```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import preprocessing
from sklearn import model_selection
from sklearn import metrics
from sklearn import linear_model
from sklearn import ensemble
from sklearn import tree
from sklearn import svm
import xgboost


from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Rainfall weather.csv")
```

analyse the data

```
df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	Wind
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	

5 rows × 23 columns

Handling missing values

```
df.isnull().sum()*100/len(df)
```

```
Date              0.000000
Location          0.000000
MinTemp           1.020899
MaxTemp           0.866905
Rainfall          2.241853
Evaporation      43.166506
Sunshine         48.009762
WindGustDir       7.098859
WindGustSpeed     7.055548
WindDir9am        7.263853
WindDir3pm        2.906641
WindSpeed9am      1.214767
WindSpeed3pm      2.105046
Humidity9am       1.824557
Humidity3pm       3.098446
Pressure9am      10.356799
Pressure3pm      10.331363
Cloud9am         38.421559
Cloud3pm         40.807095
Temp9am           1.214767
Temp3pm           2.481094
RainToday         2.241853
RainTomorrow      2.245978
dtype: float64
```

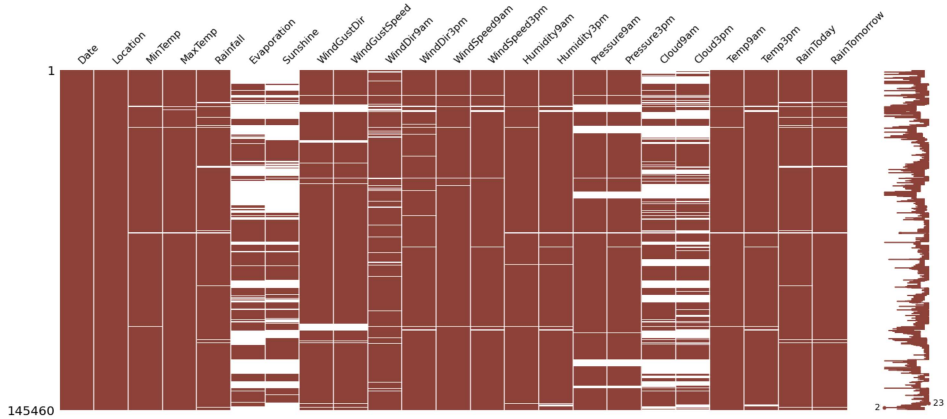
```
df.isnull().sum()
```

```
Date                0
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
```

```
import missingno as msno
```

```
msno.matrix(df, color= (0.55, 0.255, 0.225), fontsize=16)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fea09d33210>
```




```
df_cat = df[['RainToday', 'WindGustDir', 'WindDir9am', 'WindDir3pm']]
df.drop(columns=['Evaporation', 'Sunshine', 'Cloud9am', 'Cloud3pm'] ,axis=1, inplace=True)
df.drop(columns=['RainToday', 'WindGustDir', 'WindDir9am', 'WindDir3pm'] ,axis=1, inplace=Tru
```

```
df['MinTemp'].fillna (df['MinTemp'].mean(), inplace=True)
df['MaxTemp'].fillna(df['MaxTemp'].mean(), inplace=True)
df['Rainfall'].fillna (df['Rainfall'].mean(), inplace=True)
df['WindGustSpeed'].fillna (df['WindGustSpeed'].mean(), inplace=True)
df['WindSpeed9am'].fillna (df['WindSpeed9am'].mean(), inplace=True)
df['WindSpeed3pm' ].fillna (df['WindSpeed3pm'].mean(), inplace=True)
```

```
df['Humidity9am'].fillna (df['Humidity9am'].mean(), inplace=True)
df['Humidity3pm'].fillna (df['Humidity3pm'].mean(), inplace=True)
df['Pressure9am' ].fillna (df['Pressure9am'].mean(), inplace=True)
df['Pressure3pm' ].fillna (df[ 'Pressure3pm'].mean(), inplace=True)
df['Temp9am'].fillna (df['Temp9am']. mean(), inplace=True)
df['Temp3pm' ].fillna(df['Temp3pm'].mean(), inplace=True)

#Loading the names of categorical columns
cat_names=df_cat.columns
# intializing the simple imputer for missing categorical values
import numpy as np
from sklearn.impute import SimpleImputer
imp_mode= SimpleImputer(missing_values=np.nan, strategy='most_frequent')
# fitting and transforming the missing data
df_cat=imp_mode.fit_transform(df_cat)
# converting array to dataframe
df_cat= pd.DataFrame(df_cat, columns=cat_names)
# concatinating the categorical and numeric
df = pd.concat([df, df_cat], axis=1)
```

Data visulization

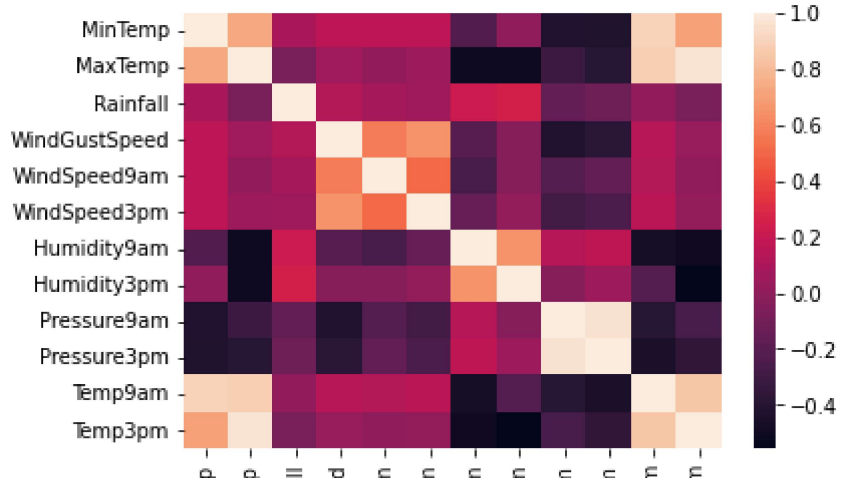
```
df.corr()
```

	MinTemp	MaxTemp	Rainfall	WindGustSpeed
MinTemp	1.000000	0.733400	0.102706	0.172553
MaxTemp	0.733400	1.000000	-0.074040	0.065895
Rainfall	0.102706	-0.074040	1.000000	0.126446
WindGustSpeed	0.172553	0.065895	0.126446	1.000000
WindSpeed9am	0.173404	0.014294	0.085925	0.577319
WindSpeed3pm	0.173058	0.049717	0.056527	0.657243
Humidity9am	-0.230970	-0.497927	0.221380	-0.207964
Humidity3pm	0.005995	-0.498760	0.248905	-0.025355
Pressure9am	-0.423584	-0.308309	-0.159055	-0.425760

```
cor = df.corr()
```

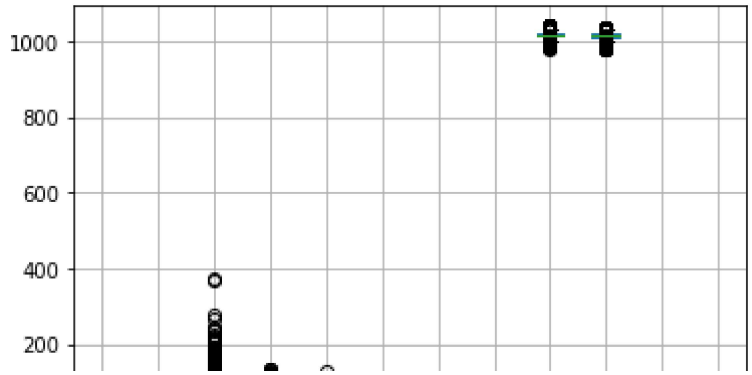
```
sns.heatmap(data=cor,xticklabels=cor.columns,yticklabels=cor.columns.values)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fea08c0c050>
```



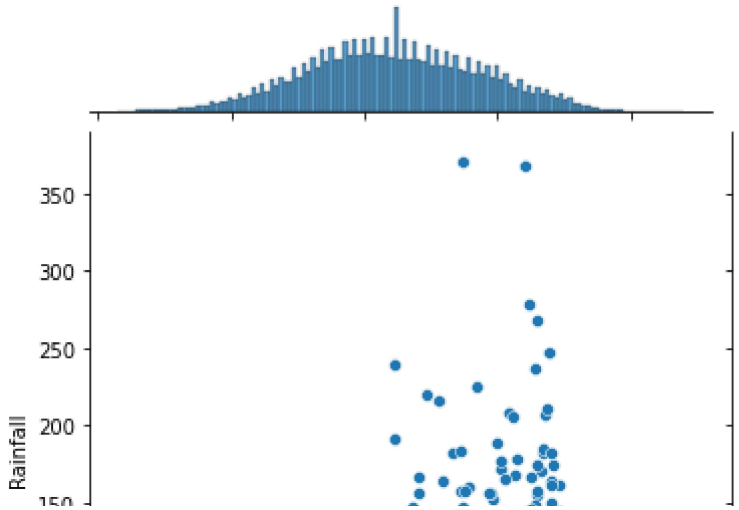

```
df.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fea08a71310>
```



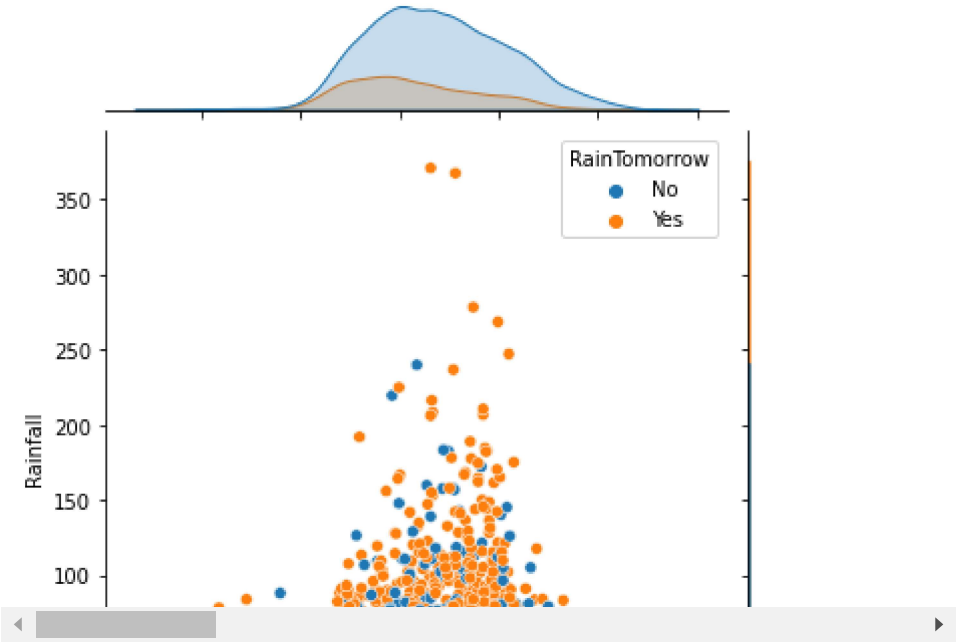

```
sns.jointplot(df["MinTemp"],df['Rainfall'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fea05d6cfd0>
```



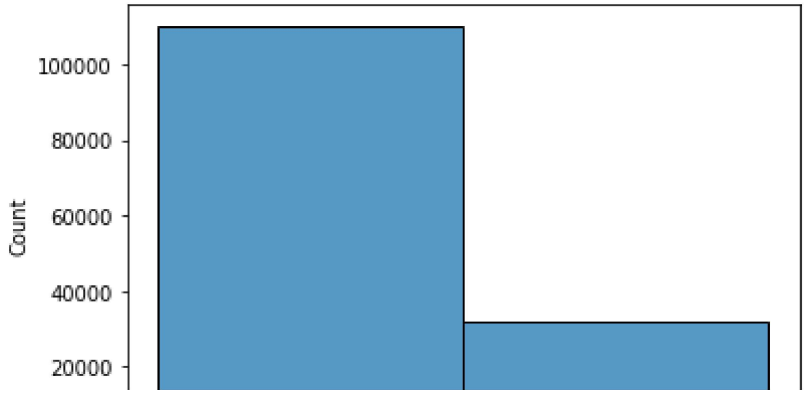

```
sns.jointplot(df["MaxTemp"],df['Rainfall'],hue=df['RainTomorrow'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fea01574150>
```



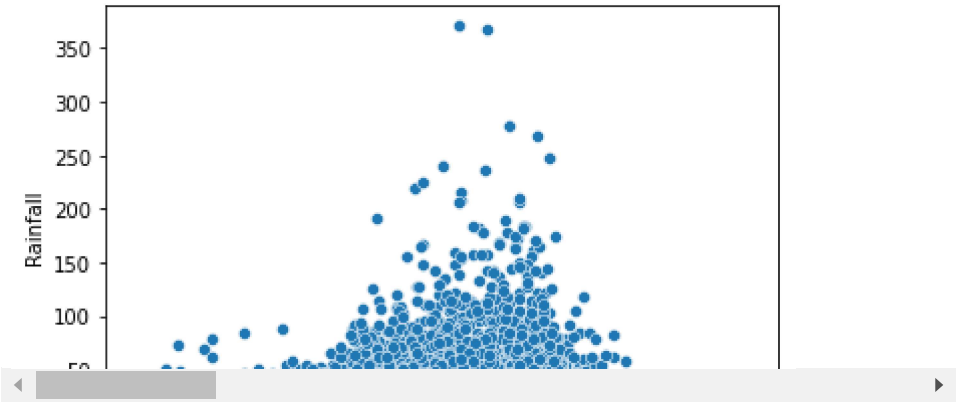

```
sns.histplot(df['RainTomorrow'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fea01115890>
```




```
sns.scatterplot(df['MaxTemp'],df['Rainfall'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fea011787d0>
```




```
sns.displot(df['MinTemp'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7fea06de4750>
```

Spliting the dataset into Dependent and independent variable

```
y=df['RainTomorrow']
x=df.drop('RainTomorrow',axis=1)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler


y=df['RainTomorrow']
x=df.drop('RainTomorrow',axis=1)


names=x.columns


names
```

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm', 'RainToday',
       'WindGustDir', 'WindDir9am', 'WindDir3pm'],
      dtype='object')
```

```
sc=StandardScaler()
```

splitting the data into train and test

```
from sklearn import model_selection


x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.2,random_state
```