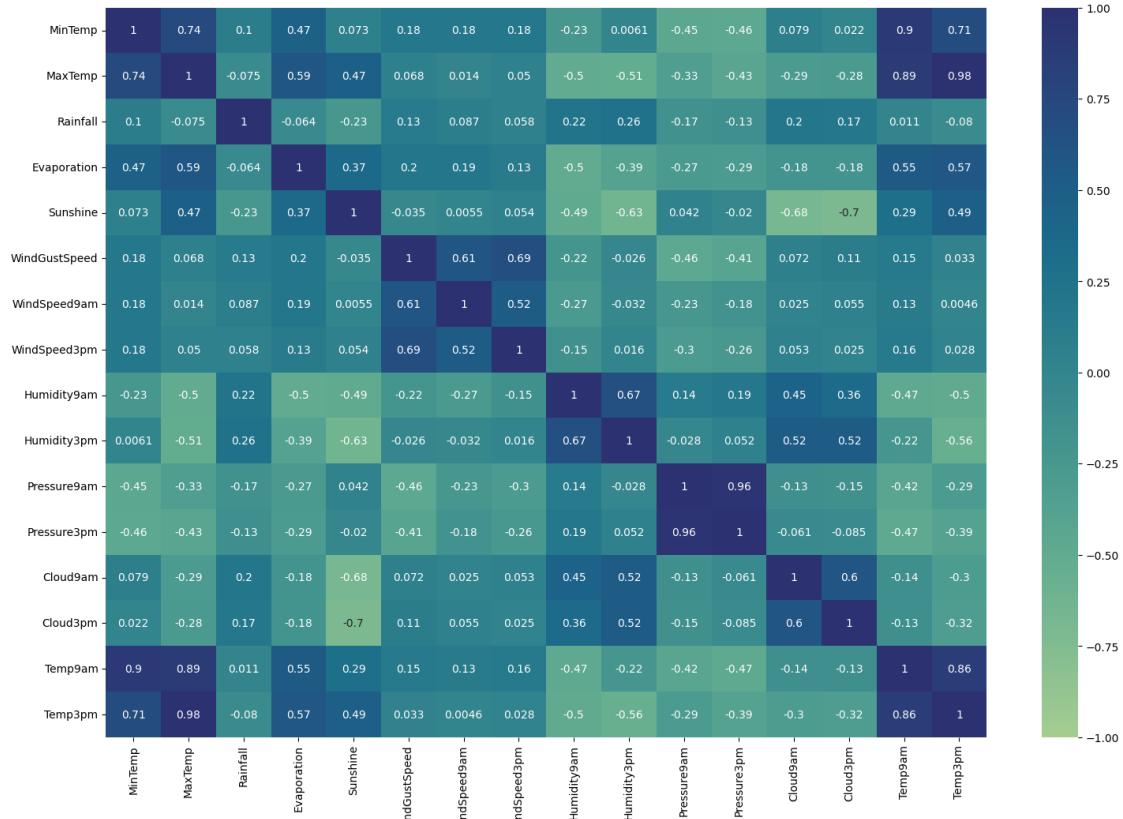
Select only attributes with numerical values

```
[ ]: datafram = origDataframe.loc[:, origDataframe.describe().columns].copy()
```

Plot heatmap

```
[ ]: plt.figure(figsize=(18,12))
sns.heatmap(dataframe.corr(), cmap='crest', vmin=-1, vmax=1, annot=True)
```

```
[ ]: <Axes: >
```



By plotting heatmap, we'll get insights about correlation between attributes:

- Some attributes are highly correlated, such as ‘MaxTemp’ and ‘MinTemp’, ‘Temp9am’ and ‘MinTemp’, ‘Humidity9am’ and ‘Humidity3pm’ and others with positive correlation
- Other attributes has negative correlation, such as ‘Cloud3pm’ and ‘Sunshine’, ‘Temp3pm’ and ‘Humidity3pm’, and so on
- The biggest correlation has attributes ‘Temp3pm’ and MaxTemp with value 0.98 (and it is positive correlation), which is close to 1 (the maximum possible correlation)
- The smallest correlation has attributes ‘WindSpeed9am’ and ‘Temp3pm’ with value 0.0046

1.5 Clustering

- Algorithm that groups records into sub-groups based on their similarity between each other (meaning similarity in attributes)

1.5.1 K-means algorithm

- We define the number of groups “K” (clusters)

Brief algorithm overview:

- Create K random centroids (points)

- Do iteratively:
 - Assign all examples from training set to the closest centroid (using Euclidean distance)
 - Move centroids by their means (Mean is computed for every centroid from assigned examples to them)

Before we can use our Clustering algorithm, we need to drop not relevant attributes

[]: origDataframe

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	\
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	
...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	
	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm		\
0	NaN	W	44.0	W	WNW		
1	NaN	WNW	44.0	NNW	WSW		
2	NaN	WSW	46.0	W	WSW		
3	NaN	NE	24.0	SE	E		
4	NaN	W	41.0	ENE	NW		
...	
145455	NaN	E	31.0	SE	ENE		
145456	NaN	NNW	22.0	SE	N		
145457	NaN	N	37.0	SE	WNW		
145458	NaN	SE	28.0	SSE	N		
145459	NaN	NaN	NaN	ESE	ESE		
	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am		\
0	20.0	24.0	71.0	22.0	1007.7		
1	4.0	22.0	44.0	25.0	1010.6		
2	19.0	26.0	38.0	30.0	1007.6		
3	11.0	9.0	45.0	16.0	1017.6		
4	7.0	20.0	82.0	33.0	1010.8		
...	
145455	13.0	11.0	51.0	24.0	1024.6		
145456	13.0	9.0	56.0	21.0	1023.5		
145457	9.0	9.0	53.0	24.0	1021.0		
145458	13.0	7.0	51.0	24.0	1019.4		
145459	17.0	17.0	62.0	36.0	1020.2		

```

Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday \
0           1007.1     8.0      NaN       16.9      21.8      No
1           1007.8     NaN      NaN       17.2      24.3      No
2           1008.7     NaN      2.0       21.0      23.2      No
3           1012.8     NaN      NaN       18.1      26.5      No
4           1006.0     7.0       8.0      17.8      29.7      No
...
145455     1020.3     NaN      NaN       10.1      22.4      No
145456     1019.1     NaN      NaN       10.9      24.5      No
145457     1016.8     NaN      NaN       12.5      26.1      No
145458     1016.5     3.0       2.0       15.1      26.0      No
145459     1017.9     8.0       8.0      15.0      20.9      No

RainTomorrow
0           No
1           No
2           No
3           No
4           No
...
145455     ...
145456     No
145457     No
145458     No
145459     NaN

```

[145460 rows x 23 columns]

By examining the available attributes in the training set, I concluded that every attribute is relevant in this scenario.

- Date is a good candidate to be not relevant, but naturally there is less rain during summer than during the autumn, so that's the reason I take the attribute as well.
- Location is relevant, some places have more rain than others (Tropical rainforest x desert)
- We just **need** to drop the label (target value) and the categorical values, either by dropping them or converting them to numerical values (e.g. by one-hot encoding or)

[]: `from sklearn.cluster import KMeans`

- Let's drop those rows with NaN values, but keep in mind that the clustering can be affected by this.

[]: `clusterDataframe = dataframe.dropna().copy()
clusterDataframeArray = clusterDataframe.values
print(clusterDataframeArray)`

[[17.9 35.2 0. ... 5. 26.6 33.4]]

```
[18.4 28.9 0. ... 1. 20.3 27. ]
[19.4 37.6 0. ... 6. 28.7 34.9]
...
[20.7 32.8 0. ... 0. 24.8 32.1]
[19.5 31.8 0. ... 1. 24.8 29.2]
[20.2 31.7 0. ... 5. 25.4 31. ]]
```

1.5.2 Use K-means clustering algorithm

- We'll use the prepared dataframe for that called clusterDataframe

Prepare the kMean model

```
[ ]: kMeans = KMeans(n_clusters=4, random_state=0, n_init='auto')
```

Fit the KMeans algorithm

- Then make predictions on the trained set and let's save cluster values to the dataframe

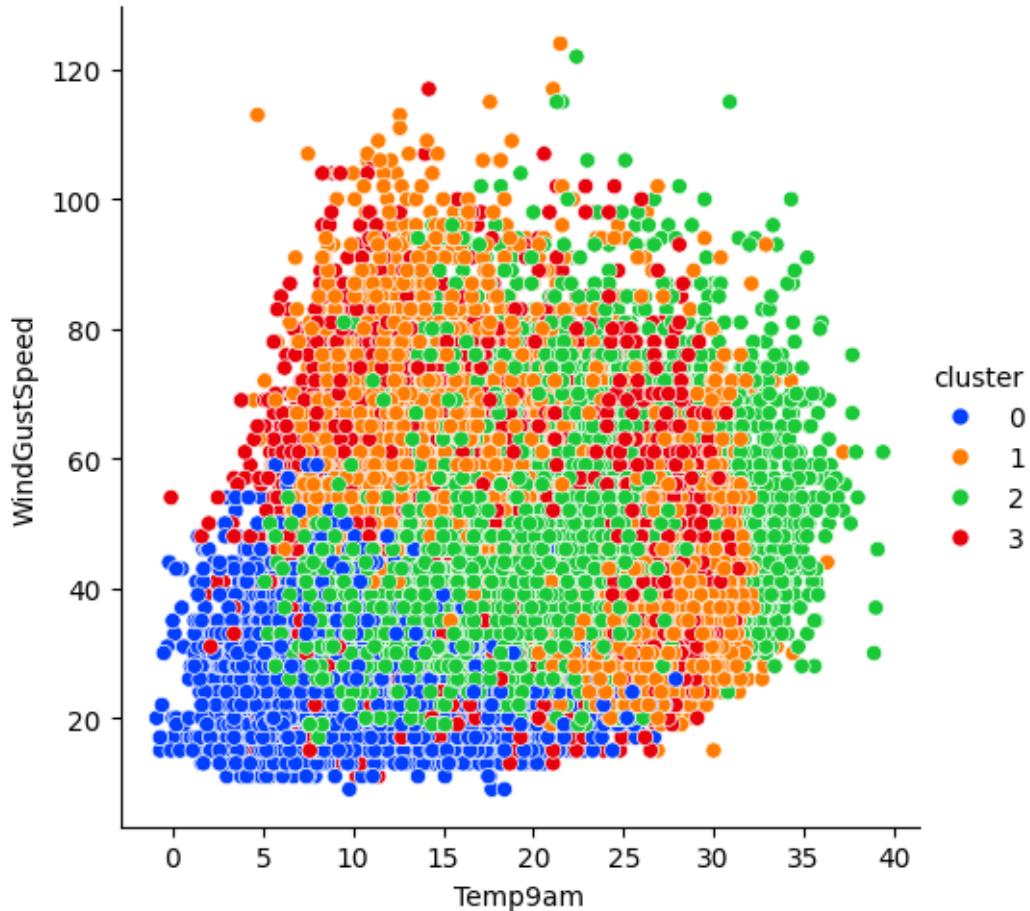
```
[ ]: kMeans.fit(clusterDataframeArray)
predictedValues = kMeans.predict(clusterDataframeArray)

clusterDataframe['cluster'] = kMeans.labels_
```

Plot the clusters on some attributes

```
[ ]: palette = sns.color_palette("bright", 4)
sns.relplot(clusterDataframe, x='Temp9am', y='WindGustSpeed', hue='cluster', palette=palette)
```

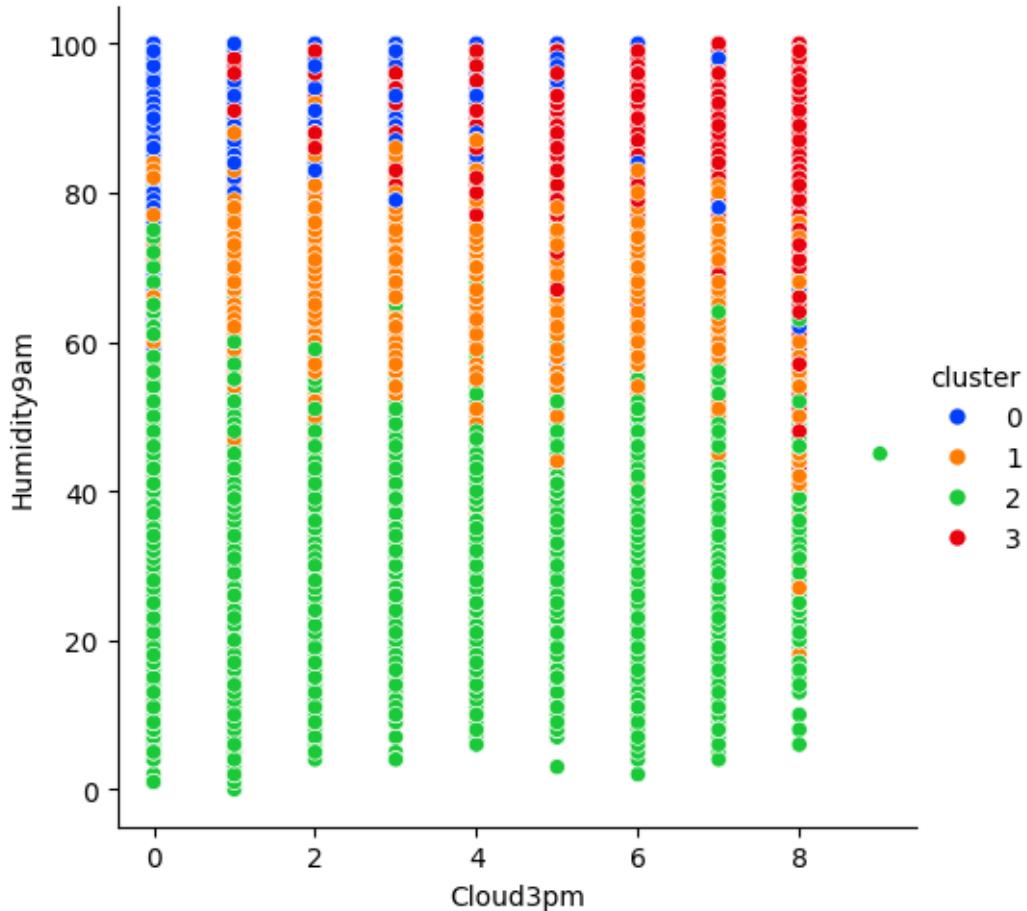
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c36acda20>
```



- As we can see from the plot above, one group is located at the top of other (the blue one in this case) indicating that this group is different than the others on this attribute correlation

```
[ ]: sns.relplot(clusterDataframe, x='Cloud3pm', y='Humidity9am', hue='cluster',  
    palette=palette)
```

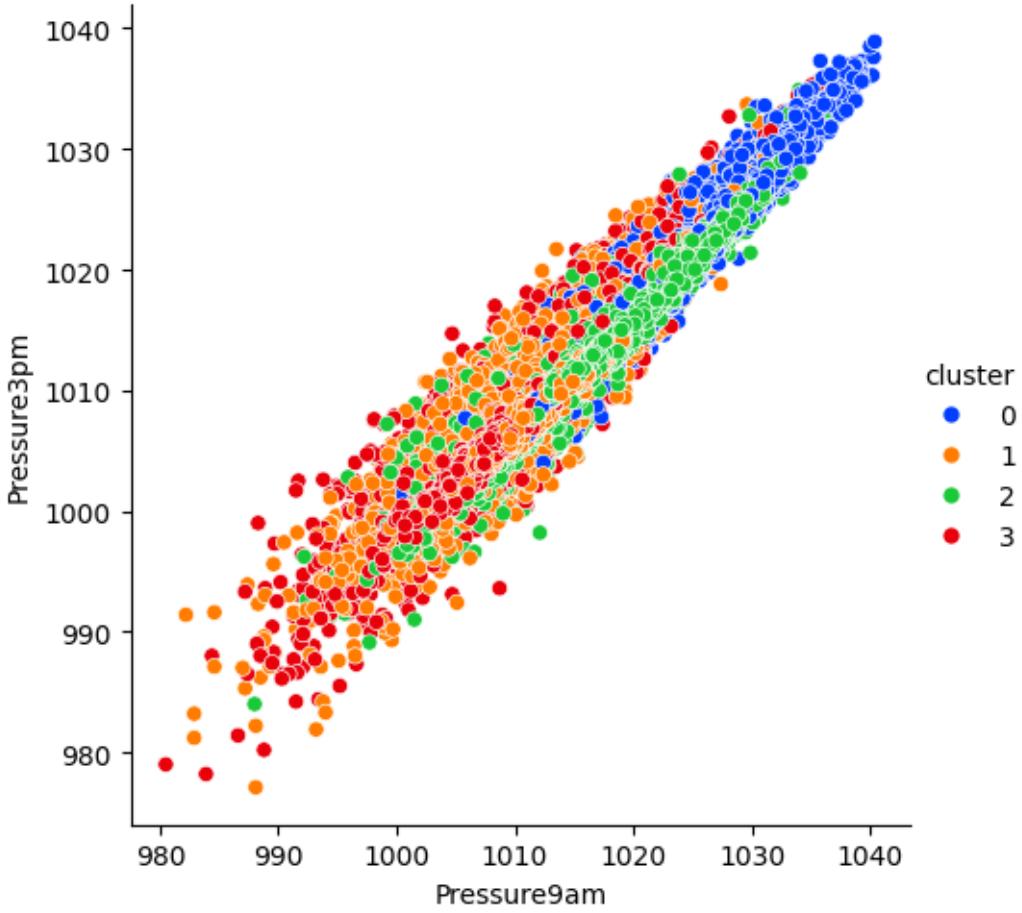
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c36ce76a0>
```



- Using different attributes, we can see that the dominant group is green and it takes first half of the plot

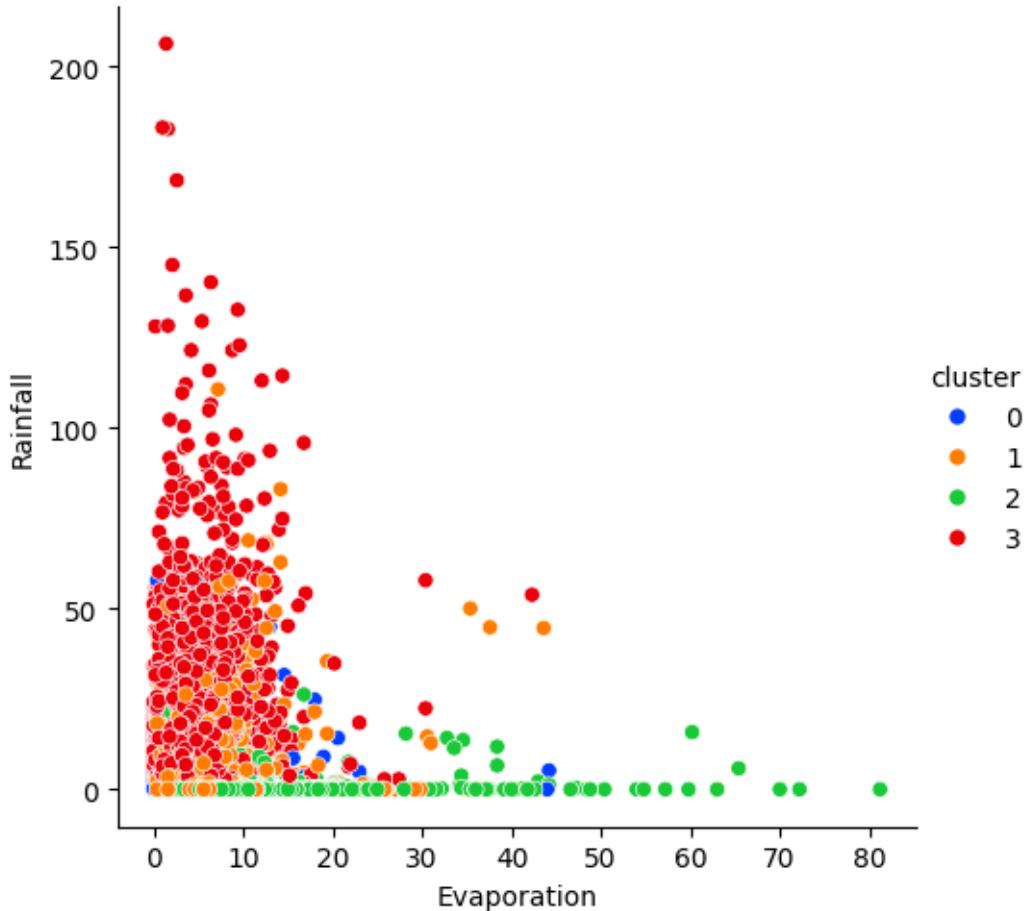
```
[ ]: sns.relplot(clusterDataframe, x='Pressure9am', y='Pressure3pm', hue='cluster',  
    palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c36d6d9f0>
```



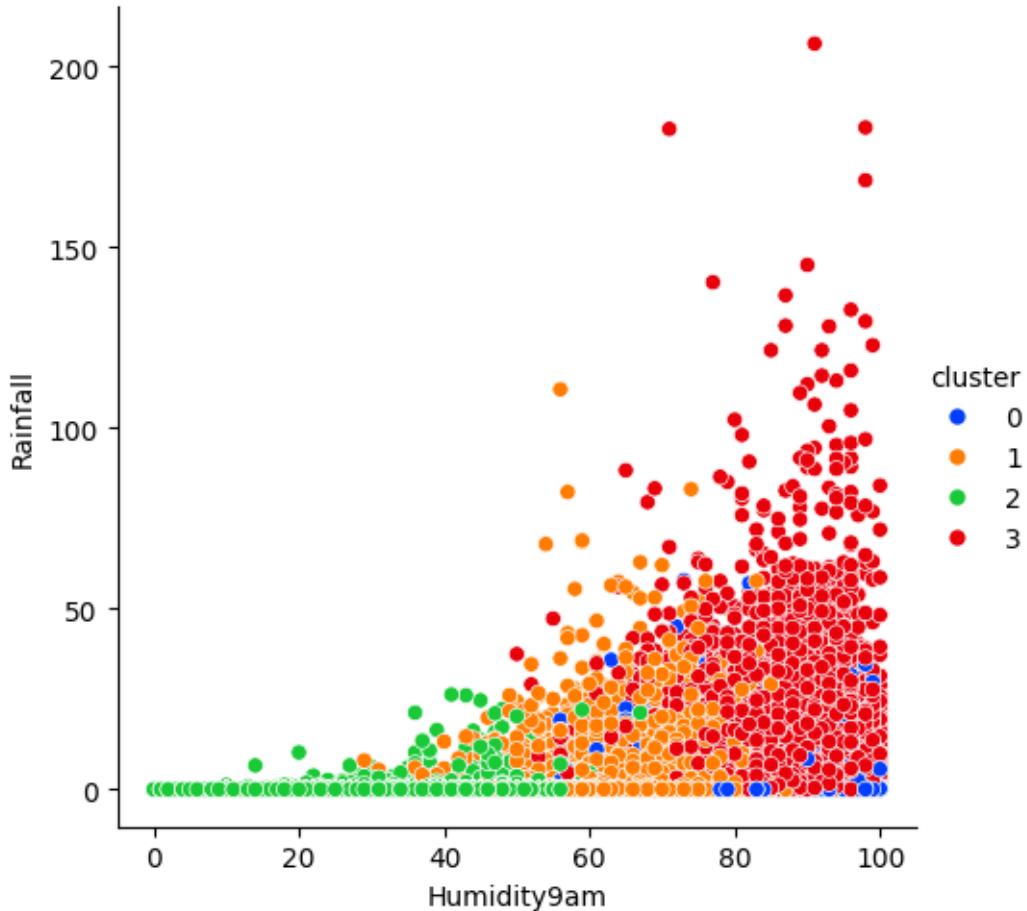
```
[ ]: sns.relplot(clusterDataframe, x='Evaporation', y='Rainfall', hue='cluster',  
    palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c37c397b0>
```



```
[ ]: sns.relplot(clusterDataframe, x='Humidity9am', y='Rainfall', hue='cluster',  
    palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c37cbd690>
```



1.5.3 Let's now use different value for K

When working with clustering, use different value for K variable. Some of the clustering may be better than others.

Also we can use different clustering algorithm called Hierarchical clustering, but I worked in the next section again with KMeans, but this time with 10 clusters and visualized them.

```
[ ]: clusterDataframe = clusterDataframe.drop('cluster', axis=1)
clusterDataframeArray = clusterDataframe.values
print(clusterDataframeArray)

kMeans = KMeans(n_clusters=10, random_state=0, n_init='auto')

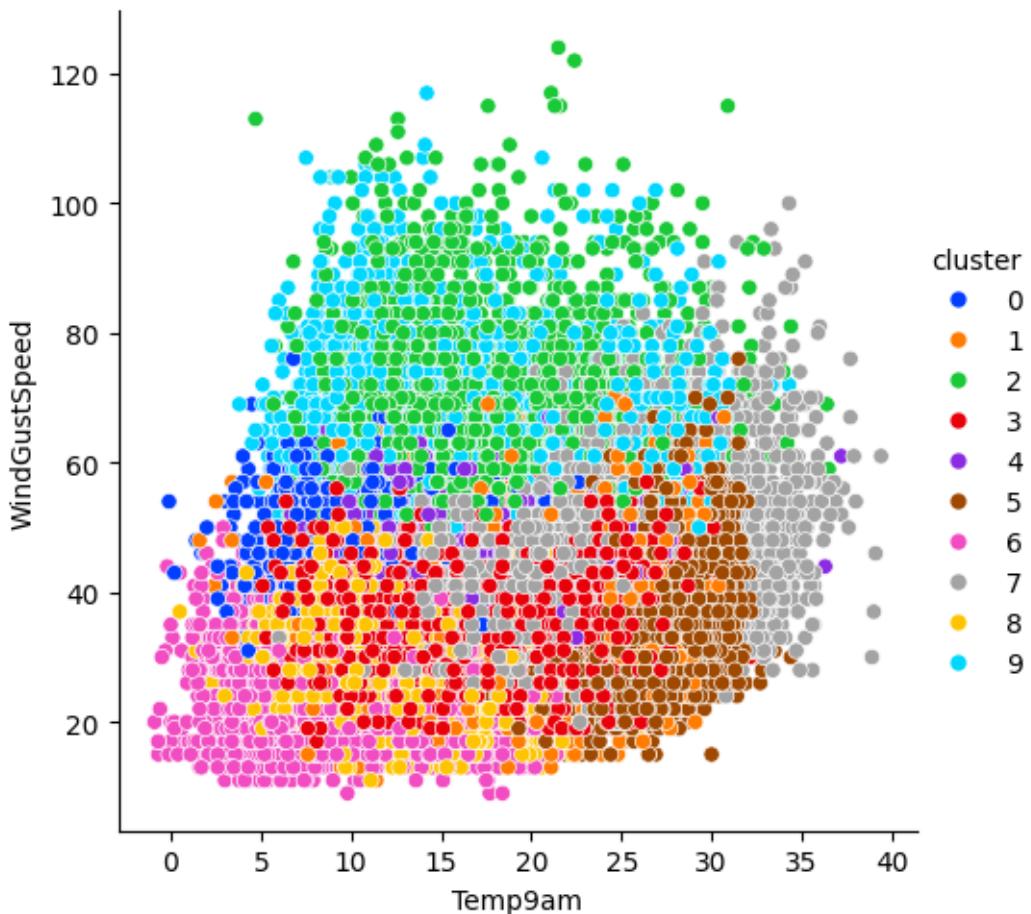
kMeans.fit(clusterDataframeArray)
predictedValues = kMeans.predict(clusterDataframeArray)

clusterDataframe['cluster'] = kMeans.labels_
```

```
[[17.9 35.2 0. ... 5. 26.6 33.4]
 [18.4 28.9 0. ... 1. 20.3 27. ]
 [19.4 37.6 0. ... 6. 28.7 34.9]
 ...
 [20.7 32.8 0. ... 0. 24.8 32.1]
 [19.5 31.8 0. ... 1. 24.8 29.2]
 [20.2 31.7 0. ... 5. 25.4 31. ]]
```

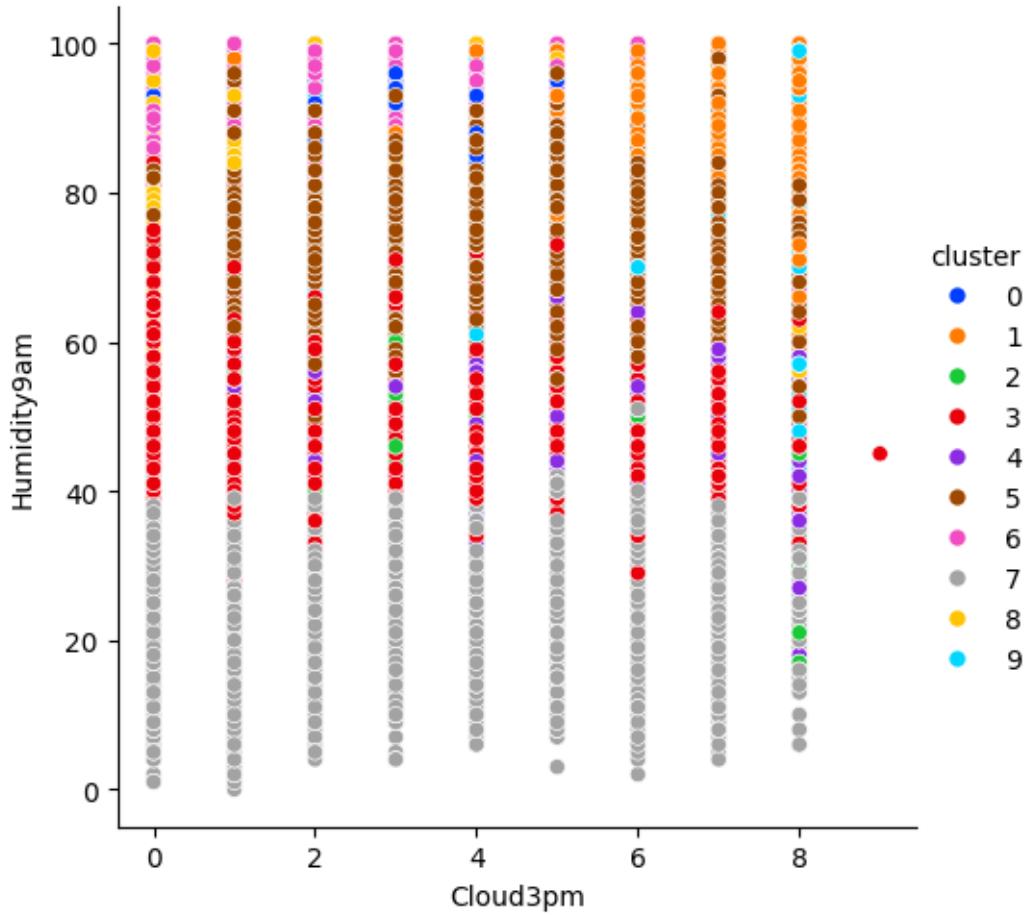
```
[ ]: palette = sns.color_palette("bright", 10)
sns.relplot(clusterDataframe, x='Temp9am', y='WindGustSpeed', hue='cluster',
            palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26bbd8c0790>
```



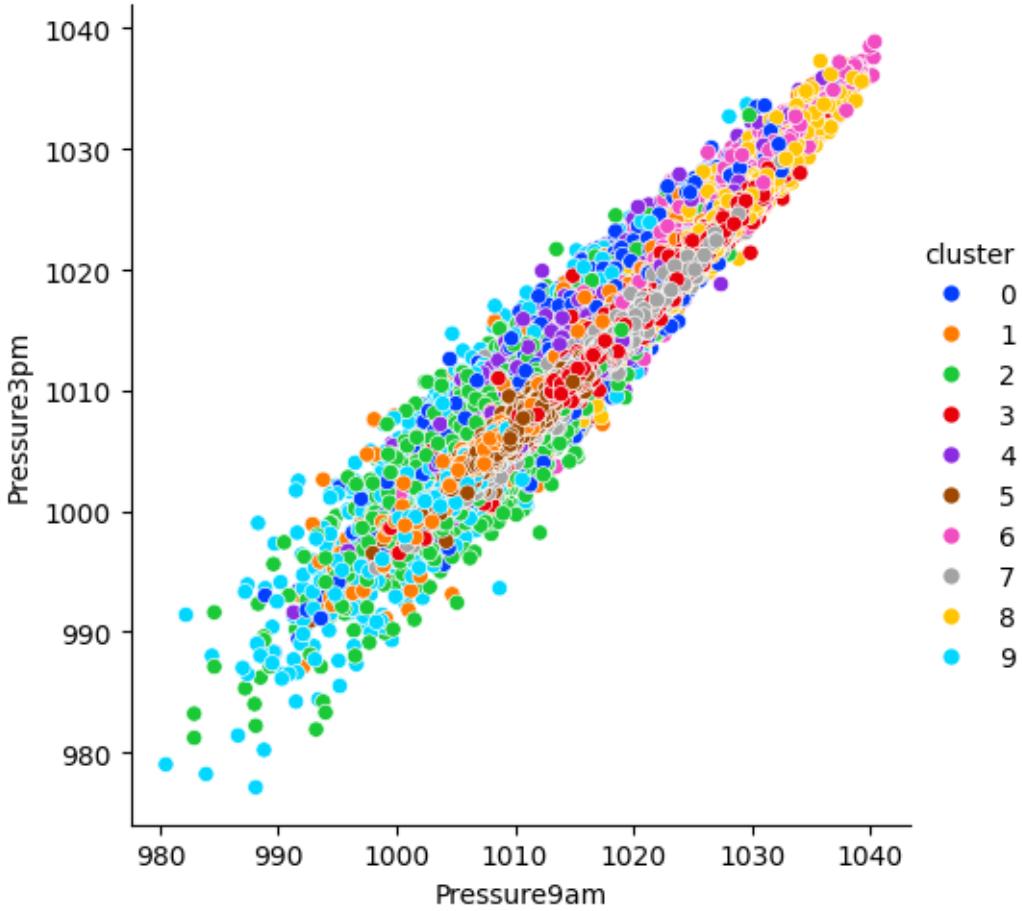
```
[ ]: sns.relplot(clusterDataframe, x='Cloud3pm', y='Humidity9am', hue='cluster',
            palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c3b2839d0>
```



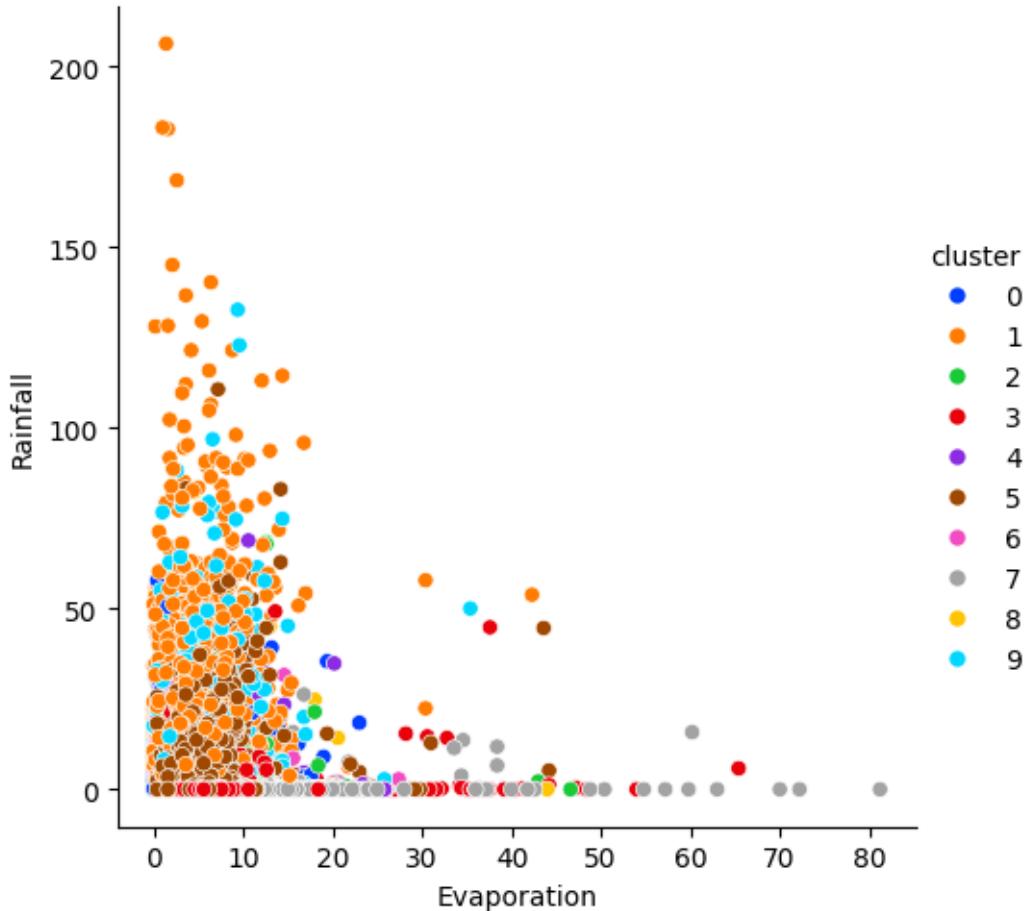
```
[ ]: sns.relplot(clusterDataframe, x='Pressure9am', y='Pressure3pm', hue='cluster',  
    ↪ palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c40fec460>
```



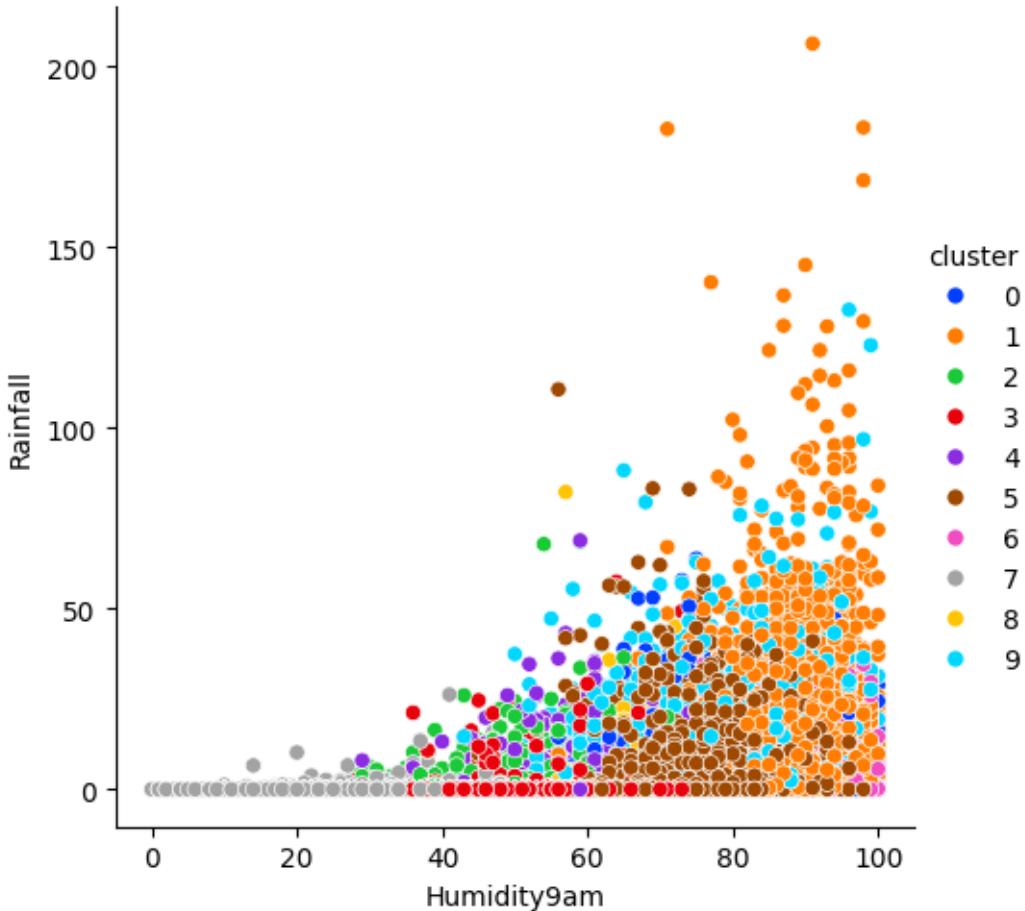
```
[ ]: sns.relplot(clusterDataframe, x='Evaporation', y='Rainfall', hue='cluster',  
    palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c41051ea0>
```



```
[ ]: sns.relplot(clusterDataframe, x='Humidity9am', y='Rainfall', hue='cluster',  
    palette=palette)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x26c417d39d0>
```



2 Summary

In this notebook, we took a look at the Australia weather dataset. Specifically, we examined the dataset by:

- By **looking at the attributes** (Whether or not they are numerical/categorical, what type of data they consist of)
- By **Attribute distribution** (What is the distribution of each attribute)
- By **Attribute covariance** (How one attribute depends on the others)
- By **Attribute correlation** (How strong is the relation between attributes)

Also we used K-Means algorithm to perform Clustering on the dataset (Firstly, with $K = 4$ and then with $K = 10$). The clustering algorithm tried to group examples that are similar with each others and gave us results as we saw in the susbequent plots.