In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
mpl.style.use(['ggplot'])
%matplotlib inline
```

In [2]:
```
from sklearn import preprocessing,metrics
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.neighbors import KNeighborsClassifier
```

In [3]:
```
Wdata =pd.read_csv('seattle-weather.csv')
```

In [4]:
```
Wdata.head()
```

Out[4]:

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain

In [5]:
```
Wdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           1461 non-null   object 
 1   precipitation  1461 non-null   float64
 2   temp_max       1461 non-null   float64
 3   temp_min       1461 non-null   float64
 4   wind           1461 non-null   float64
 5   weather        1461 non-null   object 
dtypes: float64(4), object(2)
memory usage: 68.6+ KB
```

In [6]:
```
Wdata.date=pd.to_datetime(Wdata.date)
```

In [7]:
```
Wdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   date           1461 non-null   datetime64[ns]
 1   precipitation  1461 non-null   float64       
 2   temp_max       1461 non-null   float64       
 3   temp_min       1461 non-null   float64       
 4   wind           1461 non-null   float64       
 5   weather        1461 non-null   object        
dtypes: datetime64[ns](1), float64(4), object(1)
memory usage: 68.6+ KB
```

In [8]:
```
Wdata.shape
```

Out[8]:
```
(1461, 6)
```

In [11]:
```
Wdata.columns
```

Out[11]:
```
Index(['date', 'precipitation', 'temp_max', 'temp_min', 'wind', 'weather'], dtype='object')
```

In [12]:
```
Wdata.isna().sum()
```

Out[12]:
```
date             0
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
dtype: int64
```

In [15]:
```
Wdata.duplicated().sum()
```

Out[15]:
```
0
```

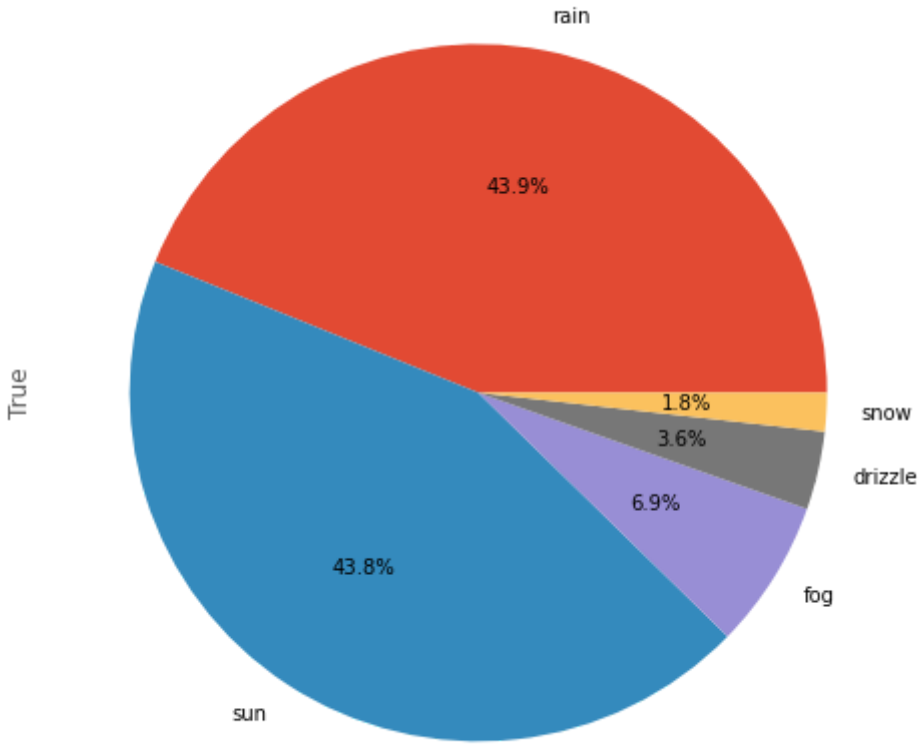
In [16]:
```
Wdata['weather'].value_counts()
```

Out[16]:
```
rain       641
sun        640
fog        101
drizzle     53
```

```
snow        26
Name: weather, dtype: int64
```

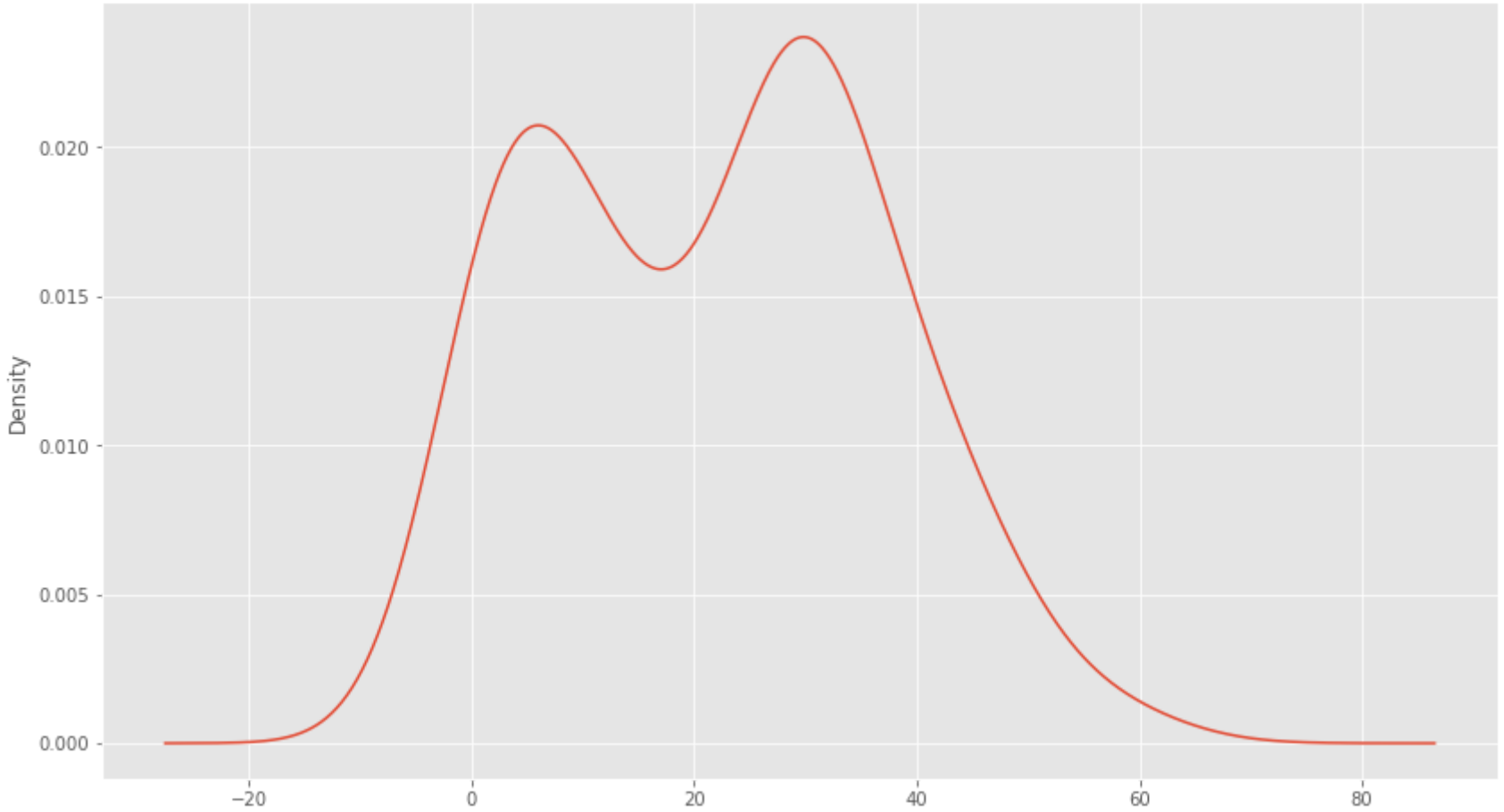
In [19]:
```
Wdata['weather'].value_counts().plot(kind='pie',
                                     figsize=(14,8),
                                     autopct='%1.1f%%',
                                     label=True)
```

Out[19]:
```
<AxesSubplot:ylabel='True'>
```



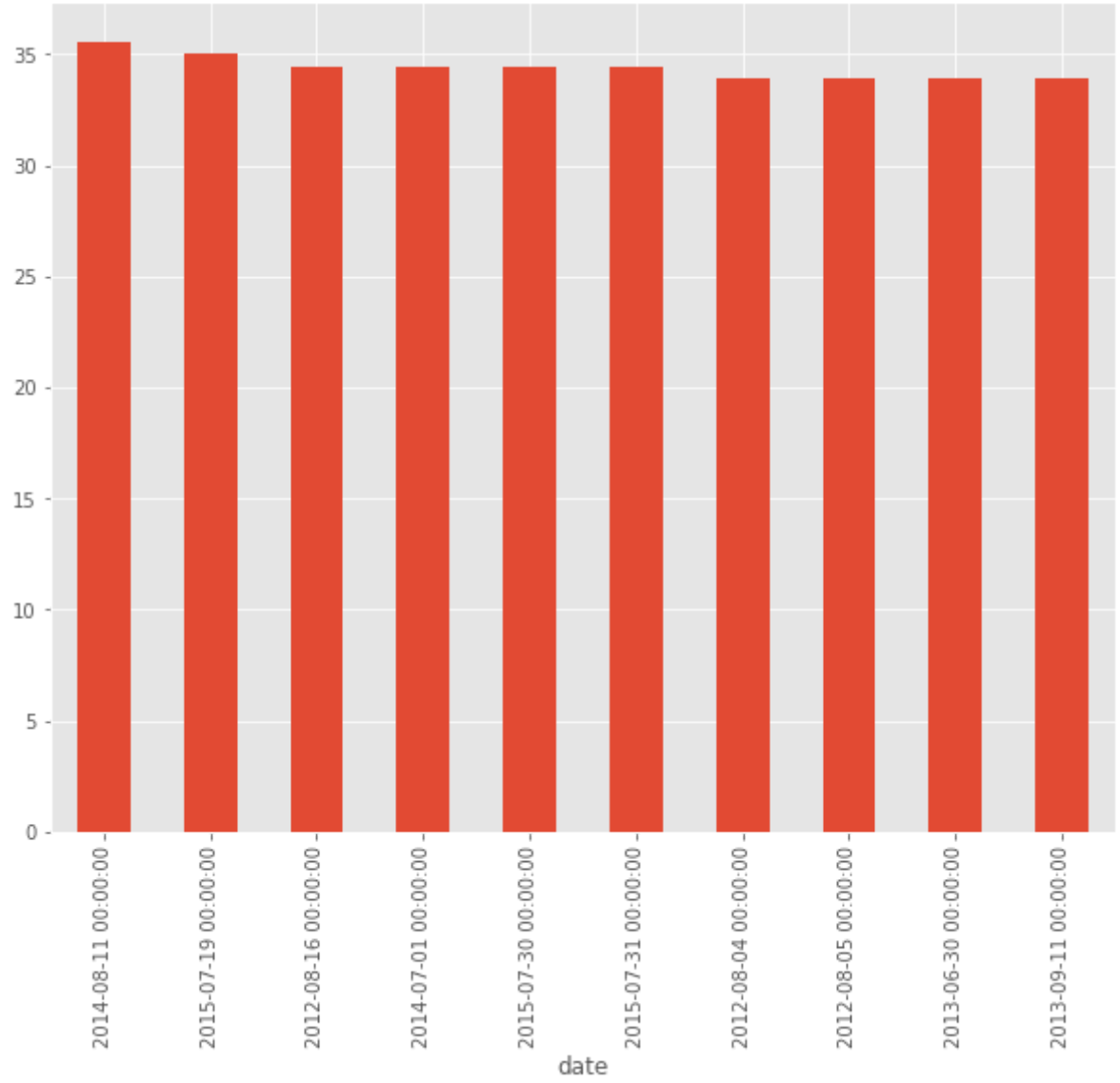
In [24]:
```
Wdata['temp_max'].value_counts().plot(kind='density',figsize=(14,8))
```

Out[24]:
```
<AxesSubplot:ylabel='Density'>
```



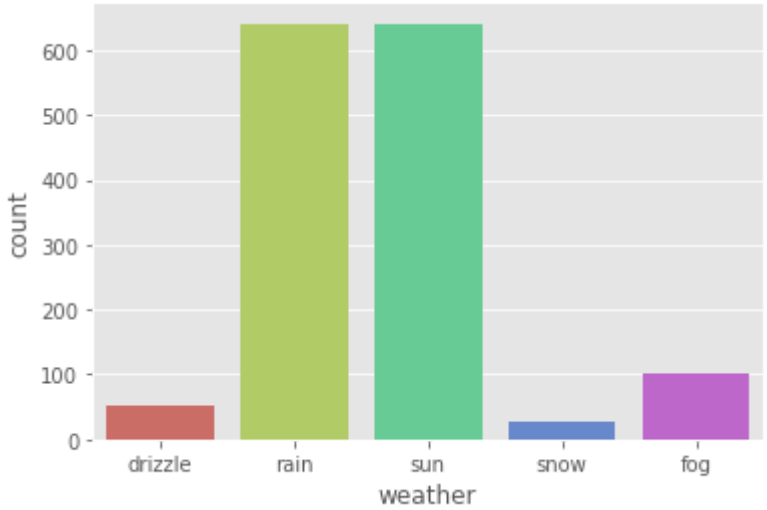
In [56]:
```
Wdata.groupby(['date'])['temp_max'].max().nlargest(10).plot(kind='bar',figsize=(10,8))
```

Out[56]:
```
<AxesSubplot:xlabel='date'>
```



In [58]:
```
import warnings
warnings.filterwarnings('ignore')
sns.countplot("weather",data=Wdata,palette="hls")
```

Out[58]:
```
<AxesSubplot:xlabel='weather', ylabel='count'>
```



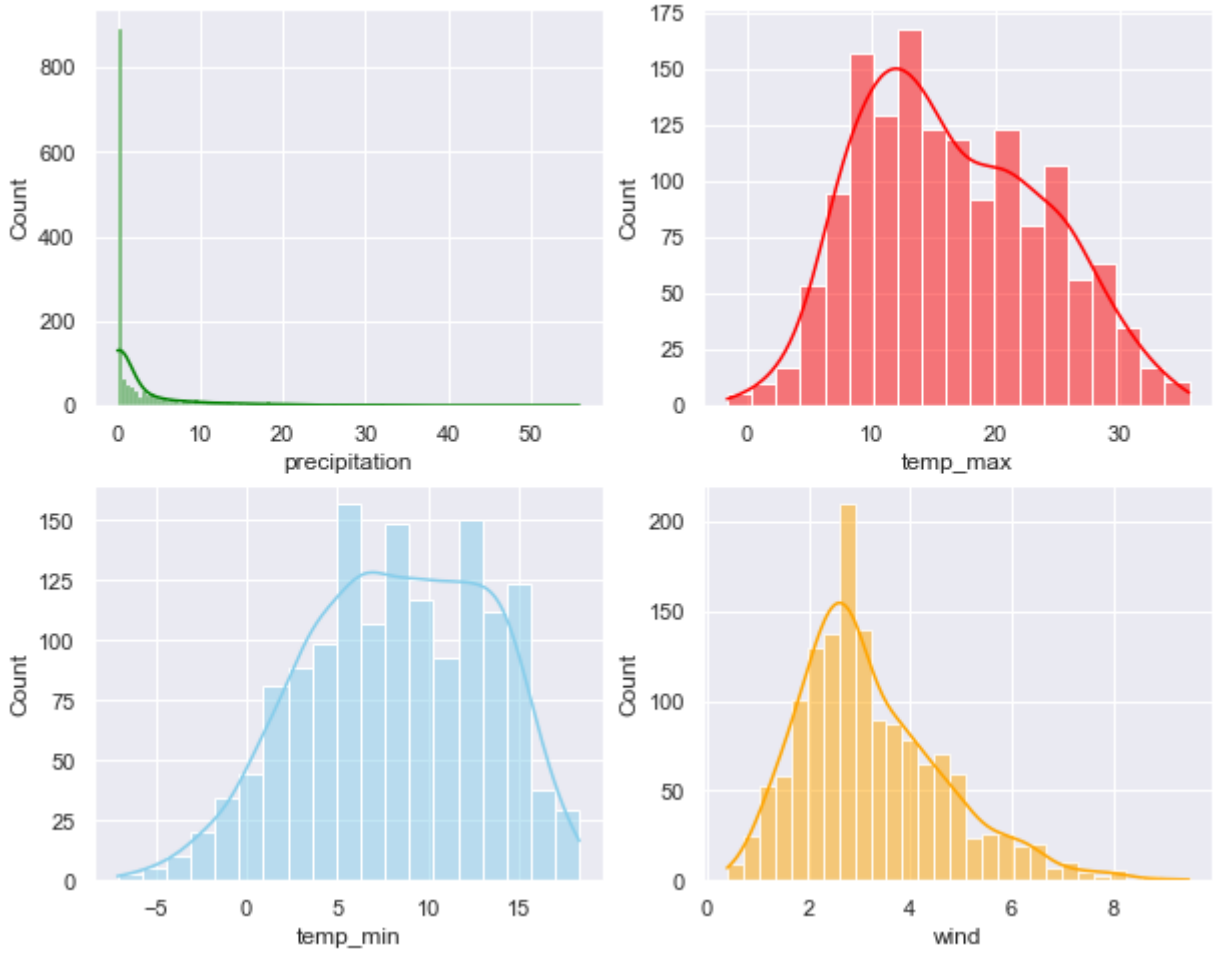
In [60]:
```
Wdata.describe()
```

Out[60]:

	precipitation	temp_max	temp_min	wind
count	1461.000000	1461.000000	1461.000000	1461.000000
mean	3.029432	16.439083	8.234771	3.241136
std	6.680194	7.349758	5.023004	1.437825
min	0.000000	-1.600000	-7.100000	0.400000
25%	0.000000	10.600000	4.400000	2.200000
50%	0.000000	15.600000	8.300000	3.000000
75%	2.800000	22.200000	12.200000	4.000000
max	55.900000	35.600000	18.300000	9.500000

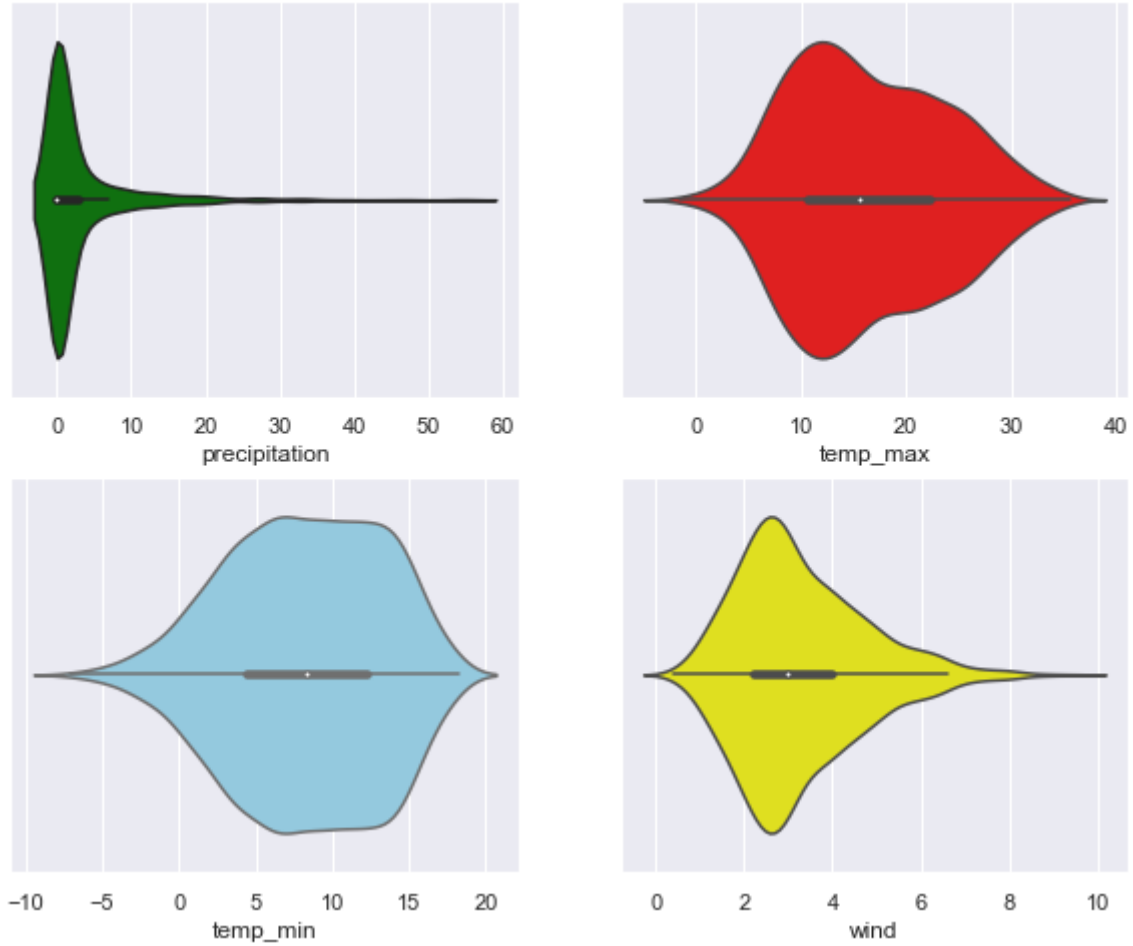
In [62]:
```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=Wdata,x="precipitation",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=Wdata,x="temp_max",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=Wdata,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=Wdata,x="wind",kde=True,ax=axs[1,1],color='orange')
```

Out[62]:
```
<AxesSubplot:xlabel='wind', ylabel='Count'>
```



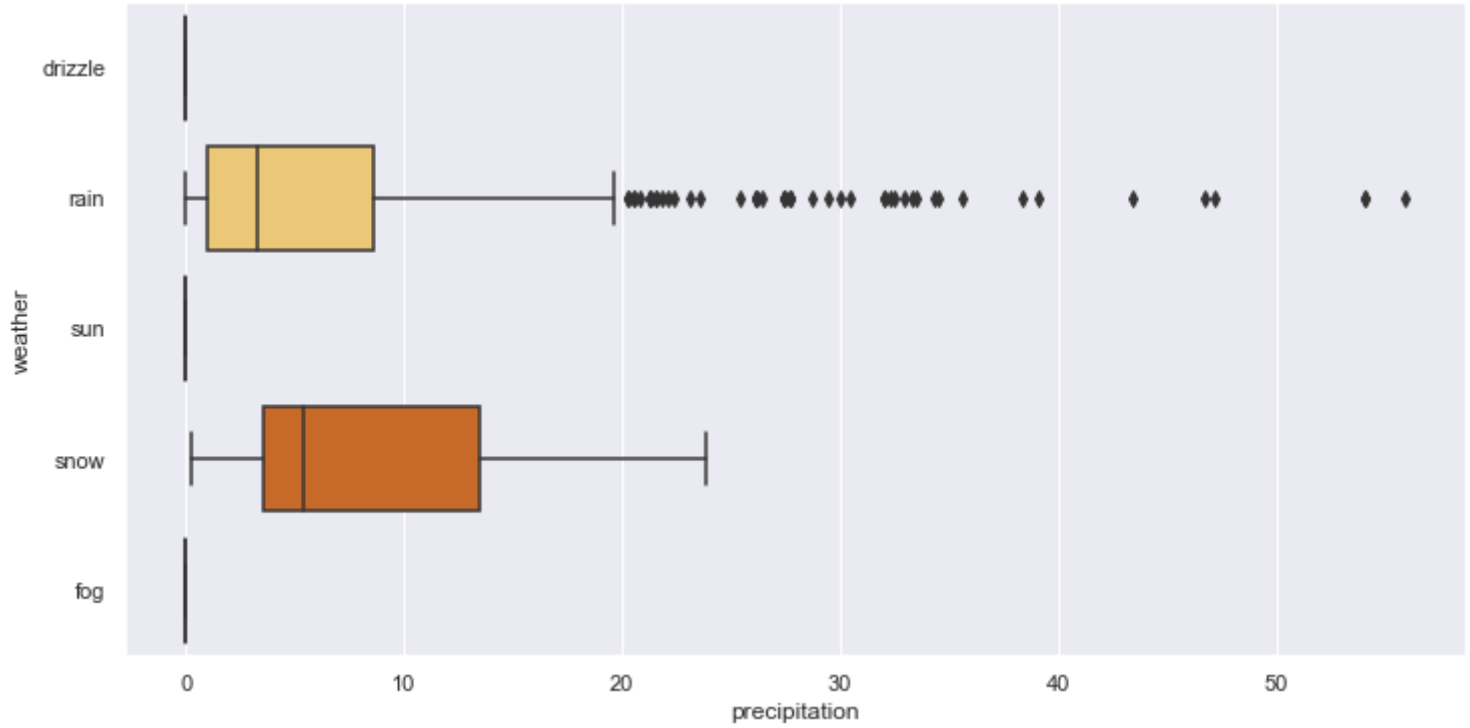
In [64]:
```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.violinplot(data=Wdata,x="precipitation",kde=True,ax=axs[0,0],color='green')
sns.violinplot(data=Wdata,x="temp_max",kde=True,ax=axs[0,1],color='red')
sns.violinplot(data=Wdata,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
sns.violinplot(data=Wdata,x="wind",kde=True,ax=axs[1,1],color='yellow')
```

Out[64]:
```
<AxesSubplot:xlabel='wind'>
```



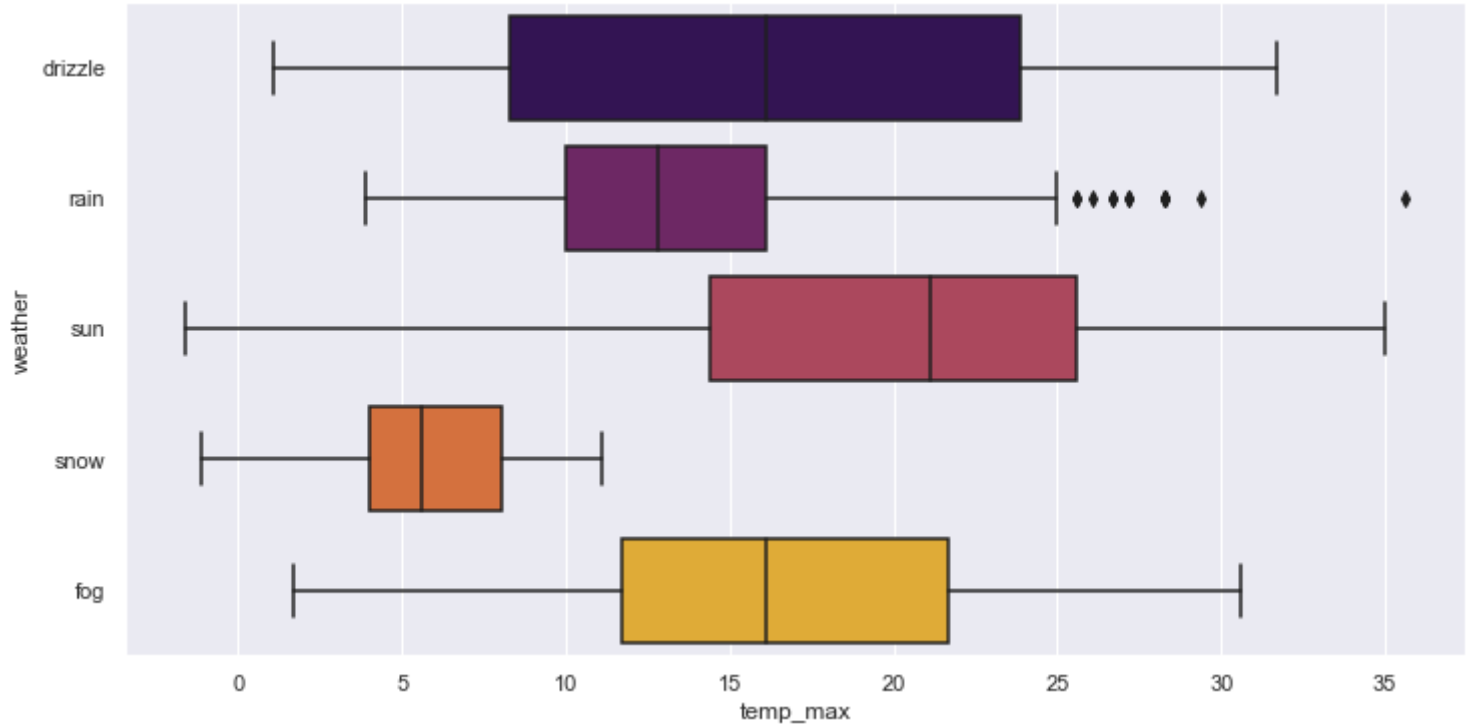
In [66]:
```
plt.figure(figsize=(12,6))
sns.boxplot("precipitation","weather",data=Wdata,palette="YlOrBr")
```

Out[66]:
```
<AxesSubplot:xlabel='precipitation', ylabel='weather'>
```



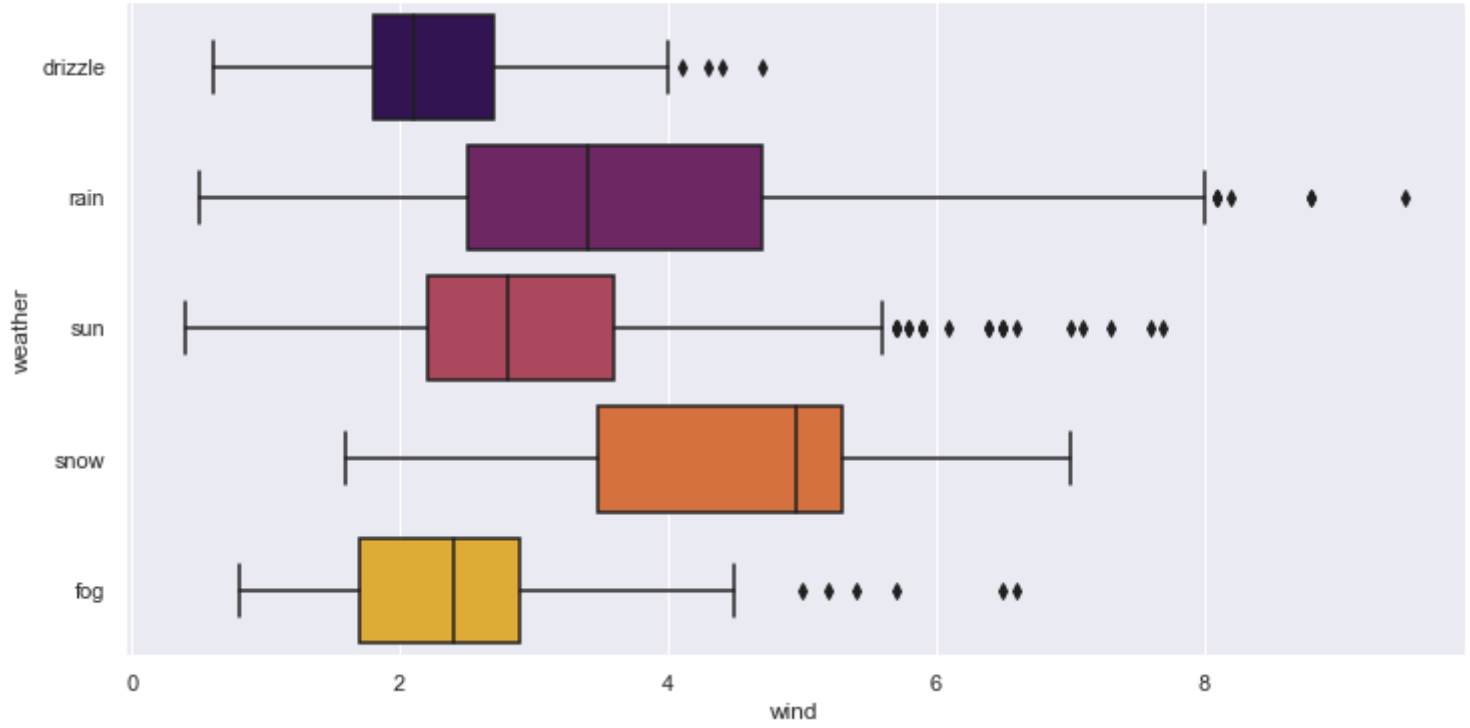
In [68]:
```
plt.figure(figsize=(12,6))
sns.boxplot("temp_max","weather",data=Wdata,palette="inferno")
```

Out[68]:
```
<AxesSubplot:xlabel='temp_max', ylabel='weather'>
```



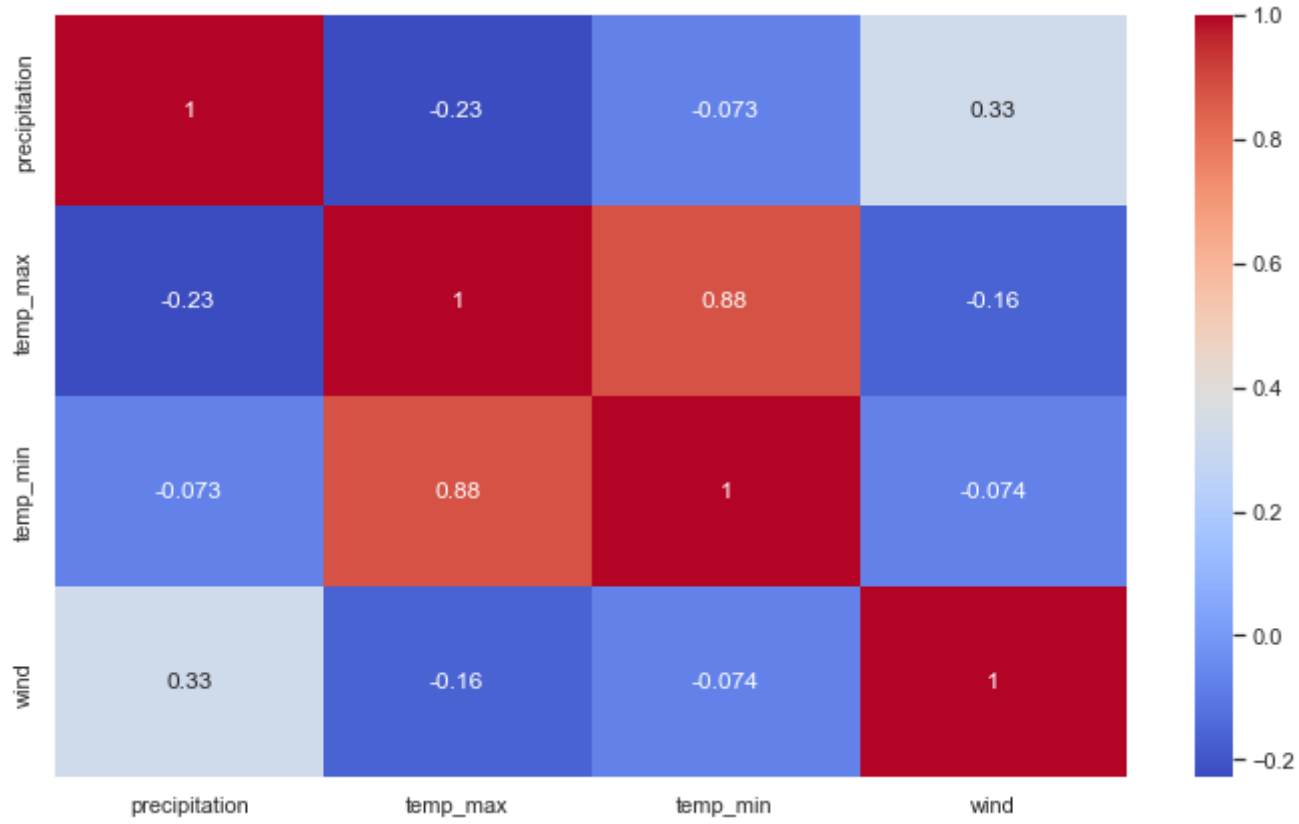
In [69]:
```
plt.figure(figsize=(12,6))
sns.boxplot("wind","weather",data=Wdata,palette="inferno")
```

Out[69]:
```
<AxesSubplot:xlabel='wind', ylabel='weather'>
```



In [70]:
```
plt.figure(figsize=(12,7))
sns.heatmap(Wdata.corr(),annot=True,cmap='coolwarm')
```

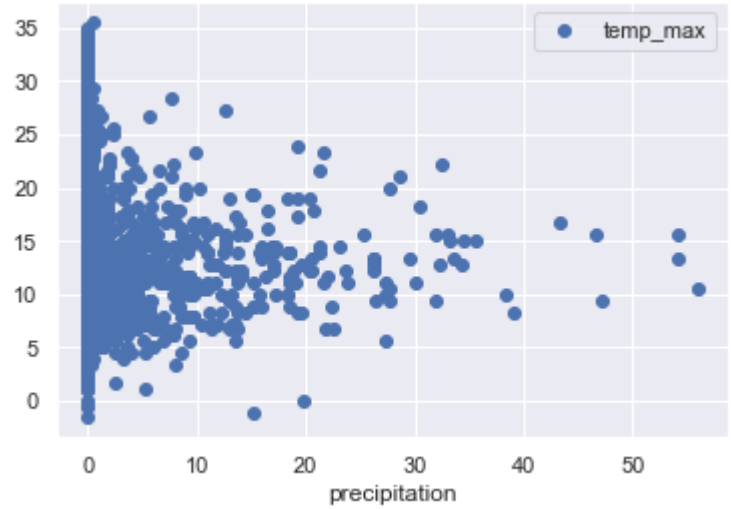
Out[70]:
```
<AxesSubplot:>
```



In [74]:
```
from scipy import stats
```

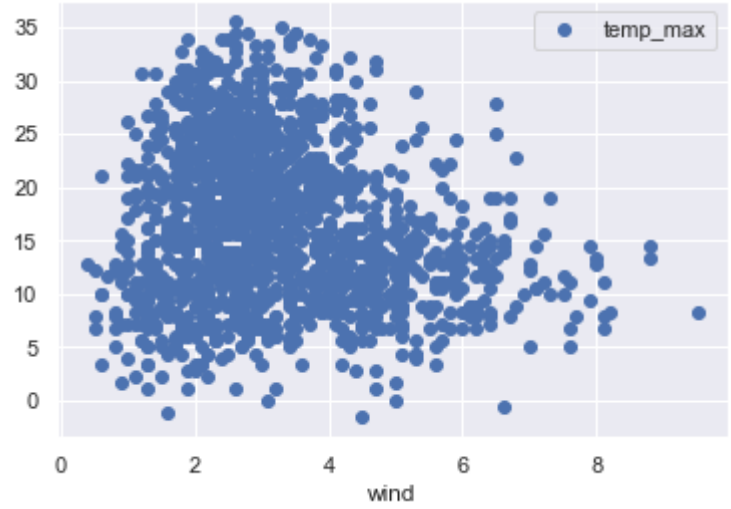
In [75]:
```
Wdata.plot("precipitation","temp_max",style='o')
print("Pearson correlation:",Wdata["precipitation"].corr(Wdata["temp_max"]))
print("T Test and P value:",stats.ttest_ind(Wdata["precipitation"],Wdata["temp_max"]))
```

```
Pearson correlation: -0.22855481643297043
T Test and P value: Ttest_indResult(statistic=-51.60685279531918, pvalue=0.0)
```



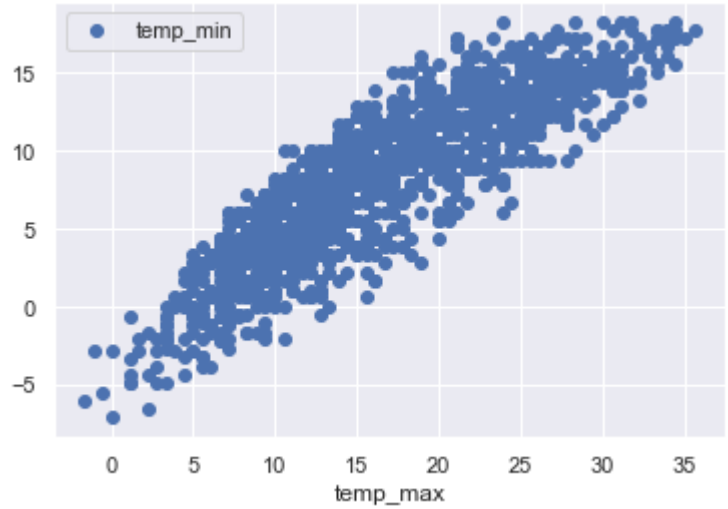
In [78]:
```
Wdata.plot("wind","temp_max",style='o')
print("Pearson correlation:",Wdata["wind"].corr(Wdata["temp_max"]))
print("T Test and P value:",stats.ttest_ind(Wdata["wind"],Wdata["temp_max"]))
```

```
Pearson correlation: -0.1648566348749548
T Test and P value: Ttest_indResult(statistic=-67.3601643301846, pvalue=0.0)
```



In [79]:
```
Wdata.plot("temp_max","temp_min",style='o')
```

Out[79]:
```
<AxesSubplot:xlabel='temp_max'>
```



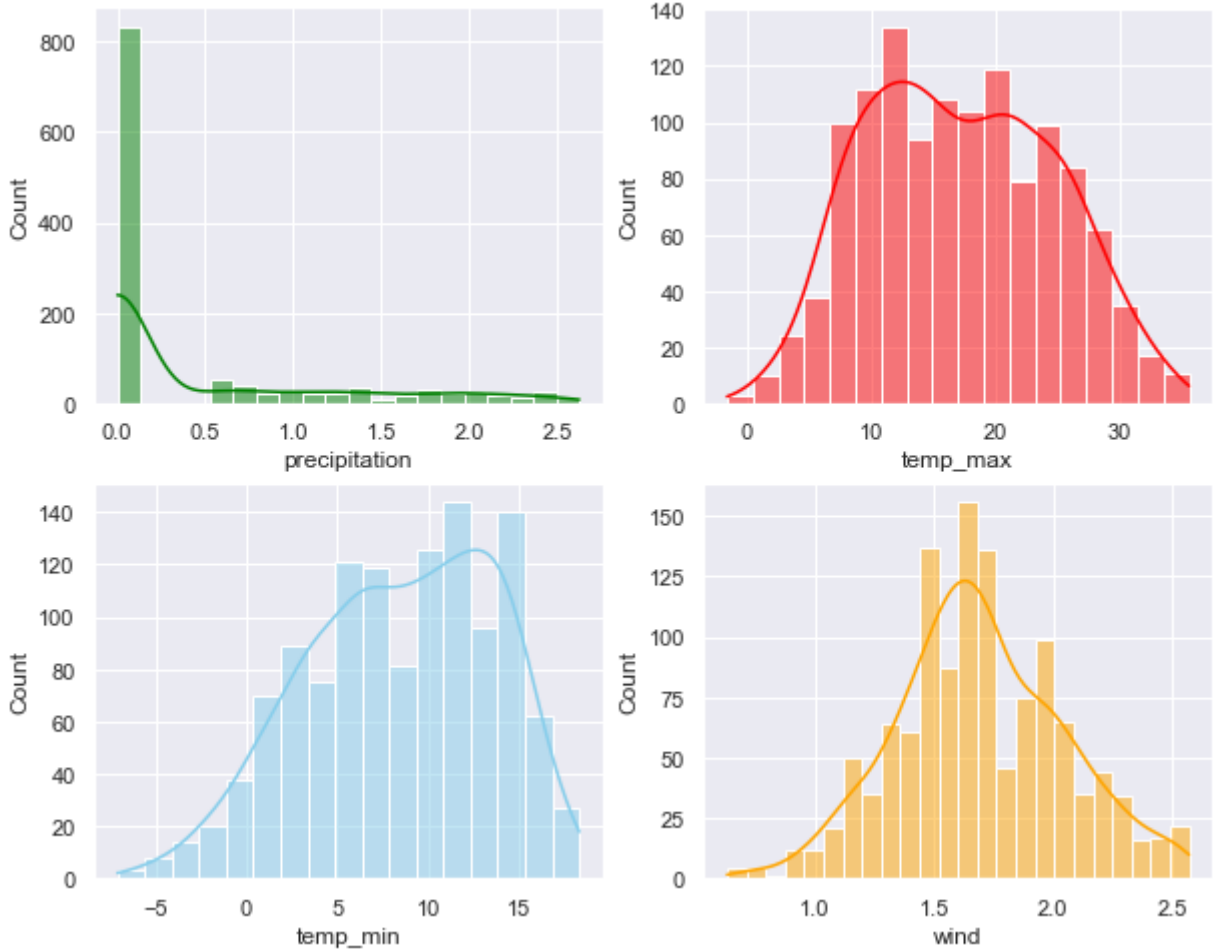
In [82]:
```
df=Wdata.drop(["date"],axis=1)
```

In [83]:
```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
df=df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
```

In [84]:
```
df.precipitation=np.sqrt(df.precipitation)
df.wind=np.sqrt(df.wind)
```

In [85]:
```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=df,x="precipitation",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=df,x="temp_max",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=df,x="temp_min",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=df,x="wind",kde=True,ax=axs[1,1],color='orange')
```

Out[85]:
```
<AxesSubplot:xlabel='wind', ylabel='Count'>
```



In [87]:
```
df.head()
```

Out[87]:

	precipitation	temp_max	temp_min	wind	weather
0	0.000000	12.8	5.0	2.167948	drizzle
2	0.894427	11.7	7.2	1.516575	rain
4	1.140175	8.9	2.8	2.469818	rain
5	1.581139	4.4	2.2	1.483240	rain
6	0.000000	7.2	2.8	1.516575	rain

In [88]:
```
df.columns
```

Out[88]:
```
Index(['precipitation', 'temp_max', 'temp_min', 'wind', 'weather'], dtype='object')
```

In [89]:
```
X = df[['precipitation', 'temp_max', 'temp_min', 'wind']].values
X[0:5]
```

Out[89]:
```
array([[ 0.        , 12.8       ,  5.        ,  2.16794834],
       [ 0.89442719, 11.7       ,  7.2       ,  1.51657509],
       [ 1.14017543,  8.9       ,  2.8       ,  2.46981781],
       [ 1.58113883,  4.4       ,  2.2       ,  1.4832397 ],
       [ 0.        ,  7.2       ,  2.8       ,  1.51657509]])
```

In [91]:
```
Y=df['weather'].values
Y[0:5]
```

Out[91]:
```
array(['drizzle', 'rain', 'rain', 'rain', 'rain'], dtype=object)
```

In [92]:
```
X=preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]
```

Out[92]:
```
array([[-0.60933199, -0.5794075 , -0.66479802,  1.34668256],
       [ 0.57361988, -0.72521075, -0.23932352, -0.49985515],
       [ 0.89864167, -1.09634631, -1.09027252,  2.20243364],
       [ 1.48185123, -1.69281417, -1.20631102, -0.59435558],
       [-0.60933199, -1.32167861, -1.09027252, -0.49985515]])
```

In [111…
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.1,random_state=2)
print('Train set:',x_train.shape,y_train.shape)
print('Test set:',x_test.shape,y_test.shape)
```

```
Train set: (1109, 4) (1109,)
Test set: (124, 4) (124,)
```

In [112…
```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
print("KNN Accuracy:{:.2f}%".format(knn.score(x_test,y_test)*100))
```

```
KNN Accuracy:84.68%
```

In [96]:
```
K=4
neigh=KNeighborsClassifier(n_neighbors=K).fit(x_train,y_train)
neigh
```

Out[96]:
```
KNeighborsClassifier(n_neighbors=4)
```

In [97]:
```
yhat=neigh.predict(x_test)
yhat[0:5]
```

Out[97]:
```
array(['sun', 'snow', 'sun', 'sun', 'snow'], dtype=object)
```

In [98]:
```
print('Train set Accuracy:',metrics.accuracy_score(y_train,neigh.predict(x_train)))
```

```
print('Test set Accuracy:',metrics.accuracy_score(y_test,yhat))
```

```
Train set Accuracy: 0.842200180342651
Test set Accuracy: 0.8225806451612904
```

In [113…]
```
ks=10
mean_acc = np.zeros((ks-1))
std_acc = np.zeros((ks-1))

for n in range(1,ks):
    neigh=KNeighborsClassifier(n_neighbors=n).fit(x_train,y_train)
    yhat=neigh.predict(x_test)
    mean_acc[n-1]=metrics.accuracy_score(y_test,yhat)
    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

Out[113]:
```
array([0.7983871 , 0.71774194, 0.81451613, 0.82258065, 0.84677419,
       0.86290323, 0.86290323, 0.83064516, 0.83064516])
```

In [106…]
```
from sklearn.preprocessing import LabelEncoder
```

In [107…]
```
lc=LabelEncoder()
df["weather"]=lc.fit_transform(df["weather"])
```

In [108…]
```
df.head()
```

Out[108]:

	precipitation	temp_max	temp_min	wind	weather
0	0.000000	12.8	5.0	2.167948	0
2	0.894427	11.7	7.2	1.516575	2
4	1.140175	8.9	2.8	2.469818	2
5	1.581139	4.4	2.2	1.483240	2
6	0.000000	7.2	2.8	1.516575	2

In [115…]
```
x=((df.loc[:,df.columns!="weather"]).astype(int)).values[:,0:]
y=df["weather"].values
```

In [110…]
```
df.weather.unique()
```

Out[110]:
```
array([0, 2, 4, 3, 1])
```

In [103…]
```
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-1.7.2-py3-none-win_amd64.whl (89.1 MB)
Requirement already satisfied: numpy in c:\users\ahmed\anaconda3\lib\site-packages (from xgboost) (1.21.5)
Requirement already satisfied: scipy in c:\users\ahmed\anaconda3\lib\site-packages (from xgboost) (1.7.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.2
```

In [104…]
```
from xgboost import XGBClassifier
```

In [127…]
```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
```

In [124…]
```
import warnings
warnings.filterwarnings('ignore')
xgb=XGBClassifier()
xgb.fit(x_train,y_train)
print("XGB Accuracy:{:.2f}%".format(xgb.score(x_test,y_test)*100))
```

```
XGB Accuracy:0.00%
```

In [128…]
```
input=[[1.140175,8.9,2.8,2.469818]]
ot=xgb.predict(input)
print("The weather is:")
if(ot==0):
    print("Drizzle")
elif(ot==1):
    print("Fog")
elif(ot==2):
    print("Rain")
elif(ot==3):
    print("snow")
else:
    print("Sun")
```

```
The weather is:
Rain
```

In [ ]: