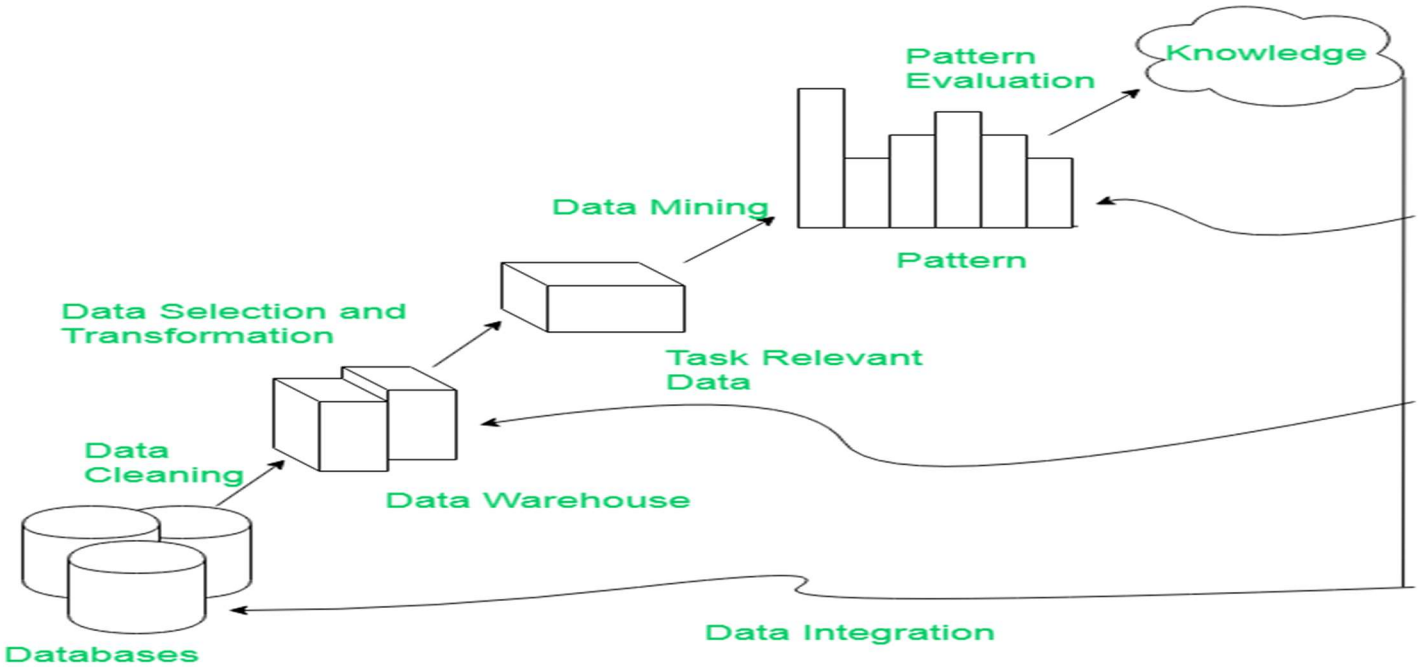
Executive Summary

- This Project is about applying Data Mining Technics which we have learned in the last 8 weeks and here is a quick summary about what we have learned. First let's talk briefly about Data Mining as It is one of the most important steps to do data analysis, it could be abbreviated as (Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data) . It also could be names in other names as KDD -Knowledge Discovery in Databases -.
- To apply Data Mining -or as we have mentioned KDD- we need to follow sequential steps we can summarise it at this image



- And the project is about applying these steps so, what is these steps?

Data cleaning (remove noise and inconsistent data)

- Cleaning in case of ***Missing values***.
- Cleaning ***noisy*** data, where noise is a random or variance error.
- Cleaning with ***Data discrepancy detection*** and ***Data transformation tools***

Data integration (multiple data sources maybe combined)

- Data integration using ***Data Migration tools***.
- Data integration using ***Data Synchronization tools***.
- Data integration using ***ETL***(Extract-Load-Transformation) process.

Data selection (data relevant to the analysis task are retrieved from database)

- Data selection using ***Neural network***.
- Data selection using ***Decision Trees***.
- Data selection using ***Naive bayes***.
- Data selection using ***Clustering***, ***Regression***, etc.

Data transformation (data transformed or consolidated into forms appropriate for mining)

- ***Data Mapping***: Assigning elements from source base to destination to capture transformations.
- ***Code generation***: Creation of the actual transformation program.

Data mining (an essential process where intelligent methods are applied to extract data patterns)

- Transforms task relevant data into ***patterns***.

- Decides purpose of model using ***classification*** or ***characterization***.

Pattern evaluation (identify the truly interesting patterns)

- Find ***interestingness score*** of each pattern.
- Uses ***summarization*** and ***Visualization*** to make data understandable by user.

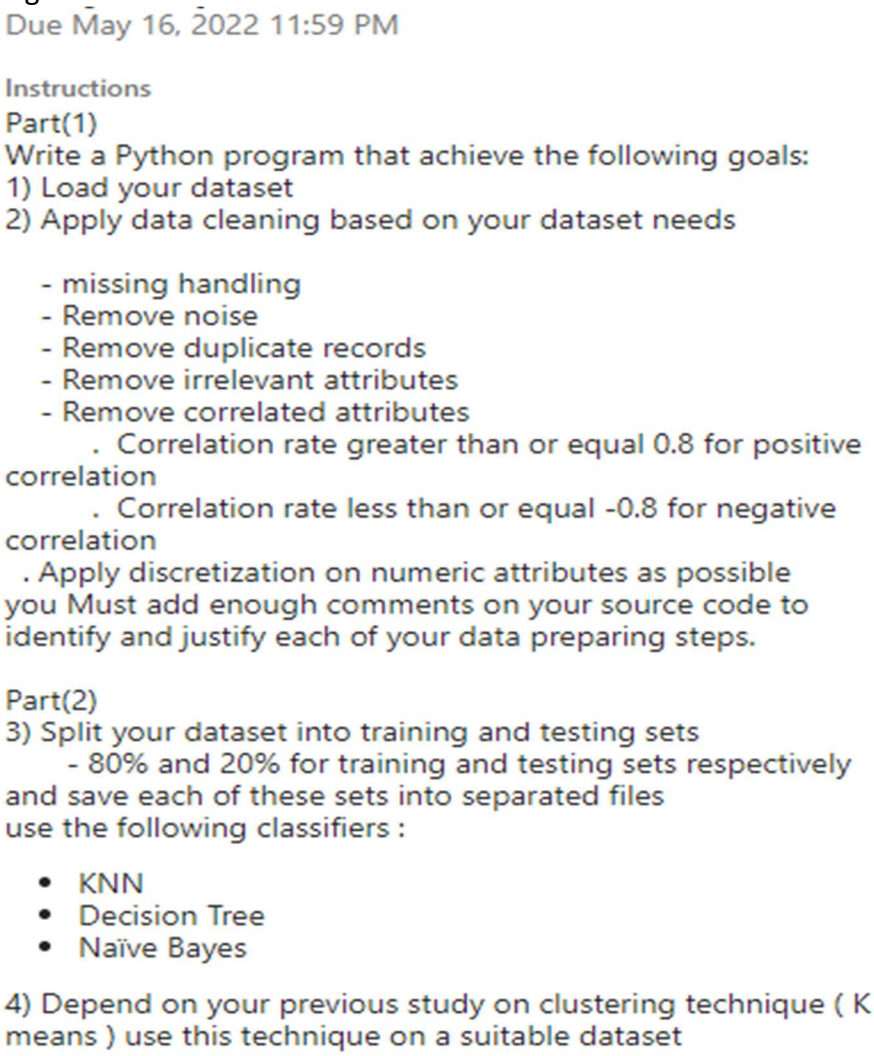
Knowledge presentation (mined knowledge is presented to the user with visualization or representation techniques)

- Generate ***reports***.
- Generate ***tables***.
- Generate ***discriminant rules***, ***classification rules***, ***characterization rules***, etc.

In each step of the project, we will talk in detail what have we done? And why?

Now let's get into the Project

- Here we have an image of the **instructions** that should be followed



Due May 16, 2022 11:59 PM

Instructions

Part(1)

Write a Python program that achieve the following goals:

1) Load your dataset

2) Apply data cleaning based on your dataset needs

- missing handling
- Remove noise
- Remove duplicate records
- Remove irrelevant attributes
- Remove correlated attributes
  . Correlation rate greater than or equal 0.8 for positive correlation
  . Correlation rate less than or equal -0.8 for negative correlation
- . Apply discretization on numeric attributes as possible

you Must add enough comments on your source code to identify and justify each of your data preparing steps.

Part(2)

3) Split your dataset into training and testing sets
   - 80% and 20% for training and testing sets respectively

and save each of these sets into separated files

use the following classifiers :

- KNN
- Decision Tree
- Naïve Bayes

4) Depend on your previous study on clustering technique (K means) use this technique on a suitable dataset

The code is written in python, and we have chosen data from Kaggle www.kaggle.com

Now it's time to show our steps:

Before Cleaning step

- Choosing Our Data:

We have chosen data called
"Rain in Australia" its link is (www.kaggle.com/datasets/jsphyg/weatherdataset-rattle-package)

About this Data

This dataset contains about 10 years of daily weather observations from numerous Australian weather stations.

RainTomorrow is the target variable to predict. It means -- did it rain the next day, Yes or No?
This column is Yes if the rain for that day was 1mm or more.

- Import the libraries which will be used:

One of the best advantages of Python is it has enormous number of libraries which can be used in different fields as Machine learning, Web development, Automation or scripting, Software testing and prototyping, Everyday tasks, Data analysis and Data Mining

Let's look at the image(1)

We have imported the following libraries (sys, pandas, NumPy, matplotlib and seaborn), they are not the only libraries here there are more down. We will talk about each library briefly and what it does in our code.

* sys library its abbreviation is System-specific parameters and functions here it is used to with the last three lines to show the hole datasets columns in order.

* Pandas library one of the most usable libraries in python for using datasets in python , it is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

* NumPy library **is** a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with

Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. And we will use it here to help showing our data.
* Matplotlib library easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc. And we will use it to make awesome visualization shapes.
* Seaborn library is a library mostly used for statistical plotting in Python. It is built on top of Matplotlib and provides beautiful default styles and color palettes to make statistical plots more attractive.
- The purpose of the last two lines as we have mentioned to show the all data set as we will see in image (3)

- Now we will load up our data set on jupyter notebook: as shown in image (2)

```
9  load our dataset

[895]: # load data set which it about rain in australia
       data = pd.read_csv('G:\\rain data\\weatherAUS.csv')
       # print first 5 row in data
       data.head()
```

2

[895]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN

1

- From pandas library we used function called **read_csv()** which load our dataset if it was csv file or we can use **read_excel()** instead after loading up our dataset that we named it "**data**" we need to show it so we write the second line of code **data.head()** which display the first 5 rows in each column in the dataset as show in image(3) .

2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN

	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	\
0	W	44.0	W	WNW	20.0	
1	WNW	44.0	NNW	WSW	4.0	
2	WSW	46.0	W	WSW	19.0	
3	NE	24.0	SE	E	11.0	
4	W	41.0	ENE	NW	7.0	

3

	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	\
0	24.0	71.0	22.0	1007.7	1007.1	8.0	
1	22.0	44.0	25.0	1010.6	1007.8	NaN	
2	26.0	38.0	30.0	1007.6	1008.7	NaN	
3	9.0	45.0	16.0	1017.6	1012.8	NaN	
4	20.0	82.0	33.0	1010.8	1006.0	7.0	

	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
0	NaN	16.9	21.8	No	No
1	NaN	17.2	24.3	No	No
2	2.0	21.0	23.2	No	No

*As we have seen in original dataset the data is full with NaN values , irrelevant data and categorical data which we won't be in need so after some steps of showing our data we will start **Data Cleaning** process*

- And also in the second cell we want to see our labels of columns so we wrote **data.columns** which make a list the labels of the datasets stored in it as shown in image(4).

```
[896]: # show columns name in data
       data.columns

[896]: Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
              'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
              'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
              'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
              'Temp3pm', 'RainToday', 'RainTomorrow'],
             dtype='object')

[897]: # take copy of data
       df=data
```

4

- In the same image we have copied the original dataset and make another dataset called "**df**" which we will continue the process on it

-And here is the start of the cleaning process

As we have seen in pervious images of the original data, we have a big amount of data that we don't need and may cause incorrect calculations, bias, and irrelevant data may be considered as metadata that can be removed without any side effects for our datasets and other columns .

So, to start Data Cleaning we should look at these specific data:

-Duplicate or noisy Data

-**Missing Data**

-Outliers

And we will see each of this process in the following screenshots

Image (5): here we had a quick look to see the each column and its data type and we have written df.info() for that line of code and the rest of the outputs is in image (6)

```
11  handle missing value

[899]: # get info about data like dtype
       df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 22 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Location       145460 non-null  object
 1   MinTemp        143975 non-null  float64
 2   MaxTemp        144199 non-null  float64
 3   Rainfall       142199 non-null  float64
 4   Evaporation    82670 non-null   float64
 5   Sunshine       75625 non-null   float64
 6   WindGustDir    135134 non-null  object
 7   WindGustSpeed  135197 non-null  float64
 8   WindDir9am     134894 non-null  object
 9   WindDir3pm     141232 non-null  object
 10  WindSpeed9am   143693 non-null  float64
```

5

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 22 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Location       145460 non-null  object
 1   MinTemp        143975 non-null  float64
 2   MaxTemp        144199 non-null  float64
 3   Rainfall       142199 non-null  float64
 4   Evaporation    82670 non-null   float64
 5   Sunshine       75625 non-null   float64
 6   WindGustDir    135134 non-null  object
 7   WindGustSpeed  135197 non-null  float64
 8   WindDir9am     134894 non-null  object
 9   WindDir3pm     141232 non-null  object
 10  WindSpeed9am   143693 non-null  float64
 11  WindSpeed3pm   142398 non-null  float64
 12  Humidity9am    142806 non-null  float64
 13  Humidity3pm    140953 non-null  float64
 14  Pressure9am    130395 non-null  float64
 15  Pressure3pm    130432 non-null  float64
 16  Cloud9am       89572 non-null   float64
 17  Cloud3pm       86102 non-null   float64
 18  Temp9am        143693 non-null  float64
 19  Temp3pm        141851 non-null  float64
 20  RainToday      142199 non-null  object
 21  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(6)
memory usage: 24.4+ MB
```

6

Calculate the number of null and nan value in image(7): by using the df.isna().sum()

```
[900]: # check the number of na in data
       df.isna().sum()

[900]: Location          0
       MinTemp        1485
       MaxTemp        1261
       Rainfall       3261
       Evaporation   62790
       Sunshine      69835
       WindGustDir   10326
       WindGustSpeed 10263
       WindDir9am    10566
       WindDir3pm     4228
       WindSpeed9am   1767
       WindSpeed3pm   3062
       Humidity9am    2654
       Humidity3pm    4507
       Pressure9am   15065
       Pressure3pm   15028
       Cloud9am      55888
       Cloud3pm      59358
       Temp9am        1767
       Temp3pm        3609
       RainToday      3261
       RainTomorrow   3267
       dtype: int64
```

7

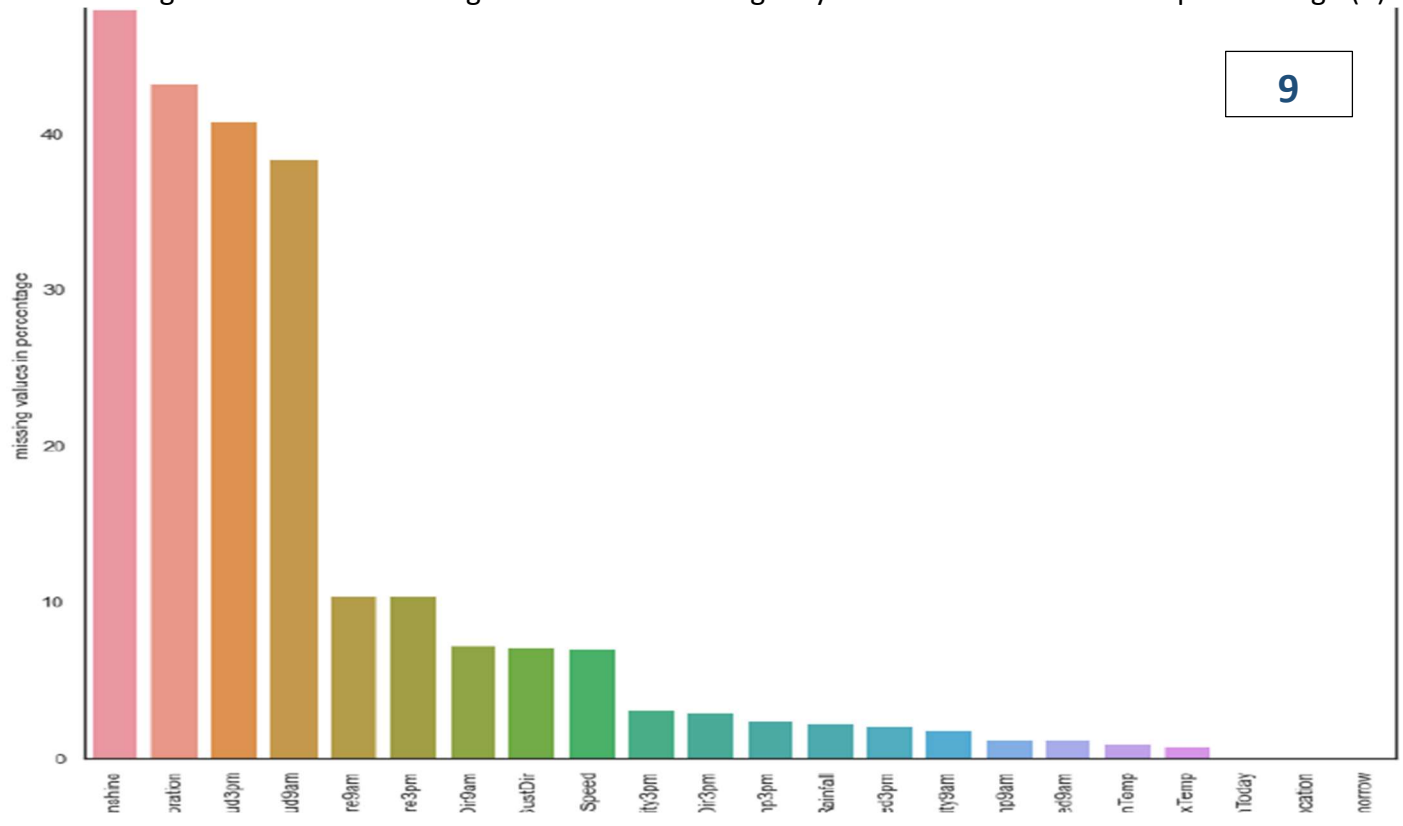
Now we will visualize it as show in image (8):

```
import plotly.express as px
# missing values
missing_value = 100 * df.isnull().sum()/len(df)
missing_value = missing_value.reset_index()
missing_value.columns = ['variables','missing values in percentage']
missing_value = missing_value.sort_values('missing values in␣
 ↪percentage',ascending=False)

# barplot
plt.figure(figsize=(12,12))
g = sns.barplot(x = missing_value['variables'], y = missing_value['missing␣
 ↪values in percentage'])
g.set_xticklabels(g.get_xticklabels(),rotation = 90)
plt.show()
```

8

As we have seen we import library called plotly.express which is an open-source library that can be used for data visualization and understanding data simply and easily. Plotly supports various types of plots like line charts, scatter plots, histograms, cox plots, etc. So you all must be wondering why Plotly over other visualization tools or libraries? Here's the answer –

Plotly has hover tool capabilities that allow us to detect any outliers or anomalies in a large number of data points. It is visually attractive that can be accepted by a wide range of audiences.
It allows us for the endless customization of our graphs that makes our plot more meaningful and understandable for others.
-After defining the axis and set the fig size and write the right syntax here we have the output in image (9)



9

-Know let's remove some columns: by using df.drop('column name',axis=)

First dropping the column of 'Date' as it is not necessary for calculations later

10 drop irrelevant data

[898]:
```
# drop data from data because it don't help us
df=df.drop(['Date'],axis=1)
```

10

Note : We will deal with Measures of Central tendency(mean , median, mode) & Dispersion of Data (mode, max, min, outliers ,box plot) to more clear visualization and detect outliers

- In image (8) we have done 3 tasks

-we removed columns ('Evapuration','Sunshine','Cloud9am','Cloud3pm') as their unknown values are to much

- we remove null values in targeted columns

-the last thing is handling null and empty values with mode -we can use mean or median instead- of the column

[901]:
```
# drop Evaporation,Sunshine,Cloud9am and Cloud3pm because the na is more than␣
 ↪the half
df =df.drop(['Evaporation','Sunshine','Cloud9am','Cloud3pm'],axis=1)
# make list to catagorical,numrical and other columns
cat_cols = []
num_cols = []
other_cols = []
# check which columns are catagorical or numrical or other
for i in df.columns:
    if df[i].dtype == "object":
        cat_cols.append(i)
    elif df[i].dtype == "float64":
        num_cols.append(i)
    else:
        other_cols.append(i)
# drop na from our target because we don't want any problem in train and test ␣
 ↪
df=df.dropna(axis=0,subset=['RainTomorrow','RainToday'])
# fill catagorical columns with mode
for i in cat_cols:
    df[i].fillna(value=df[i].mode()[0],inplace=True)
# fill numrical columns with median
for k in num_cols:
    df[k].fillna(value=df[k].median(),inplace=True)
```

11

After handling nan values and filling it with an appropriate data(mode) will display the summation of na to see the difference as image (12) show

```
[902]: # check if there is na after remove
       df.isna().sum()
```

```
[902]: Location        0
       MinTemp         0
       MaxTemp         0
       Rainfall        0
       WindGustDir     0
```

12

```
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
```

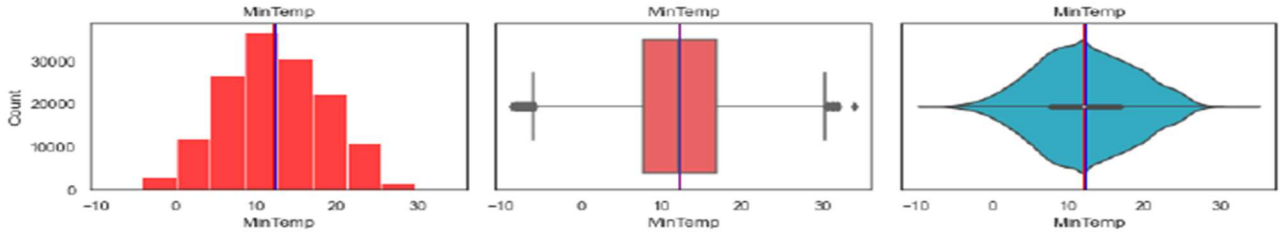
**We still in the step of Navigating and Cleaning of Data **

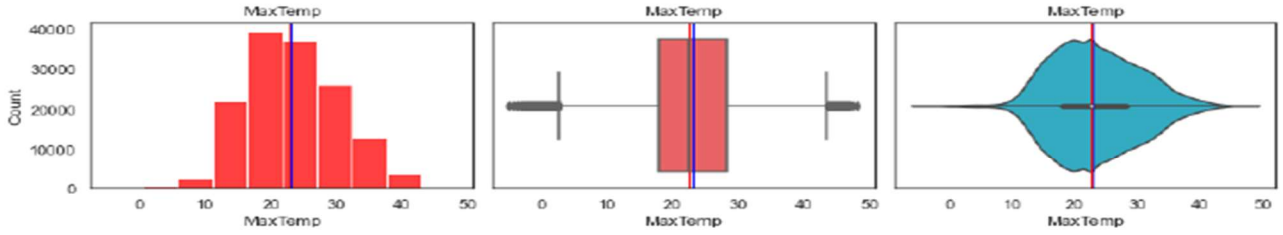
After getting rid of null values, we want to visualize the data to go to next step which is removing outliers
The best shapes to show numerical data are (histogram, boxplot and barplot)
As seen in the next image (13):

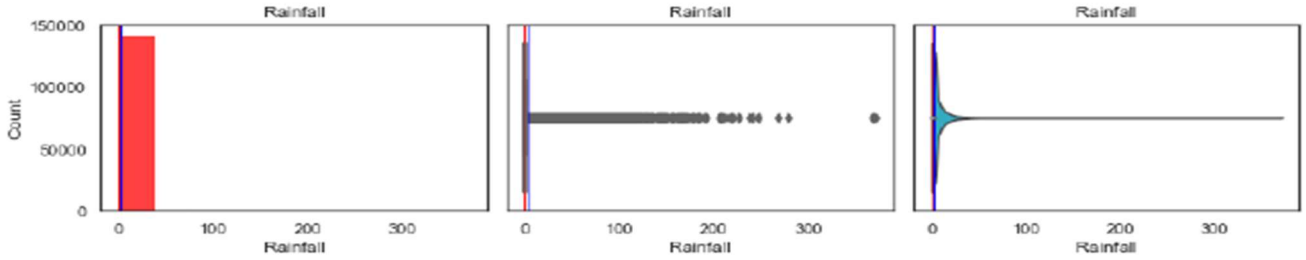
13

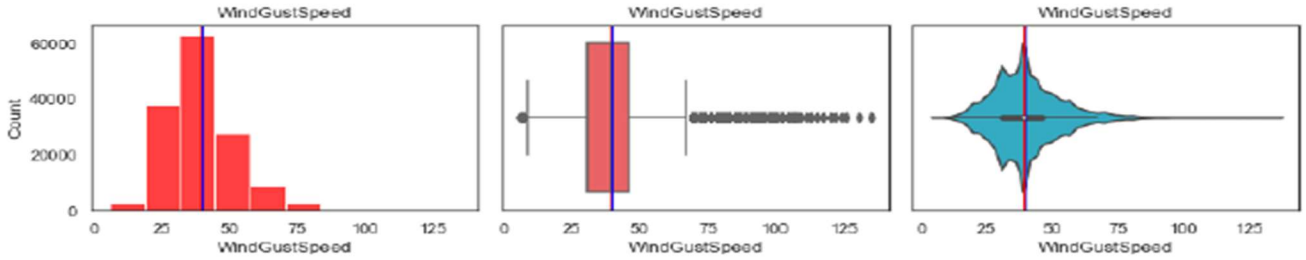
```
# plot histplot and boxplot and violinplot to all numrical features to know
 ↪skewness
for i in num_cols:

    fig, ax = plt.subplots(1,3,figsize=(12, 3))

    sns.set_style('white')
    sns.set_context(context = 'notebook',font_scale=1)

    sns.histplot(x=df[i],bins=10,color='red',kde=False,ax=ax[0]);
    sns.boxplot(x=df[i], ax = ax[1],color='#ff4e50');
    sns.violinplot(x=df[i],bins=10,color='#1ebbd9',ax=ax[2]);

    ax[0].title.set_text(i);
    ax[1].title.set_text(i);
    ax[2].title.set_text(i);

    ax[0].axvline(df[i].mean(), color='b', linewidth=1)
    ax[1].axvline(df[i].mean(), color='b', linewidth=1)
    ax[2].axvline(df[i].mean(), color='b', linewidth=1)

    ax[0].axvline(df[i].median(), color='r', linewidth=1)
    ax[1].axvline(df[i].median(), color='r', linewidth=1)
    ax[2].axvline(df[i].median(), color='r', linewidth=1)

    plt.tight_layout()
```

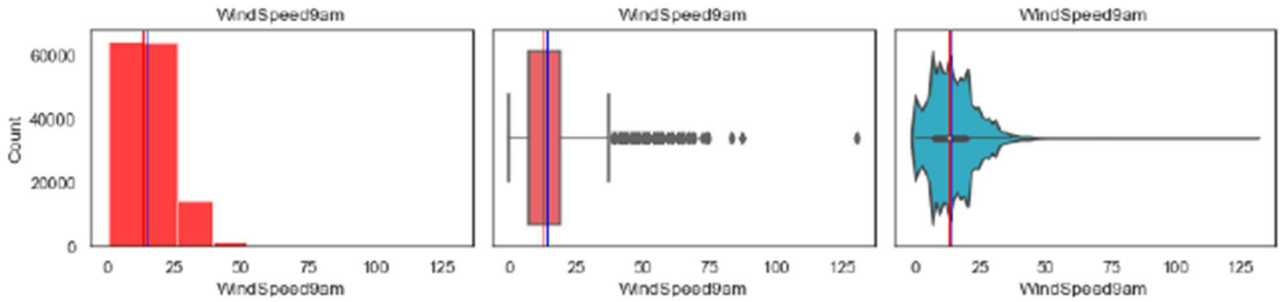
As we have seen in the for loop it loops over the columns of df and get it's histogram, boxplot and violin plot in the next images we will see the great output of this code in the following images

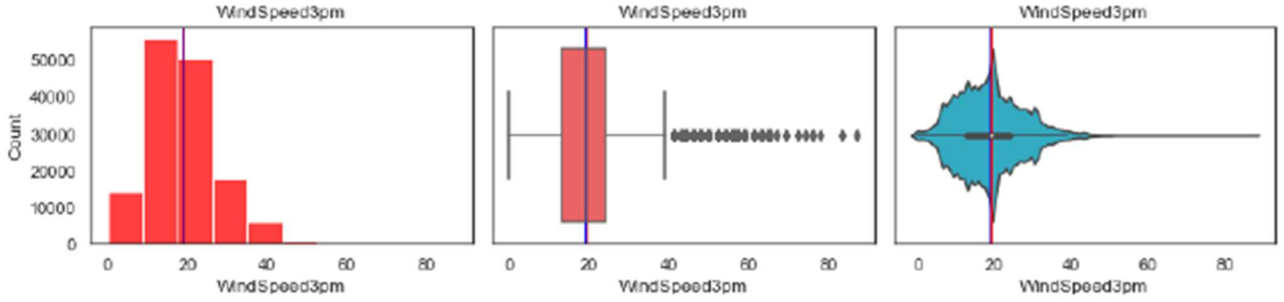


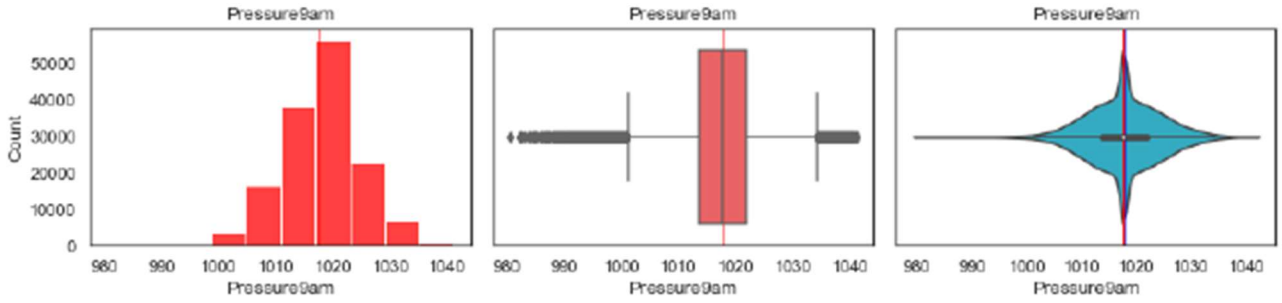


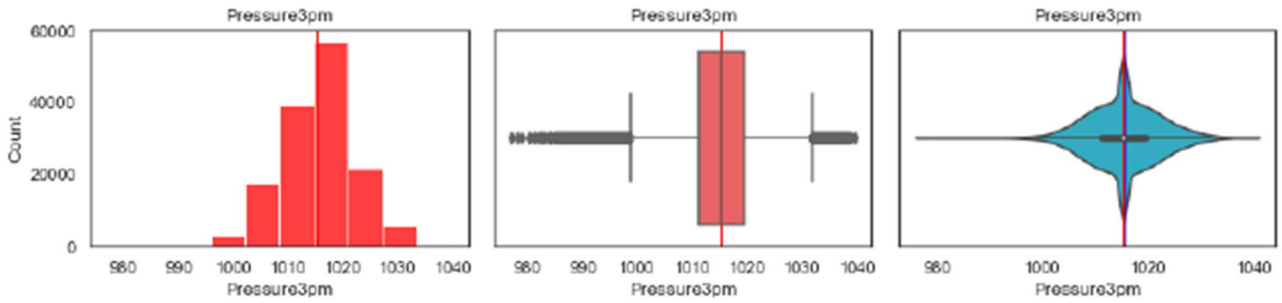


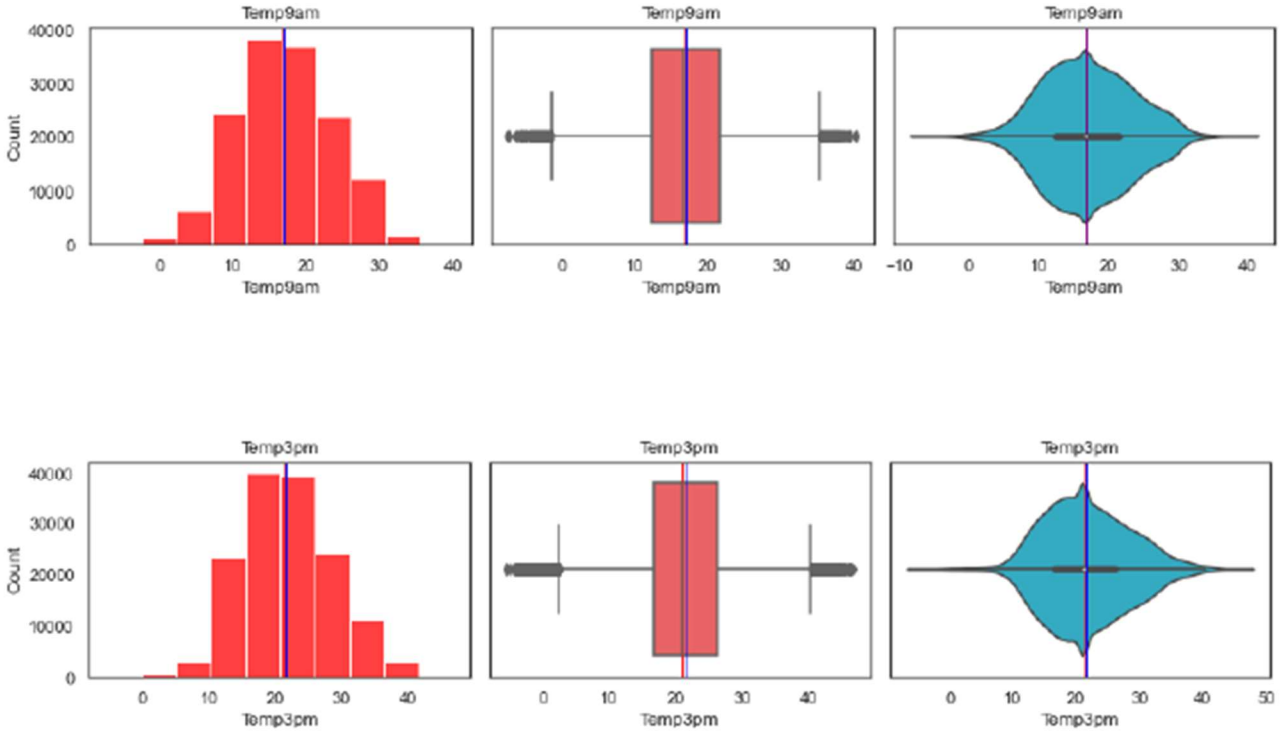












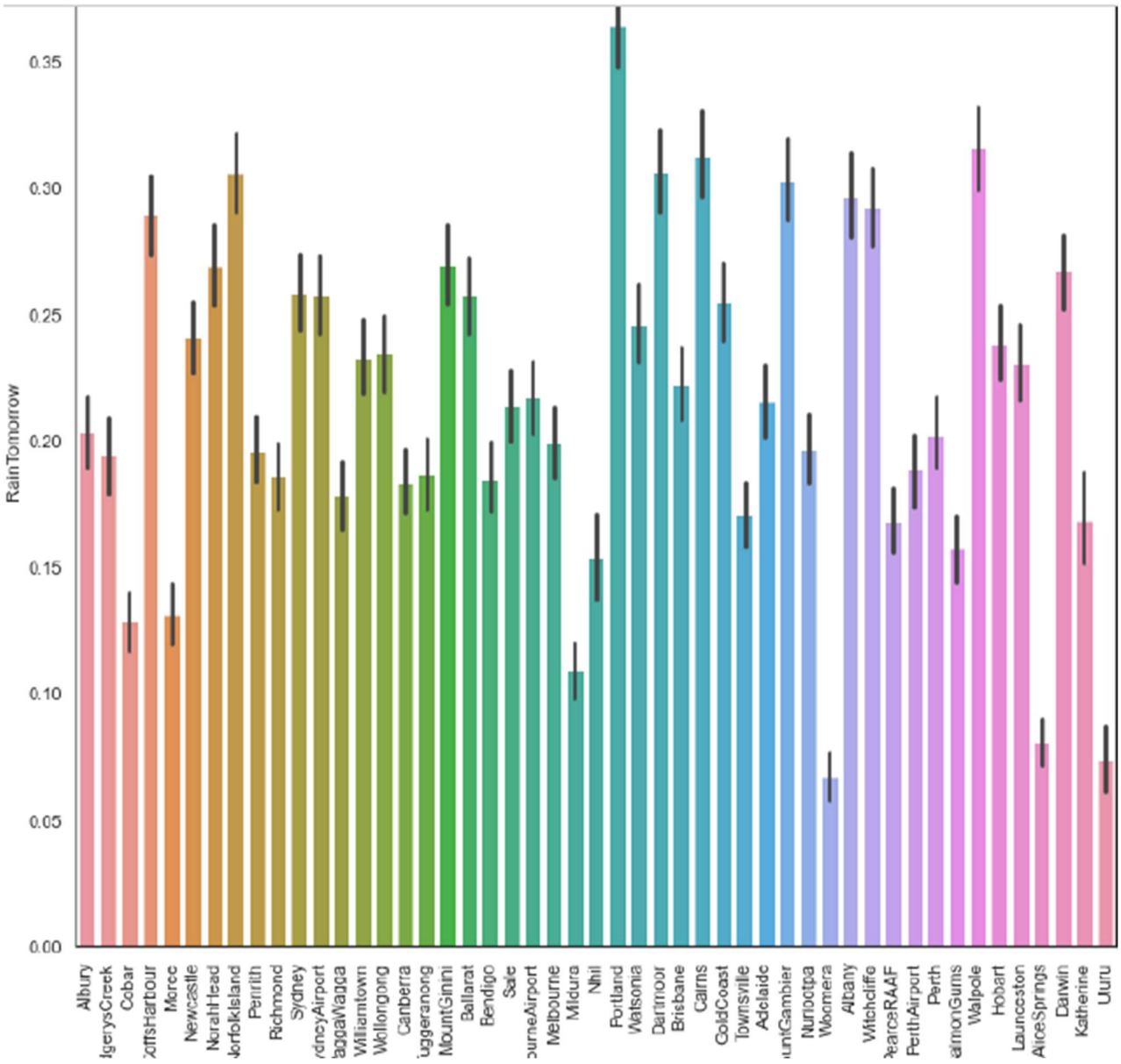
As we have seen this photo, we recognize a great number of outliers and most of the data are normally distributed, so we will visualize categorical data then remove outliers and duplicate data

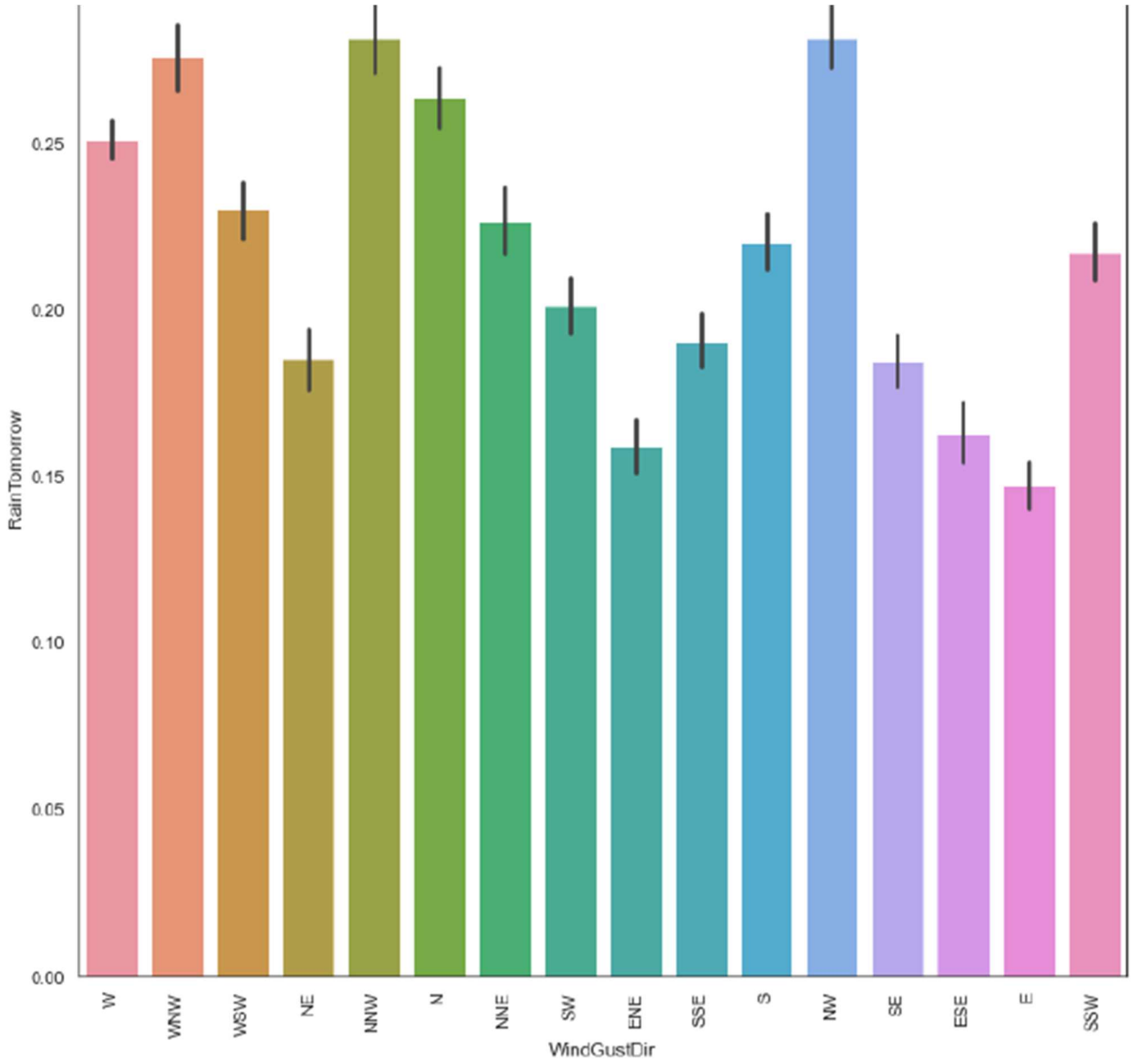
In the following image (14) we will visualize the categorical data

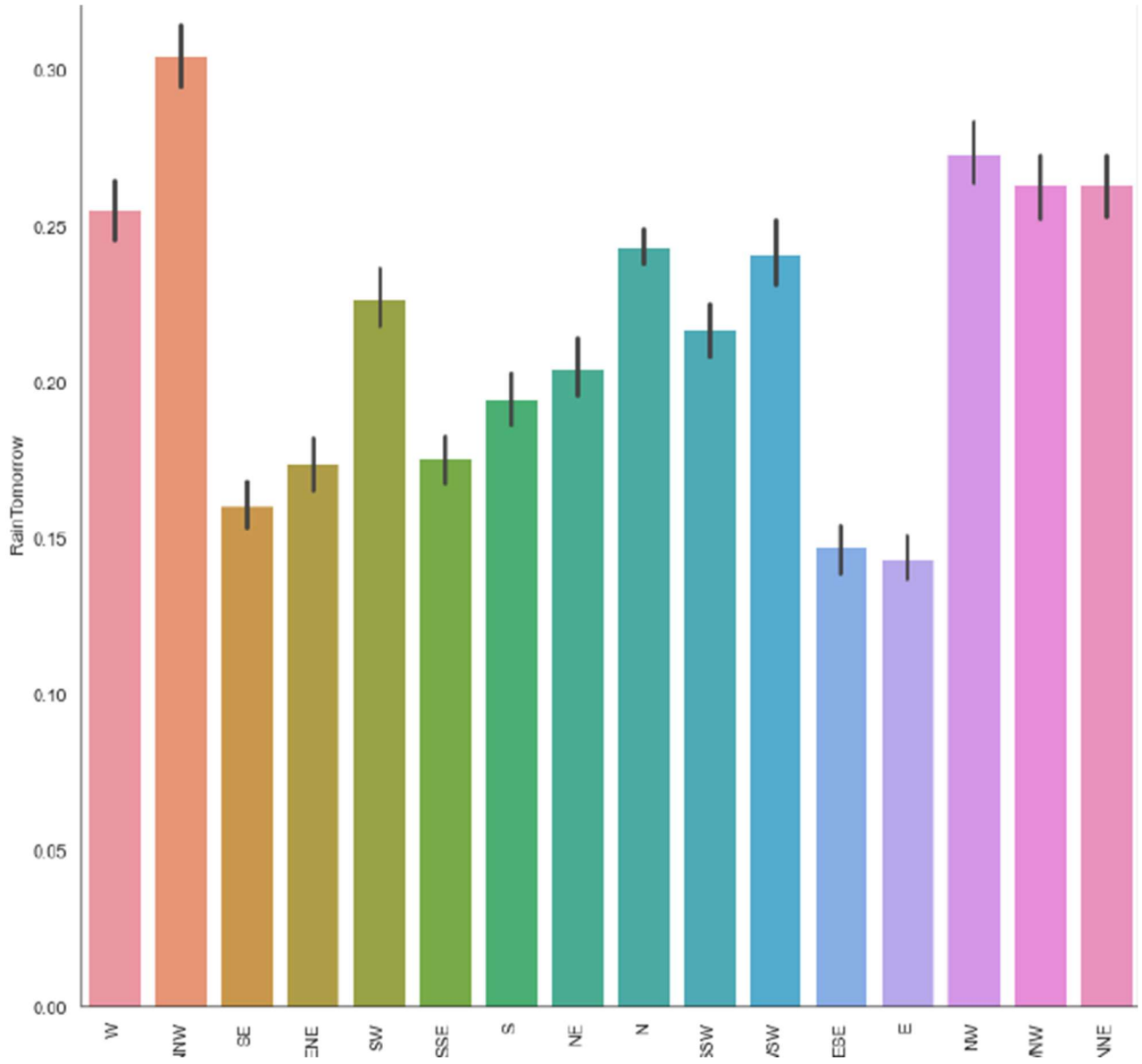
14

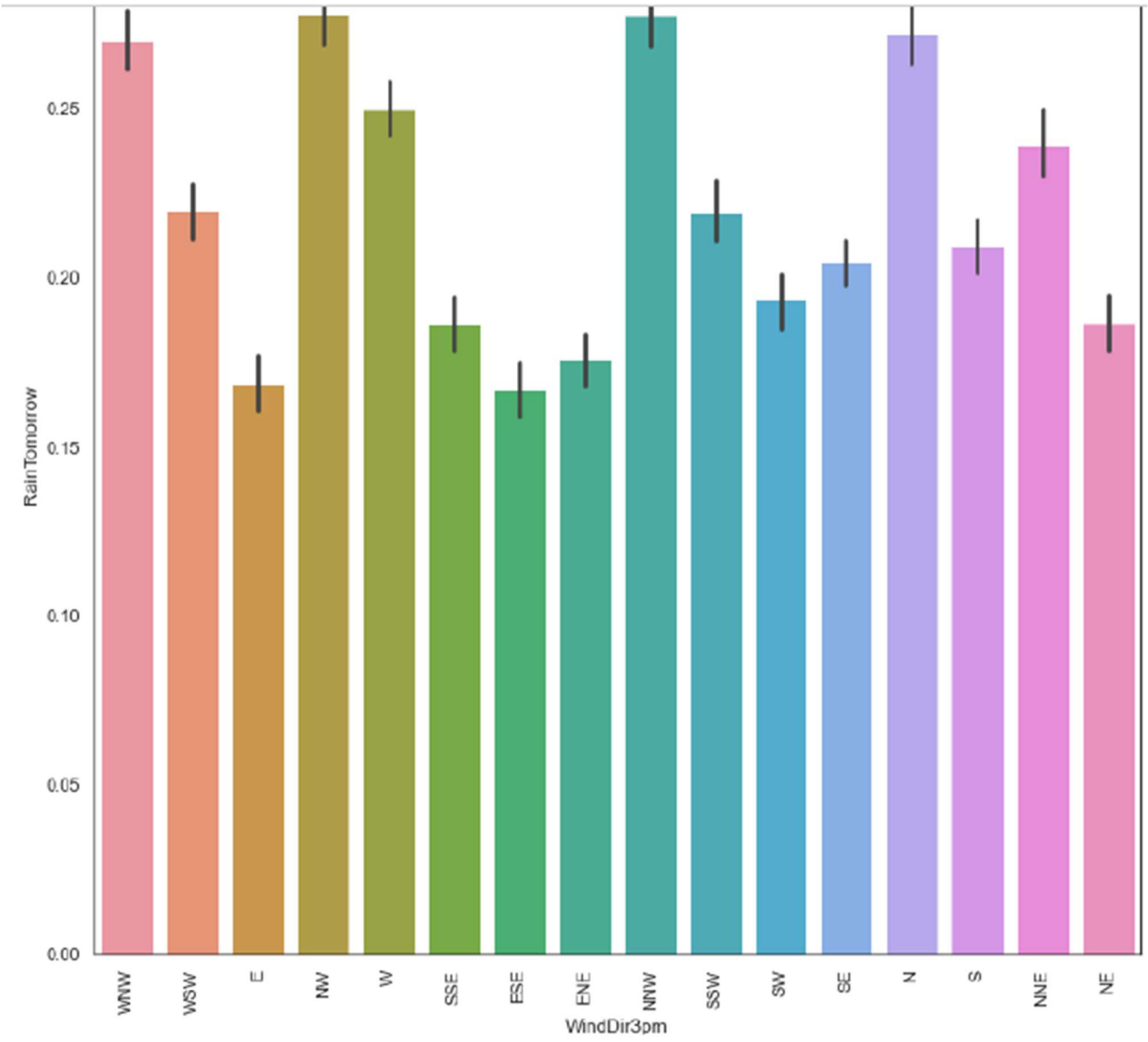
```
# Function for visualization of categorical features
def ctgplt(variable,to):
    f, ax = plt.subplots(figsize = (12,12))
    g = sns.barplot(x = variable, y = to, data = df)
    g.set_xticklabels(g.get_xticklabels(),rotation = 90)
    plt.show()
```

So here we use histogram cause it is the most suitable shape to describe the frequency of data we create a function that create a counter for each unique element in the column and show it near the targeted column as we will see in the following images

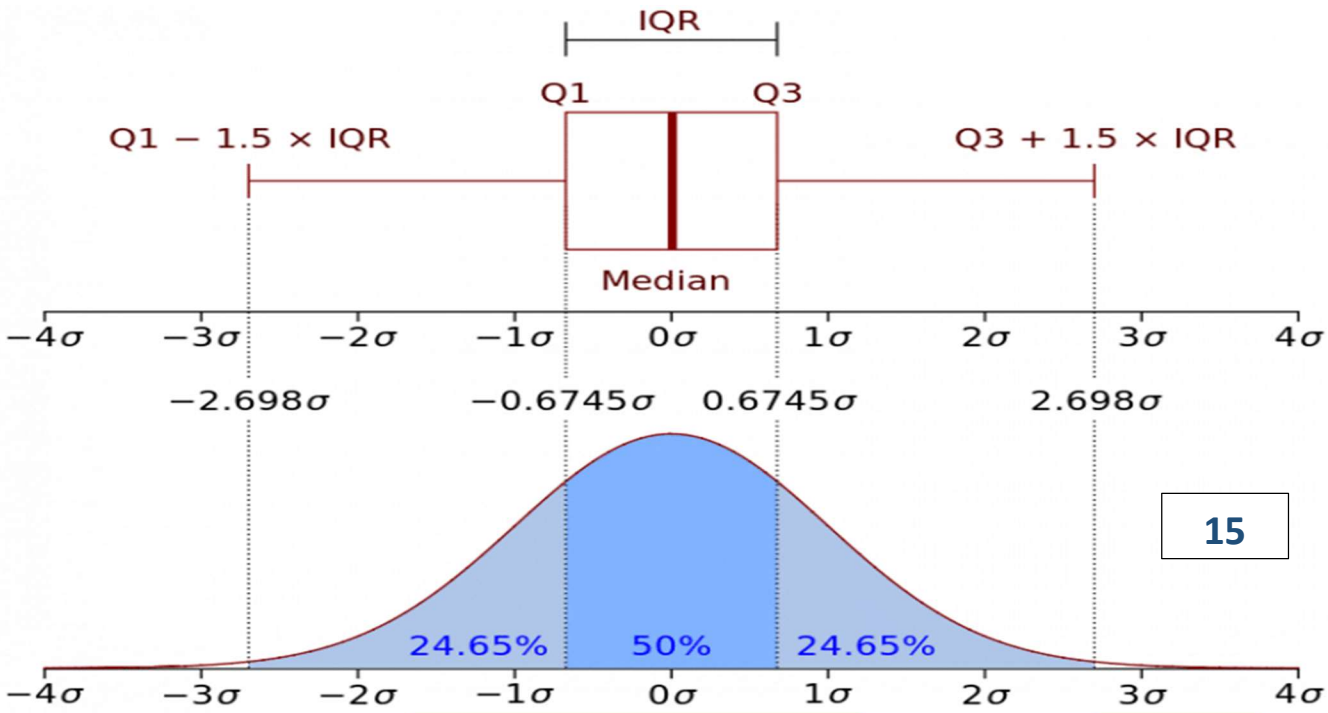








After seeing the visualized data and see this number of **outliers** it time for detection and removing of it if you want to see more about outliers see the link below on freeCodeCamp
(www.freecodecamp.org/news/what-is-an-outlier-definition-and-how-to-find-outliers-in-statistics)
And before the code let's have a look on the method of removing it image (15)



We get 5-number summary elements (mean, min, Q1, median, Q3, max) after that we get IQR = Q3-Q1 then we declare L(lower) = (Q1-1.5*IQR) & U(upper)=(Q3+1.5*IQR) and then filter the elements as every element greater than U and every element smaller than L will be refused and considered as outlier

In the following image(16,17) we will see the function of getting index of outliers and function for deleting it

```
[552]: # method to get the index of outliers
       def outliers(data,feature):
           Q1= data[feature].quantile(0.25)
```



```
    Q3= data[feature].quantile(0.75)
    IQR= Q3-Q1
    lower= Q1-1.5*IQR
    upper= Q3+1.5*IQR
    outlier= data.index[(data[feature]<lower) | (data[feature]>upper)]
    return outlier
```

[553]:
```
# save the index or outliers in list named outlier
outlier=[]
for feature in num_cols:
    outlier.extend(outliers(df,feature))
```

17

[554]:
```
# method to remove the outlier
def remove(data,outlier):
    outlier=sorted(set(outlier))
    data = data.drop(outlier)
    return data
```

[555]:
```
# remove outlier from the data
df =remove(df,outlier)
```

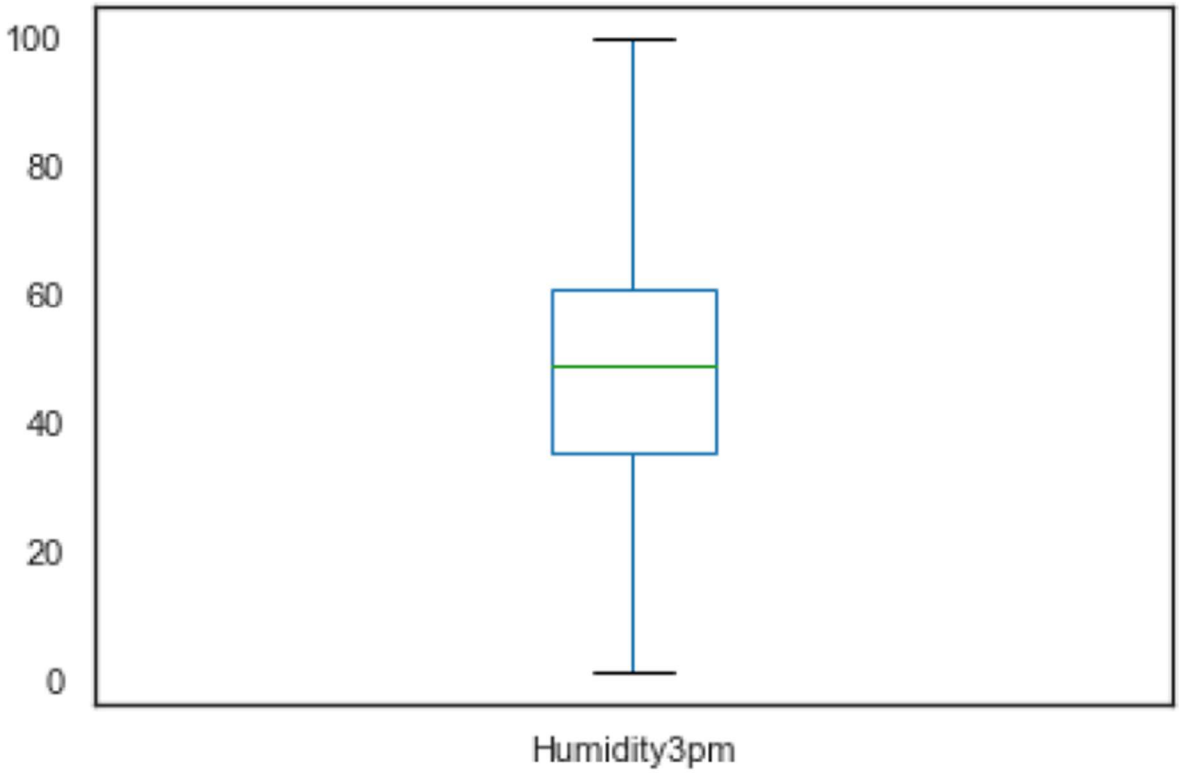
[556]:
```
# get number of rows afer remove the outliers
df.shape
```

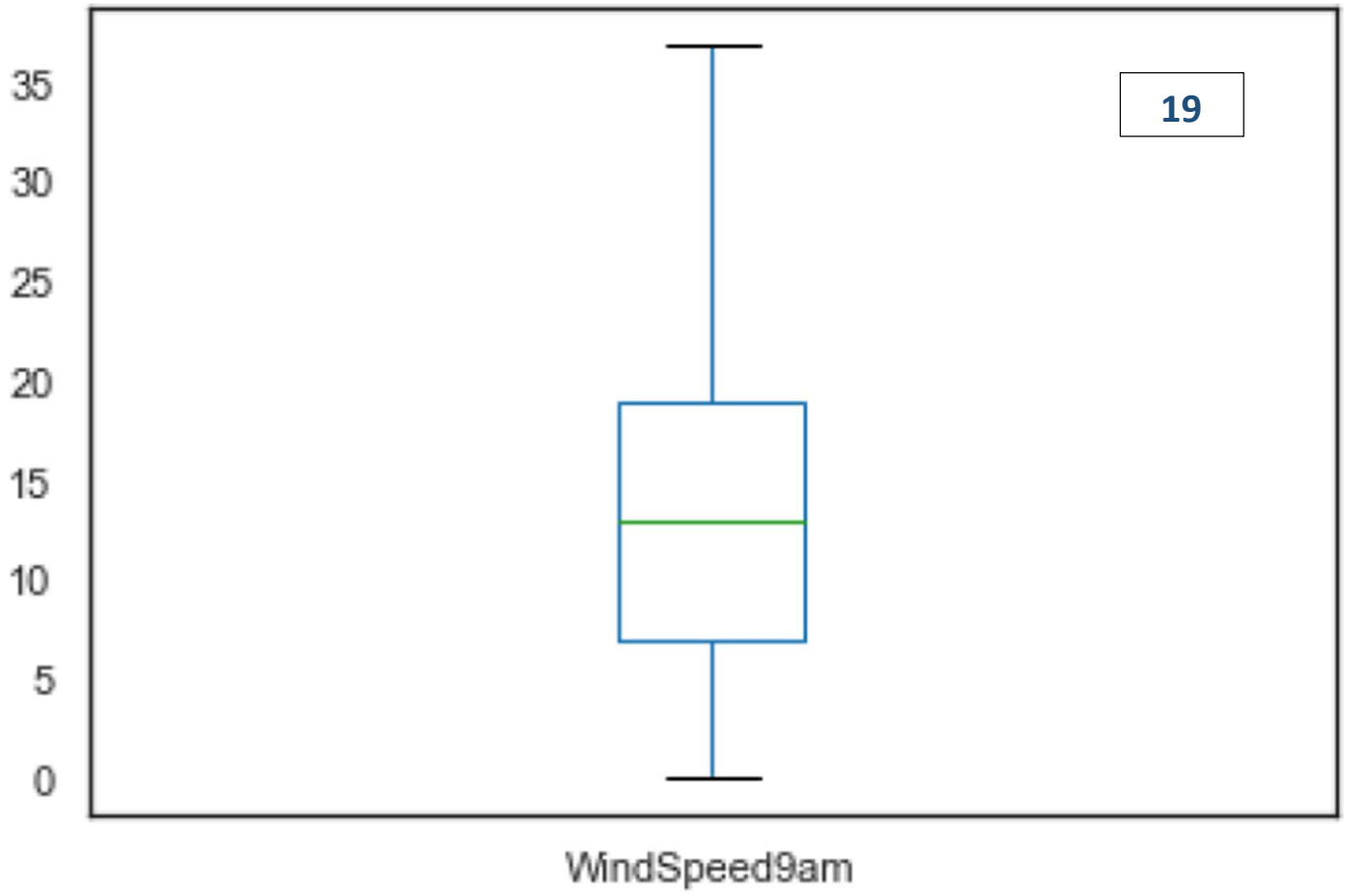
[556]: (108029, 18)

-We have removed outliers and we will see the effect in the following images(18,19,20,21)

```
# box plot after remove outliers
box_plot(df,'Humidity3pm')
box_plot(df,'WindSpeed9am')
```

18





19

Know we are in the last steps of Data Cleaning process as we see in this image (20)

20

```
# change a value of RainTomorrow and RainToday to 1 if = Yes and 0 if = No
df["RainTomorrow"] = [1 if each == "Yes" else 0 for each in df["RainTomorrow"]]
df["RainToday"] = [1 if each == "Yes" else 0 for each in df["RainToday"]]
```

Here to avoid types error we transform data in targeted columns to numbers to classify it later

[557]:
```
# get the count and and mean and 5 number summary of each column after remove
 ↪the outliers
df.describe()
```

[557]:

21

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	\
count	108029.000000	108029.000000	108029.000000	108029.000000	
mean	11.963318	23.891871	0.102784	37.525766	
std	6.347131	6.685311	0.271186	10.442607	
min	-5.900000	2.800000	0.000000	9.000000	
25%	7.400000	18.900000	0.000000	30.000000	
50%	11.900000	23.400000	0.000000	37.000000	
75%	16.500000	28.700000	0.000000	44.000000	
max	30.200000	43.500000	1.500000	67.000000	

	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	\
count	108029.000000	108029.000000	108029.000000	108029.000000	
mean	12.949736	17.704820	66.633348	48.308389	
std	7.868914	7.718722	17.583896	18.916048	
min	0.000000	0.000000	18.000000	1.000000	
25%	7.000000	13.000000	55.000000	35.000000	
50%	13.000000	17.000000	67.000000	49.000000	
75%	19.000000	22.000000	79.000000	61.000000	
max	37.000000	39.000000	100.000000	100.000000	

22

	Pressure9am	Pressure3pm	Temp9am	Temp3pm	\
count	108029.000000	108029.000000	108029.000000	108029.000000	
mean	1018.488475	1015.836213	17.173619	22.358104	
std	5.914551	5.924498	6.281269	6.441343	
min	1001.100000	998.700000	-1.400000	2.500000	
25%	1014.800000	1012.000000	12.700000	17.700000	
50%	1017.600000	1015.200000	16.900000	21.700000	
75%	1022.400000	1019.700000	21.600000	26.800000	
max	1034.200000	1031.800000	35.100000	40.400000	

	RainToday	RainTomorrow
count	108029.000000	108029.000000
mean	0.024688	0.149154
std	0.155173	0.356242
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1.000000	1.000000

Know it's time to remove the correlated columns:

First let's talk about correlation

What is correlation?

Correlation is a way to determine if two variables in a dataset are related in any way. Correlations have many real-world applications. We can see if using certain search terms are correlated to views on YouTube. Or we can see if ads are correlated to sales. When building machine learning models correlations are an important factor in determining features. Not only can this help us to see which features are linear related, but if features are strongly correlated, we can remove them to prevent duplicating information.

How do we measure correlation?

In data science we can use the r value, also called Pearson's correlation coefficient. This measures how closely two sequences of numbers(i.e., columns, lists, series, etc.) are correlated.

The r value is a number between -1 and 1. It tells us whether two columns are positively correlated, not correlated, or negatively correlated. The closer to 1, the stronger the positive correlation. The closer to -1, the stronger the negative correlation (i.e., the more "opposite" the columns are). The closer to 0, the weaker the correlation.

And here is the general formula of correlation.

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

for more information about correlation copy

(towards data science simple correlation with pandas and numpy)

Now let's see the code ----->

note: we have dropped duplicates before doing the detection

First we need to detect the correlation between the columns so we have used **df.corr()** and store it in a variable called **cor** then we showed it in a heatmap as seen in images(23,24)

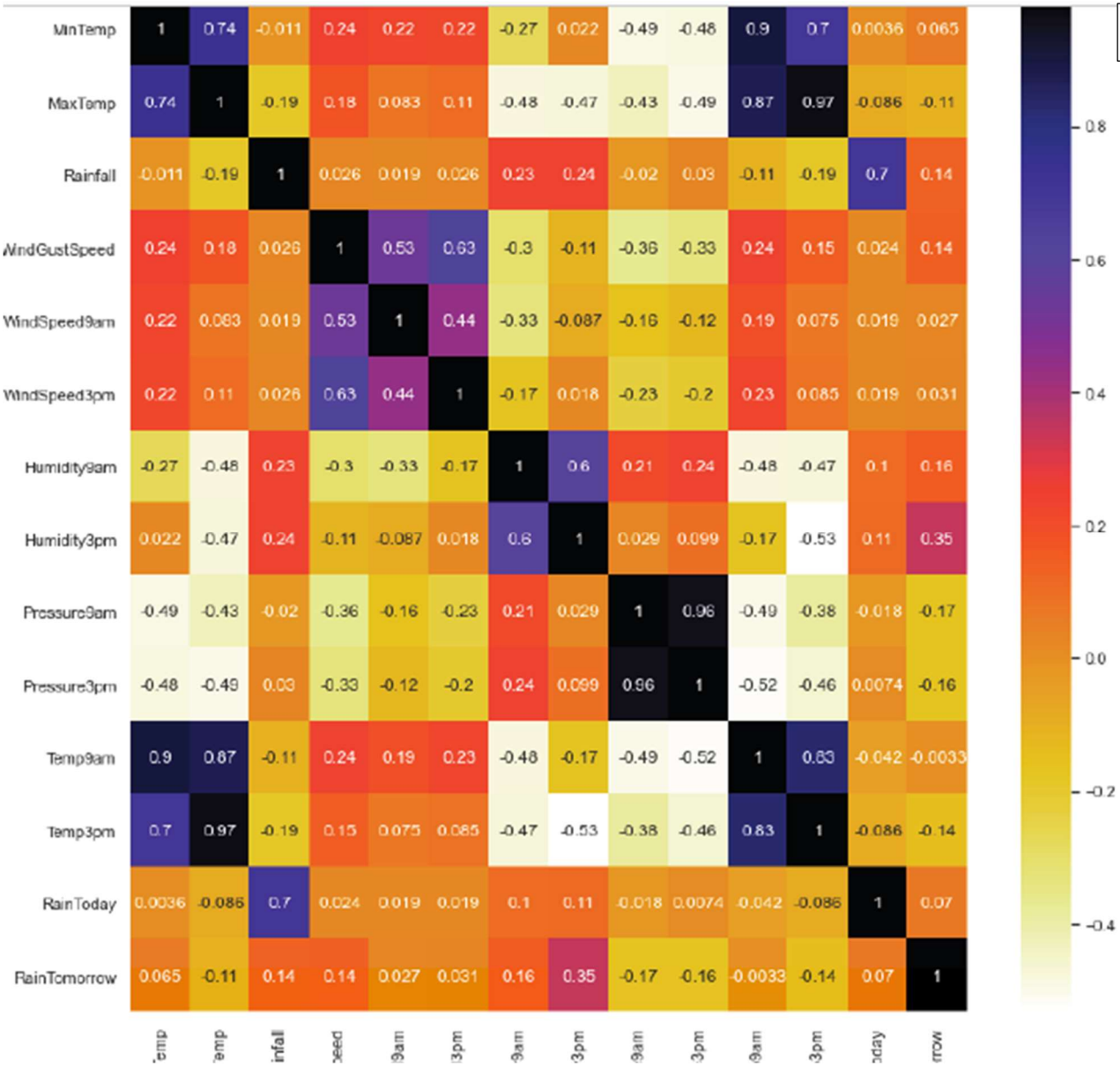
14 drop duplicates data

23

```
df=df.drop_duplicates()
```

15 feature selection using correlation

```
# get correlation between columns and represent as heat map
feature=df[num_cols]
cor=df.corr()
plt.figure(figsize=(13,13))
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
```



Know after we have seen correlated columns, we should remove them.
As shown in the terms of project the correlation (cor) must be (0.8 > cor> -.08)
So our measure of removal will be as shown.
We will remove cor greater than 0.8 and lesser than -0.8 and we will see it that we will loop on each column and remove the columns which exceeds the condition in the next images (25,26,27)

```
# method to get the columns which correlation is bigger than or equal the rate␣
 ↪or smaller than or equal -rate
def correlation(data,rate):
    col_cor=set()
    cor=data.corr()
    for i in range(len(cor.columns)):
        for j in range(i):
            if((cor.iloc[i,j])>=rate) | ((cor.iloc[i,j])<=-rate) :
                col_cor.add(cor.columns[i])
    return col_cor
```

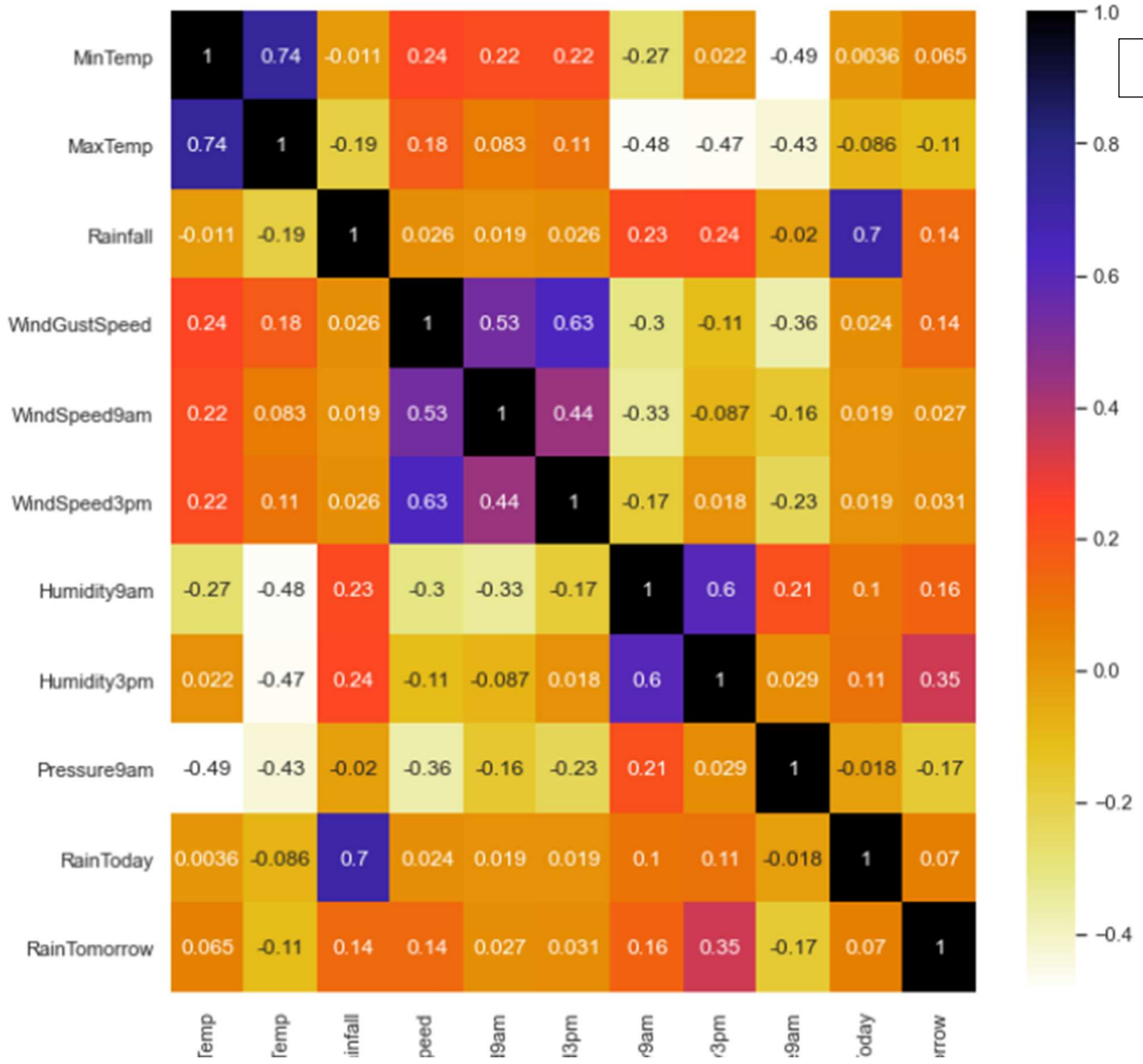
25

```
#get the columns which correlation is bigger than or equal 0.8 or smaller than␣
 ↪or equal -0.8
feature=df[num_cols]
corr_fet=correlation(feature,0.8)
print(corr_fet)
#drop these columns
df=df.drop(corr_fet,axis=1)
```

26

```
{'Pressure3pm', 'Temp9am', 'Temp3pm'}
```

```
# correlation heat plot after drop feature bigger than or equal 0.8 or smaller␣
 ↪than or equal -0.8
cor = df.corr()
plt.figure(figsize=(10,10))
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
```



Here we will see the first 5 columns in our dataset in images(28,29)

```
# show first 5 rows of data after feature selection
df.head()
```

28

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am
0	Albury	13.4	22.9	0.6	W	44.0	W
1	Albury	7.4	25.1	0.0	WNW	44.0	NNW
2	Albury	12.9	25.7	0.0	WSW	46.0	W
3	Albury	9.2	28.0	0.0	NE	24.0	SE
4	Albury	17.5	32.3	1.0	W	41.0	ENE

	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	\
0	WNW	20.0	24.0	71.0	22.0	

29

1	WSW	4.0	22.0	44.0	25.0
2	WSW	19.0	26.0	38.0	30.0
3	E	11.0	9.0	45.0	16.0
4	NW	7.0	20.0	82.0	33.0

	Pressure9am	RainToday	RainTomorrow
0	1007.7	0	0
1	1010.6	0	0
2	1007.6	0	0
3	1017.6	0	0
4	1010.8	0	0

Before Balancing our data, we will discretize our data to look more cleaned and get rid of categorical data

So, what is discretization?

Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy. In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss. There are two forms of data discretization first is supervised discretization, and the second is unsupervised discretization. Supervised discretization refers to a method in which the class data is used. Unsupervised discretization refers to a method depending upon the way which operation proceeds. It means it works on the top-down splitting strategy and bottom-up merging strategy.

For more information see this link(https://www.javatpoint.com/discretization-in-data-mining)

So first we will discretize numerical data first then categorical data as seen in images(30,31)

[564]:
```
# split continuous to 9 bins
for i in␣
 ↪['MinTemp','MaxTemp','WindGustSpeed','WindSpeed9am','WindSpeed3pm','Humidity9am','Humidity3
 ↪
    df[i]=pd.qcut(x=df[i],q=9,labels=False)
```

30

[565]:
```
df.head()
```

[565]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	\
0	Albury	5	4	0.6	W	6	W	
1	Albury	2	5	0.0	WNW	6	NNW	
2	Albury	5	5	0.0	WSW	7	W	
3	Albury	3	6	0.0	NE	0	SE	
4	Albury	7	7	1.0	W	5	ENE	

	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	\
0	WNW	7	6	5	0	
1	WSW	0	6	0	1	
2	WSW	6	7	0	1	
3	E	3	0	0	0	
4	NW	1	5	7	1	

	Pressure9am	RainToday	RainTomorrow
0	0	0	0
1	0	0	0

17 change catagorical columns numrical

[566]:
```
from category_encoders import TargetEncoder
encoder = TargetEncoder()
```

31

[567]:
```
# get percent of every location value to RainTomorrow=1
df['Location_encoder']=encoder.
 ↪fit_transform(X=df['Location'],y=df['RainTomorrow'])
```

[568]:
```
# get percent of every WindGustDir value to RainTomorrow=1
encoder = TargetEncoder()
df['WGD_encoder']=encoder.
 ↪fit_transform(X=df['WindGustDir'],y=df['RainTomorrow'])
```

[569]:
```
# get percent of every WindDir9am value to RainTomorrow=1
encoder = TargetEncoder()
df['WD9_encoder']=encoder.fit_transform(X=df['WindDir9am'],y=df['RainTomorrow'])
```

[570]:
```
# get percent of every WindDir3pm value to RainTomorrow=1
encoder = TargetEncoder()
df['WD3_encoder']=encoder.fit_transform(X=df['WindDir3pm'],y=df['RainTomorrow'])
```

In the previous image we have used library called **TargetEncoder** and here what it does: For the case of categorical target: features are replaced with a blend of posterior probability of the target given particular categorical value and the prior probability of the target over all the training data. And know every categorical column has taken a numerical value which will ease the calculations And here is the output in image (32)

32

```
[571]: # print first 5 rows after preprocessing
       df.head()
```

[571]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am
0	Albury	5	4	0.6	W	6	W
1	Albury	2	5	0.0	WNW	6	NNW
2	Albury	5	5	0.0	WSW	7	W
3	Albury	3	6	0.0	NE	0	SE
4	Albury	7	7	1.0	W	5	ENE

	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
0	WNW	7	6	5	0
1	WSW	0	6	0	1
2	WSW	6	7	0	1
3	E	3	0	0	0
4	NW	1	5	7	1

	Pressure9am	RainToday	RainTomorrow	Location_encoder	WGD_encoder
0	0	0	0	0.145025	0.176415
1	0	0	0	0.145025	0.194627
2	0	0	0	0.145025	0.141605
3	3	0	0	0.145025	0.141836
4	0	0	0	0.145025	0.176415

	WD9_encoder	WD3_encoder

And then we will remove the original categorical columns as shown in image (33)

[572]:
```
# remove categorical columns
df=df.drop(['Location','WindGustDir','WindDir9am','WindDir3pm'],axis=1)
```

32

[573]:
```
df.head()
```

[573]:

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	\
0	5	4	0.6	6	7	6	
1	2	5	0.0	6	0	6	
2	5	5	0.0	7	6	7	
3	3	6	0.0	0	3	0	
4	7	7	1.0	5	1	5	

	Humidity9am	Humidity3pm	Pressure9am	RainToday	RainTomorrow	\
0	5	0	0	0	0	
1	0	1	0	0	0	
2	0	1	0	0	0	
3	0	0	3	0	0	
4	7	1	0	0	0	

	Location_encoder	WGD_encoder	WD9_encoder	WD3_encoder
0	0.145025	0.176415	0.159149	0.186651

after finishing the previous step, we almost done with data cleaning
so lets see the last step in the next images(33,34)

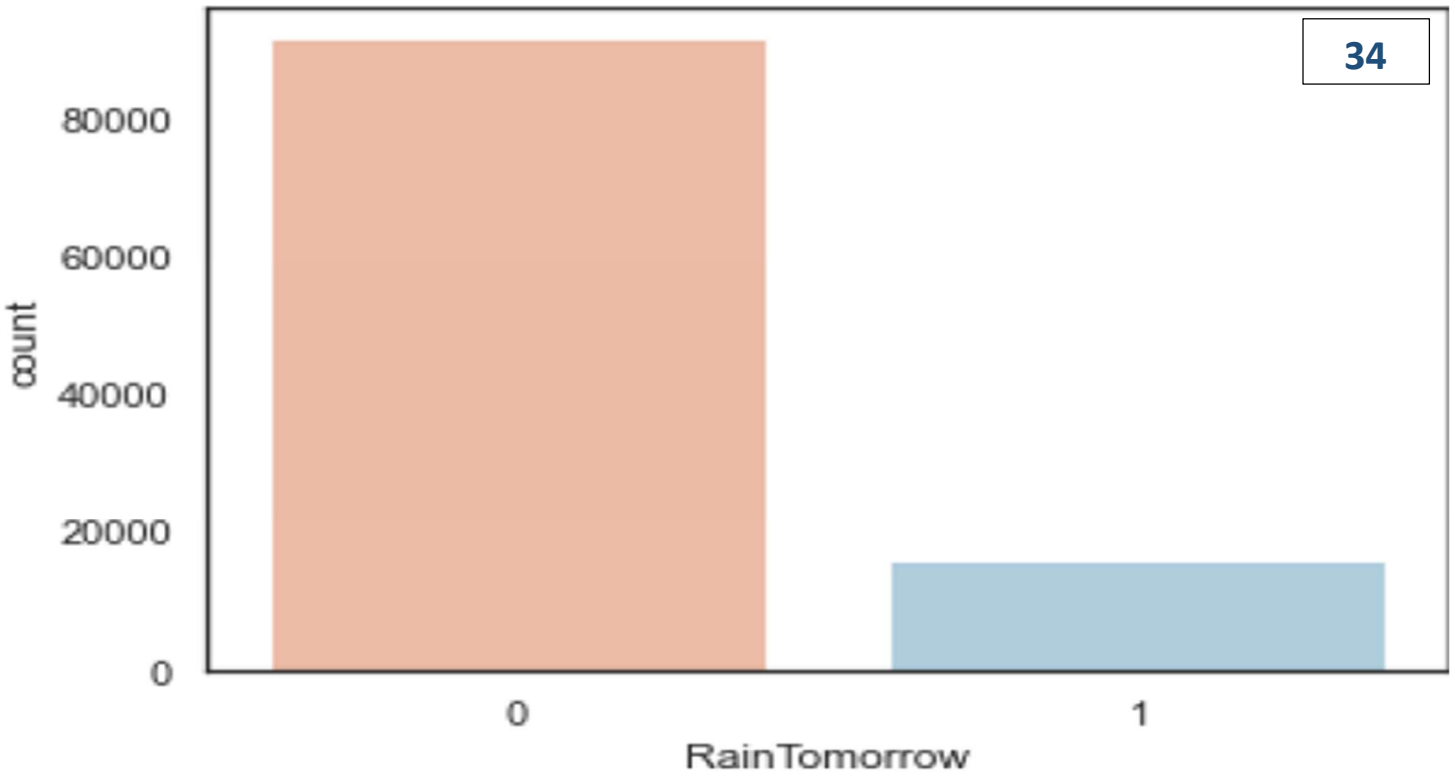
18 Balanced Data

[574]:
```
df.RainTomorrow.value_counts()
```

33

[574]:
```
0    91633
1    16108
Name: RainTomorrow, dtype: int64
```

[575]:
```
# plot to know count of 0 ,1 in RainTomorrow
sns.countplot(x = "RainTomorrow", data=df, palette = "RdBu")
plt.show()
```



As we have seen there is a very large number of zeros compared to ones which means there are great number or no to the number of yes and this will cause a bias in later calculations so we will apply some statistics to make the data unbiased as we see in the next image(35)

35

```
# get count of 0 in df_0_count and count of 1 in df_1_count of RainTomorrow
df_0_count , df_1_count=df.RainTomorrow.value_counts()
df_0=df[df['RainTomorrow']==0]
df_1=df[df['RainTomorrow']==1]
```

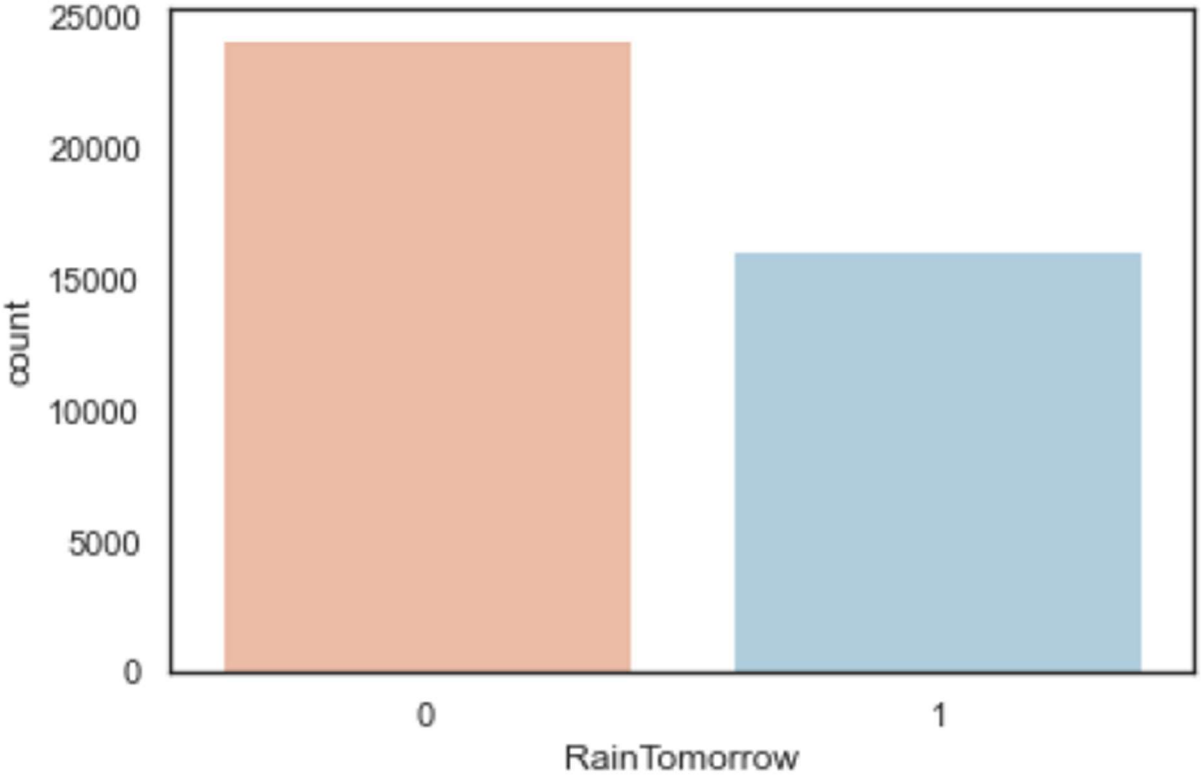
```
# get sample of RainTomorrow =0 equal RainTomorrow=1 count
df_0_under=df_0.sample(int(df_1_count*1.5))
balance_df=pd.concat([df_0_under,df_1],axis=0)
```

```
balance_df.RainTomorrow.value_counts()
```

```
0    24162
1    16108
Name: RainTomorrow, dtype: int64
```

```
sns.countplot(x = "RainTomorrow", data=balance_df, palette = "RdBu")
plt.show()
```

So we have taken a random sample from large data (no) and concatenate it with column which had fewer number of data and the percent is 60/40 as shown in the next image(36)



36

And Finally we here we have the Data after Cleaning showed in image (37):

```
balance_df.head()
```

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	\
84218	8	7	0.2	3	5	4	
118283	2	6	0.0	4	3	7	
50128	2	4	0.0	6	0	7	
47149	5	4	0.0	5	0	4	
138780	0	5	0.0	0	0	0	

	Humidity9am	Humidity3pm	Pressure9am	RainToday	RainTomorrow	\
84218	3	5	2	0	0	
118283	0	0	5	0	0	
50128	3	3	0	0	0	
47149	6	4	4	0	0	
138780	1	0	6	0	0	

	Location_encoder	WGD_encoder	WD9_encoder	WD3_encoder
84218	0.163722	0.104573	0.097423	0.110395
118283	0.120497	0.105186	0.207187	0.129860
50128	0.145515	0.206702	0.159149	0.155572
47149	0.137196	0.209525	0.097423	0.209209
138780	0.053452	0.117579	0.186718	0.118427

Sources:

www.geeksforgeeks.org/python-sys-module/#:~:text=The%20sys%20module%20in%20Python,interact%20strongly%20with%20the%20interpreter

www.towardsdatascience.com/dealing-with-categorical-variables-by-using-target-encoder-a0f1733a4c69

www.youtu.be/du8YTUgOCJ0

www.youtu.be/JnlM4yLFNuo