In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
summer_data = pd.read_csv('indian_summer.csv')
```

In [3]:

```
summer_data.head()
```

Out[3]:

	City	Date	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	v
0	New Delhi	01-04-2021	34.0	19.0	27.1	31.6	19.0	26.1	3.1	22.60	
1	New Delhi	02-04-2021	33.9	16.0	25.8	31.8	16.0	24.9	4.5	27.62	
2	New Delhi	03-04-2021	34.8	14.6	26.0	32.2	14.6	25.1	1.3	23.18	
3	New Delhi	04-04-2021	36.8	16.9	27.1	34.2	16.9	26.0	4.8	28.00	
4	New Delhi	05-04-2021	38.8	21.0	29.9	37.1	21.0	28.9	8.1	28.85	

In [4]:

```
summer_data.tail()
```

Out[4]:

	City	Date	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	h
13645	Hyderabad	26-06-2012	32.1	22.1	25.8	35.9	22.1	26.7	19.9	
13646	Hyderabad	27-06-2012	31.8	21.1	25.5	33.3	21.1	26.1	19.0	
13647	Hyderabad	28-06-2012	31.8	23.1	26.8	33.3	23.1	27.6	19.1	
13648	Hyderabad	29-06-2012	32.8	23.1	26.7	35.1	23.1	27.5	19.5	
13649	Hyderabad	30-06-2012	32.9	23.1	27.7	34.5	23.1	28.6	18.8	

In [5]:

```
summer_data.shape
```

Out[5]:

```
(13650, 20)
```

In [6]:

```
summer_data.columns
```

Out[6]:

```
Index(['City', 'Date', 'tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'windspeed', 'windd
ir',
       'sealevelpressure', 'cloudcover', 'visibility', 'sunrise', 'sunse
t',
       'moonphase', 'conditions', 'description'],
      dtype='object')
```

In [7]:

```
summer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13650 entries, 0 to 13649
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   City              13650 non-null  object 
 1   Date              13650 non-null  object 
 2   tempmax           13615 non-null  float64
 3   tempmin           13615 non-null  float64
 4   temp              13605 non-null  float64
 5   feelslikemax      13614 non-null  float64
 6   feelslikemin      13614 non-null  float64
 7   feelslike         13604 non-null  float64
 8   dew               13605 non-null  float64
 9   humidity          13605 non-null  float64
 10  windspeed         13605 non-null  float64
 11  winddir           13600 non-null  float64
 12  sealevelpressure  10631 non-null  float64
 13  cloudcover        13605 non-null  float64
 14  visibility        13605 non-null  float64
 15  sunrise           13650 non-null  object 
 16  sunset            13650 non-null  object 
 17  moonphase         13650 non-null  float64
 18  conditions        13605 non-null  object 
 19  description       13605 non-null  object 
dtypes: float64(14), object(6)
memory usage: 2.1+ MB
```

In [8]:

```
summer_data.describe()
```

Out[8]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike
count	13615.000000	13615.000000	13605.000000	13614.000000	13614.000000	13604.000000
mean	36.728248	25.821160	31.151510	40.212605	27.221324	33.704535
std	4.115452	3.212167	3.074874	5.389016	4.907125	4.666616
min	0.000000	0.000000	19.900000	0.000000	0.000000	19.900000
25%	34.000000	23.700000	29.200000	36.500000	23.700000	30.200000
50%	37.000000	26.000000	31.100000	40.000000	26.000000	33.500000
75%	39.800000	28.100000	33.200000	43.700000	31.100000	37.200000
max	50.000000	37.000000	40.500000	79.200000	43.300000	48.500000

In [9]:

```
summer_data.isnull().sum()
```

Out[9]:

```
City                  0
Date                  0
tempmax              35
tempmin              35
temp                 45
feelslikemax         36
feelslikemin         36
feelslike            46
dew                  45
humidity             45
windspeed            45
winddir              50
sealevelpressure   3019
cloudcover           45
visibility           45
sunrise               0
sunset                0
moonphase             0
conditions           45
description          45
dtype: int64
```

In [14]:

```
summer_data = summer_data.drop(['sealevelpressure'], axis = 1)
```

In [15]:

```
summer_data.columns
```

Out[15]:

```
Index(['City', 'Date', 'tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'windspeed', 'windd
ir',
       'cloudcover', 'visibility', 'sunrise', 'sunset', 'moonphase',
       'conditions', 'description'],
      dtype='object')
```

In [16]:

```
summer_data.dropna(inplace = True)
```

In [17]:

```
summer_data.shape
```

Out[17]:

```
(13599, 19)
```

In [20]:

```
summer_data['conditions'].unique()
```

Out[20]:

```
array(['Clear', 'Partially cloudy', 'Rain, Partially cloudy',
       'Rain, Overcast', 'Overcast', 'Rain'], dtype=object)
```

In [21]:

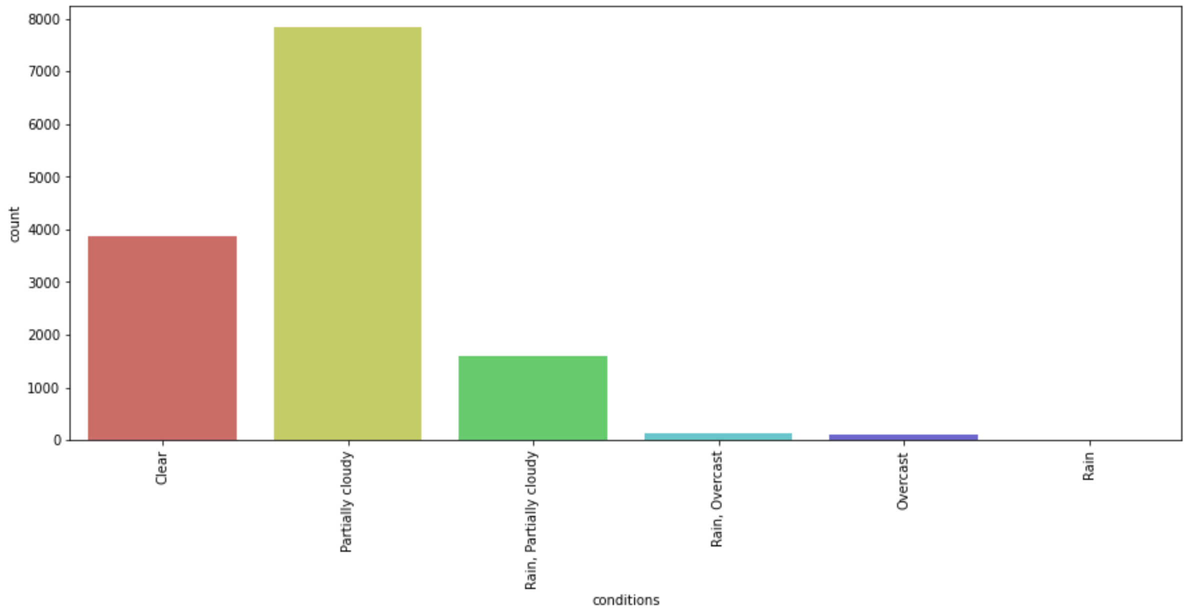
```
summer_data['conditions'].value_counts()
```

Out[21]:

```
Partially cloudy          7852
Clear                     3880
Rain, Partially cloudy    1609
Rain, Overcast             136
Overcast                   113
Rain                         9
Name: conditions, dtype: int64
```

In [22]:

```
plt.figure(figsize=(15,6))
sns.countplot('conditions', data = summer_data, palette='hls')
plt.xticks(rotation = 90)
plt.show()
```



In [24]:

```
count_clear=len(summer_data[summer_data.conditions=="Clear"])
count_pcloudy=len(summer_data[summer_data.conditions=="Partially cloudy"])
count_rpcloudy=len(summer_data[summer_data.conditions=="Rain, Partially cloudy"])
count_ro=len(summer_data[summer_data.conditions=="Rain, Overcast"])
count_overcast=len(summer_data[summer_data.conditions=="Overcast"])
count_rain=len(summer_data[summer_data.conditions=="Rain"])
```

In [25]:

```
print("Percent of Clear:{:2f}%".format((count_clear/(len(summer_data.conditions))*100))
print("Percent of Partial Cloudy:{:2f}%".format((count_pcloudy/(len(summer_data.conditi
print("Percent of Rain Partial Cloudy:{:2f}%".format((count_rpcloudy/(len(summer_data.co
print("Percent of Rain Overcast:{:2f}%".format((count_ro/(len(summer_data.conditions))*1
print("Percent of Overcast:{:2f}%".format((count_overcast/(len(summer_data.conditions))*
print("Percent of Rain:{:2f}%".format((count_rain/(len(summer_data.conditions))*100)))
```

```
Percent of Clear:28.531510%
Percent of Partial Cloudy:57.739540%
Percent of Rain Partial Cloudy:11.831752%
Percent of Rain Overcast:1.000074%
Percent of Overcast:0.830943%
Percent of Rain:0.066181%
```

In [26]:

```
summer_data[["humidity","tempmax","tempmin","windspeed"]].describe()
```

Out[26]:

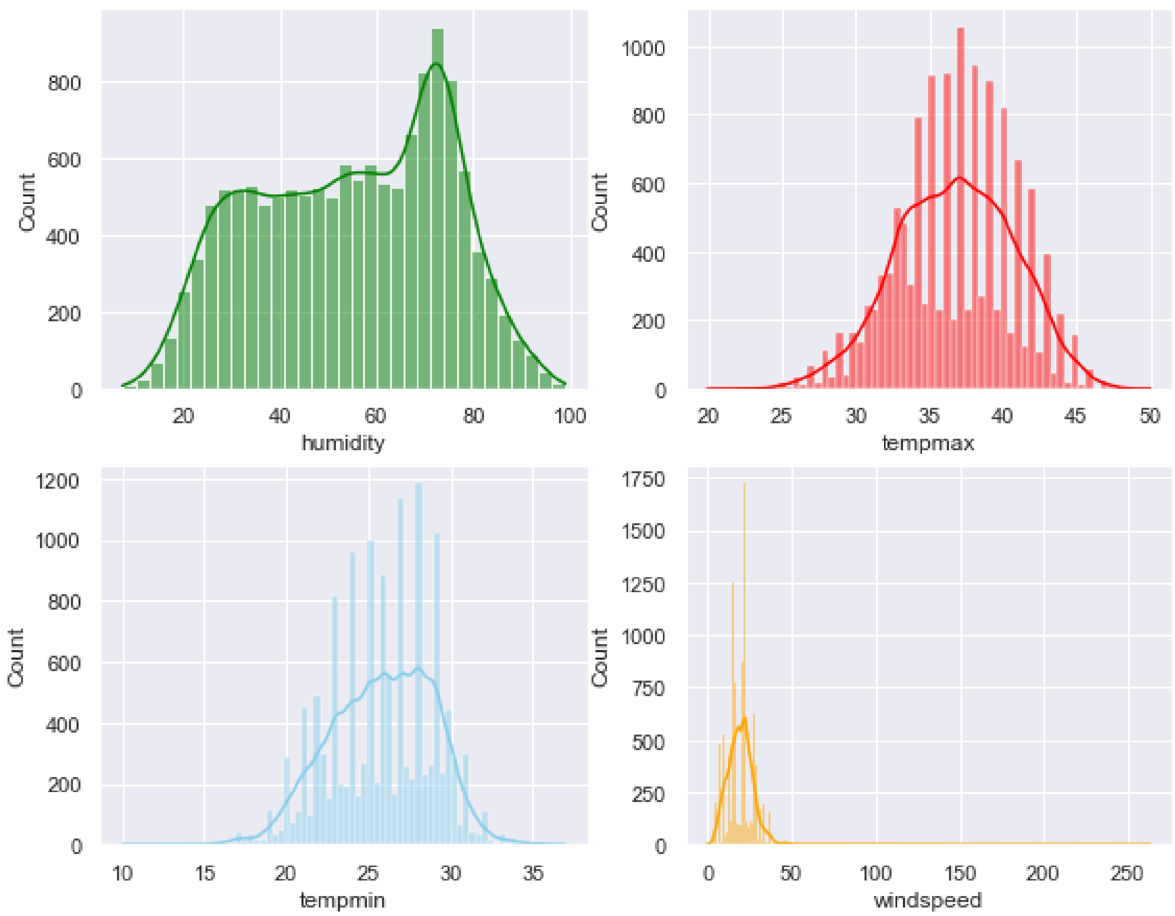
	humidity	tempmax	tempmin	windspeed
count	13599.000000	13599.000000	13599.000000	13599.000000
mean	54.643784	36.756828	25.837937	20.083293
std	19.519355	3.993890	3.133552	9.885514
min	7.410000	19.900000	10.000000	0.000000
25%	38.195000	34.000000	23.700000	14.800000
50%	56.120000	37.000000	26.000000	19.500000
75%	71.420000	39.800000	28.100000	24.100000
max	99.040000	50.000000	37.000000	263.200000

In [28]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=summer_data,x="humidity",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=summer_data,x="tempmax",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=summer_data,x="tempmin",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=summer_data,x="windspeed",kde=True,ax=axs[1,1],color='orange')
```

Out[28]:

```
<AxesSubplot:xlabel='windspeed', ylabel='Count'>
```

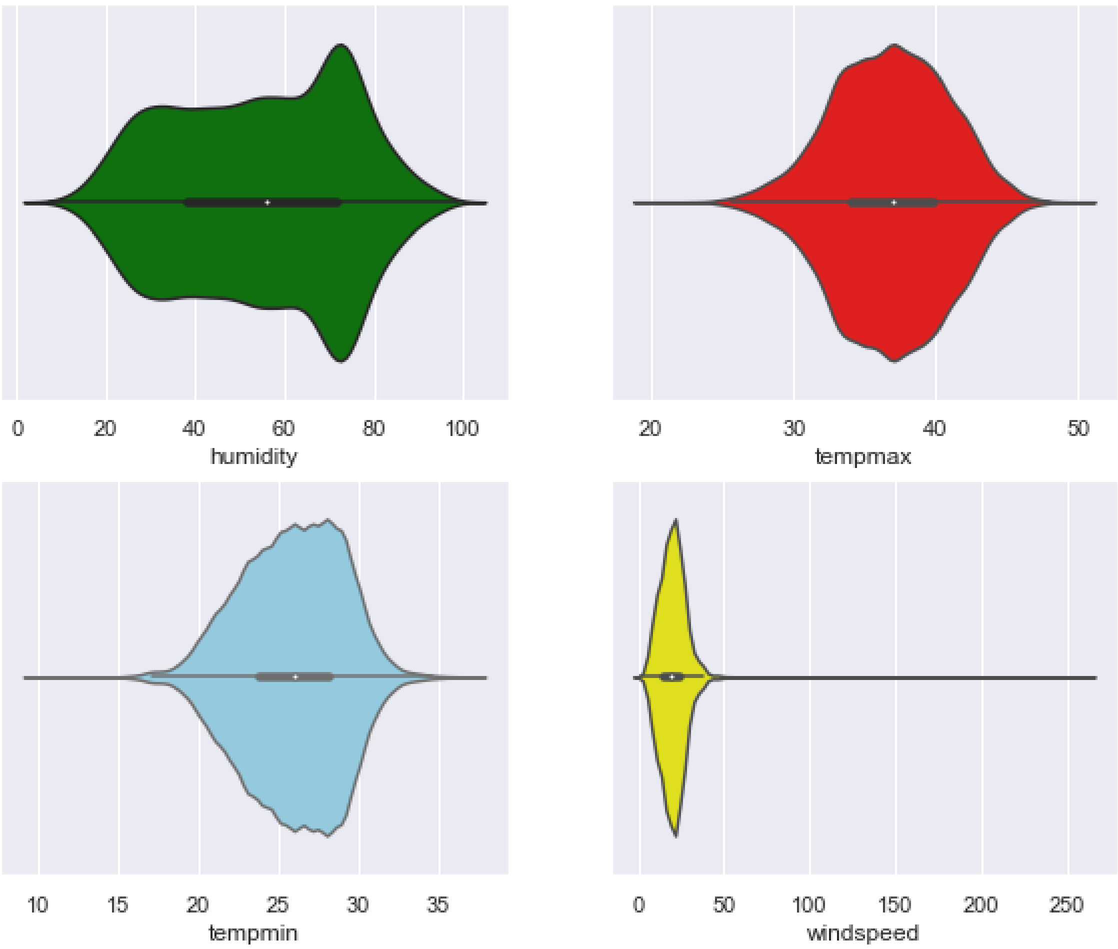


In [29]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.violinplot(data=summer_data,x="humidity",kde=True,ax=axs[0,0],color='green')
sns.violinplot(data=summer_data,x="tempmax",kde=True,ax=axs[0,1],color='red')
sns.violinplot(data=summer_data,x="tempmin",kde=True,ax=axs[1,0],color='skyblue')
sns.violinplot(data=summer_data,x="windspeed",kde=True,ax=axs[1,1],color='yellow')
```

Out[29]:

```
<AxesSubplot:xlabel='windspeed'>
```

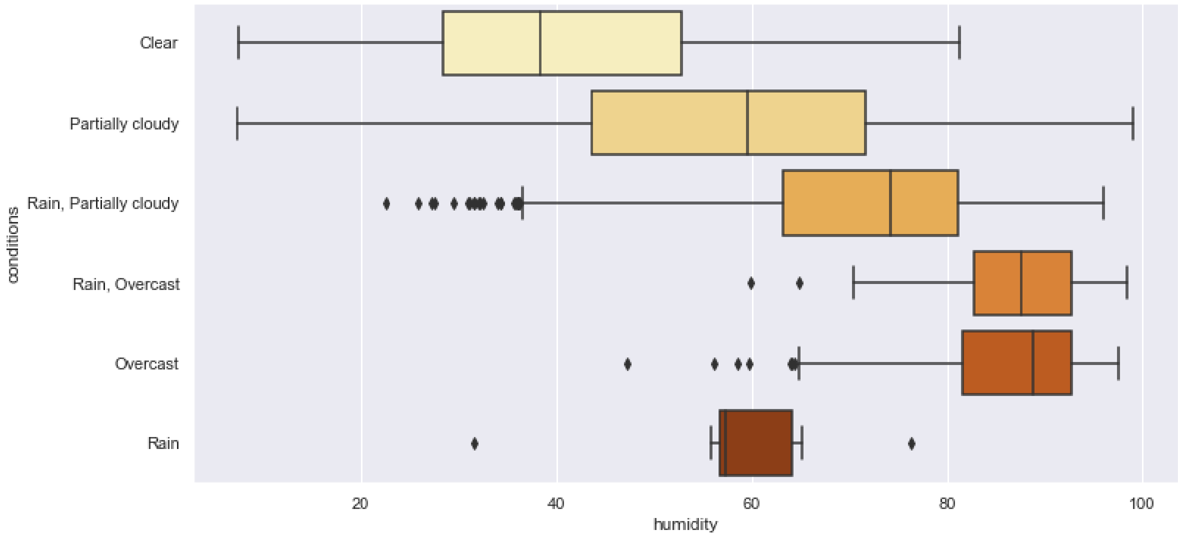


In [30]:

```
plt.figure(figsize=(12,6))
sns.boxplot("humidity","conditions",data=summer_data,palette="YlOrBr")
```

Out[30]:

```
<AxesSubplot:xlabel='humidity', ylabel='conditions'>
```

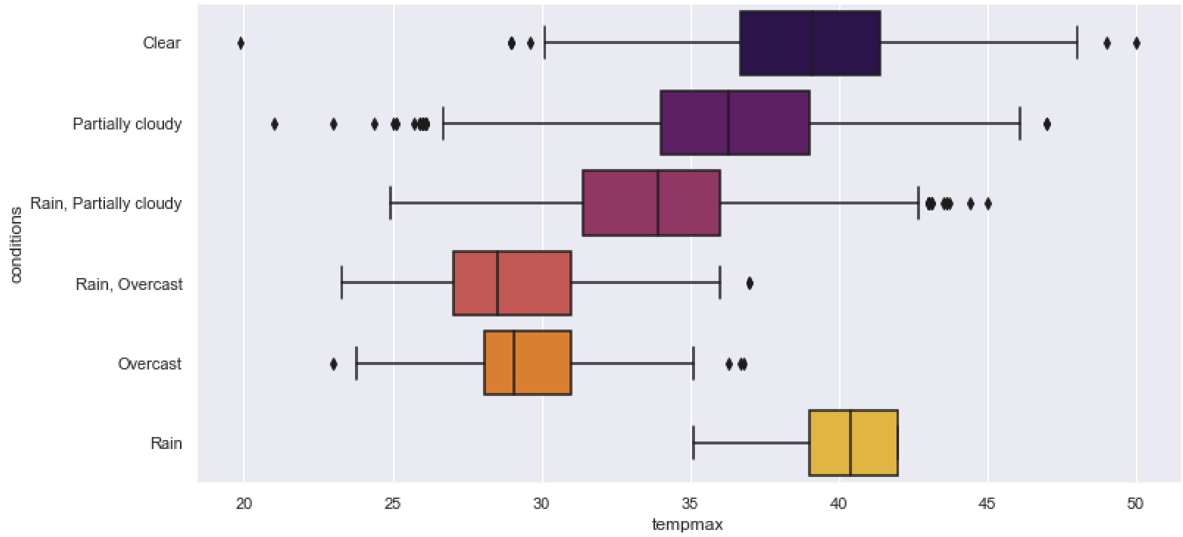


In [31]:

```
plt.figure(figsize=(12,6))
sns.boxplot("tempmax","conditions",data=summer_data,palette="inferno")
```

Out[31]:

<AxesSubplot:xlabel='tempmax', ylabel='conditions'>

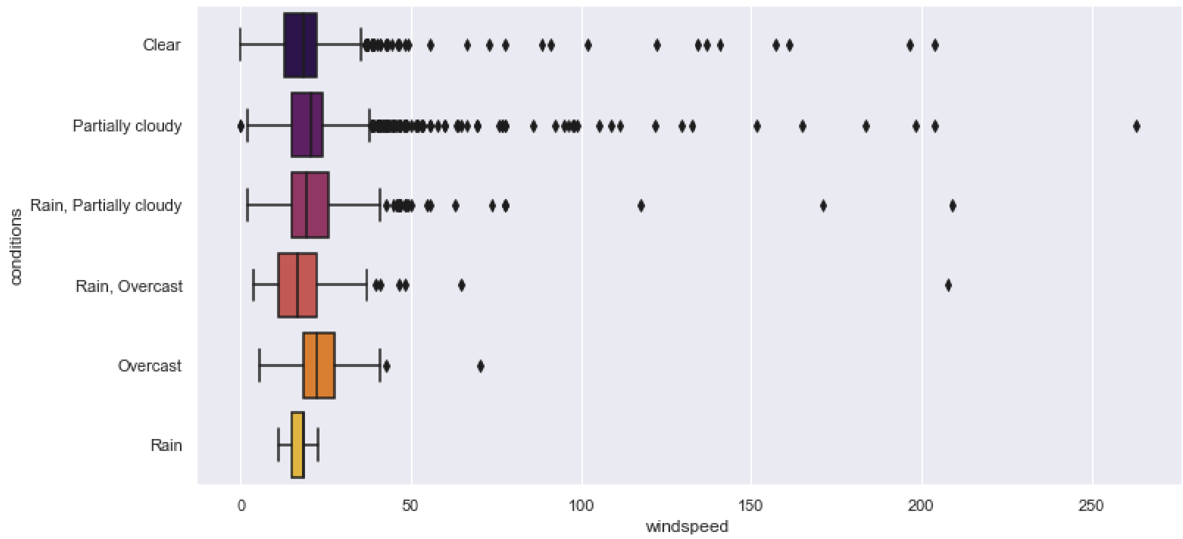


In [32]:

```
plt.figure(figsize=(12,6))
sns.boxplot("windspeed","conditions",data=summer_data,palette="inferno")
```

Out[32]:

<AxesSubplot:xlabel='windspeed', ylabel='conditions'>

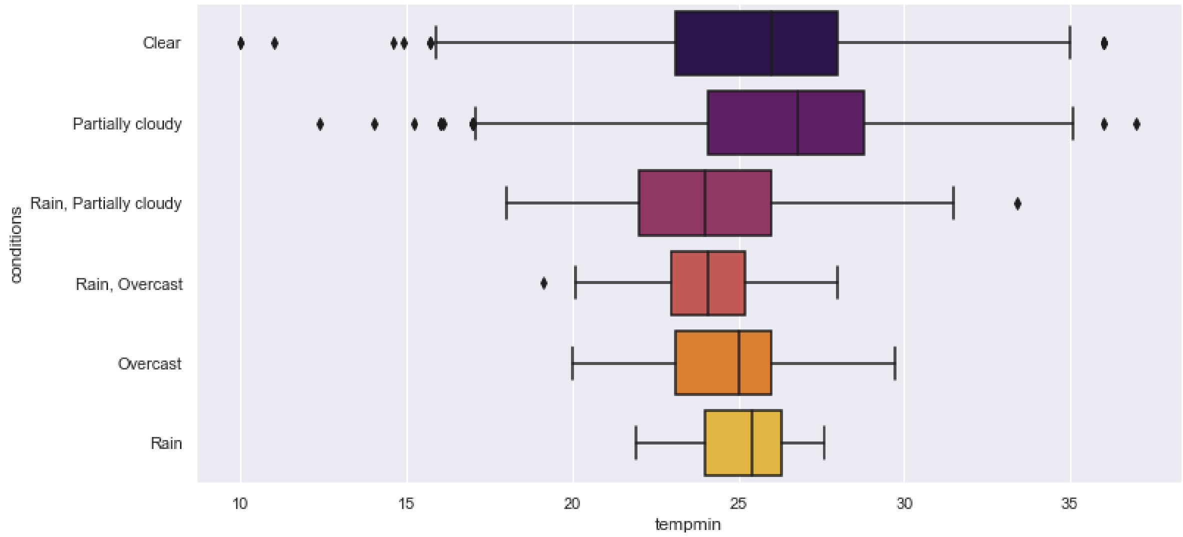


In [33]:

```
plt.figure(figsize=(12,6))
sns.boxplot("tempmin","conditions",data=summer_data,palette="inferno")
```

Out[33]:

```
<AxesSubplot:xlabel='tempmin', ylabel='conditions'>
```

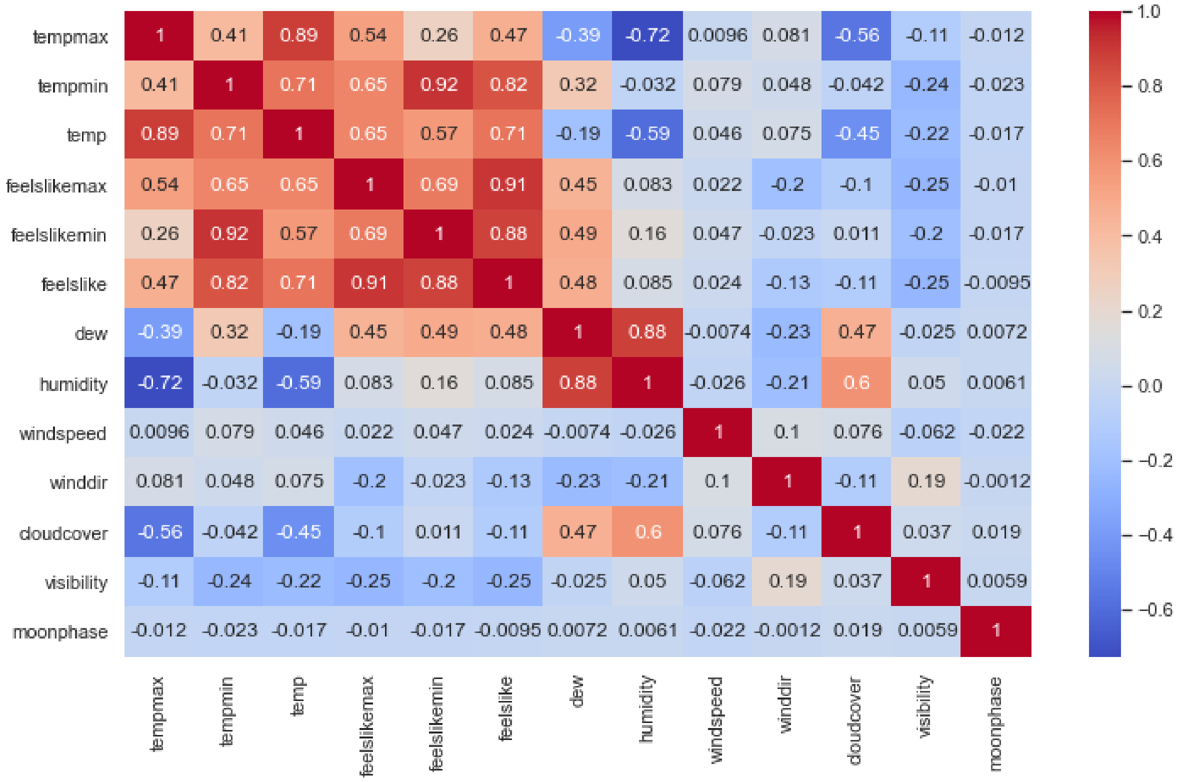


In [34]:

```
plt.figure(figsize=(12,7))
sns.heatmap(summer_data.corr(),annot=True,cmap='coolwarm')
```

Out[34]:

<AxesSubplot:>



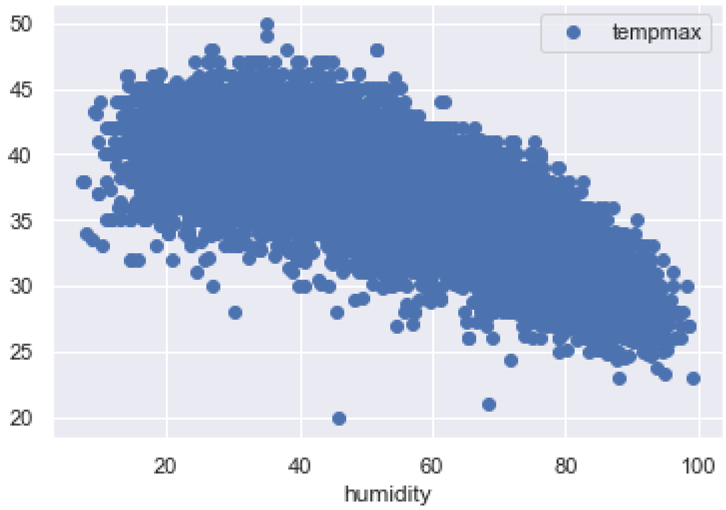
In [36]:

```
from scipy import stats
```

In [39]:

```
summer_data.plot("humidity","tempmax",style='o')
print("Pearson correlation:",summer_data["humidity"].corr(summer_data["tempmax"]))
print("T Test and P value:",stats.ttest_ind(summer_data["humidity"],summer_data["tempmax
```

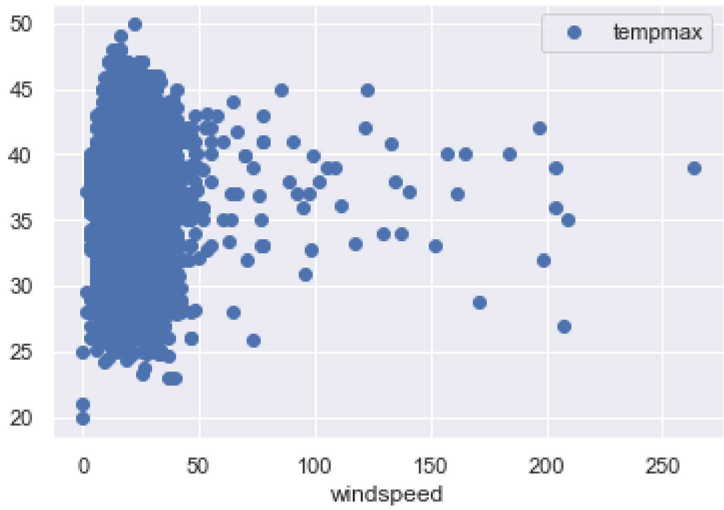
```
Pearson correlation: -0.724123660887335
T Test and P value: Ttest_indResult(statistic=104.69321537002257, pvalue=
0.0)
```



In [40]:

```
summer_data.plot("windspeed","tempmax",style='o')
print("Pearson correlation:",summer_data["windspeed"].corr(summer_data["tempmax"]))
print("T Test and P value:",stats.ttest_ind(summer_data["windspeed"],summer_data["tempma
```

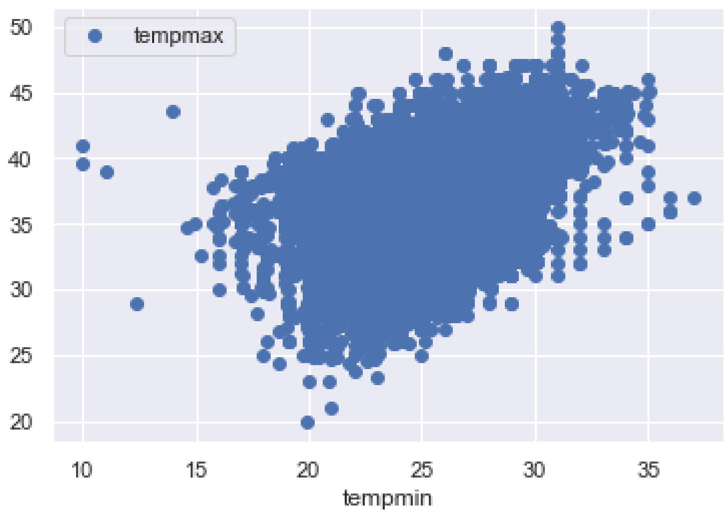
```
Pearson correlation: 0.009607384829183989
T Test and P value: Ttest_indResult(statistic=-182.368393165443, pvalue=0.
0)
```



In [41]:

```
summer_data.plot("tempmin","tempmax",style='o')
print("Pearson correlation:",summer_data["tempmin"].corr(summer_data["tempmax"]))
print("T Test and P value:",stats.ttest_ind(summer_data["tempmin"],summer_data["tempmax"
```

```
Pearson correlation: 0.41026701973380403
T Test and P value: Ttest_indResult(statistic=-250.82583693772827, pvalue=
0.0)
```



In [44]:

```
df=summer_data.drop(['Date', 'sunrise', 'sunset', 'description'],axis=1)
```

In [45]:

```
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 [46]:

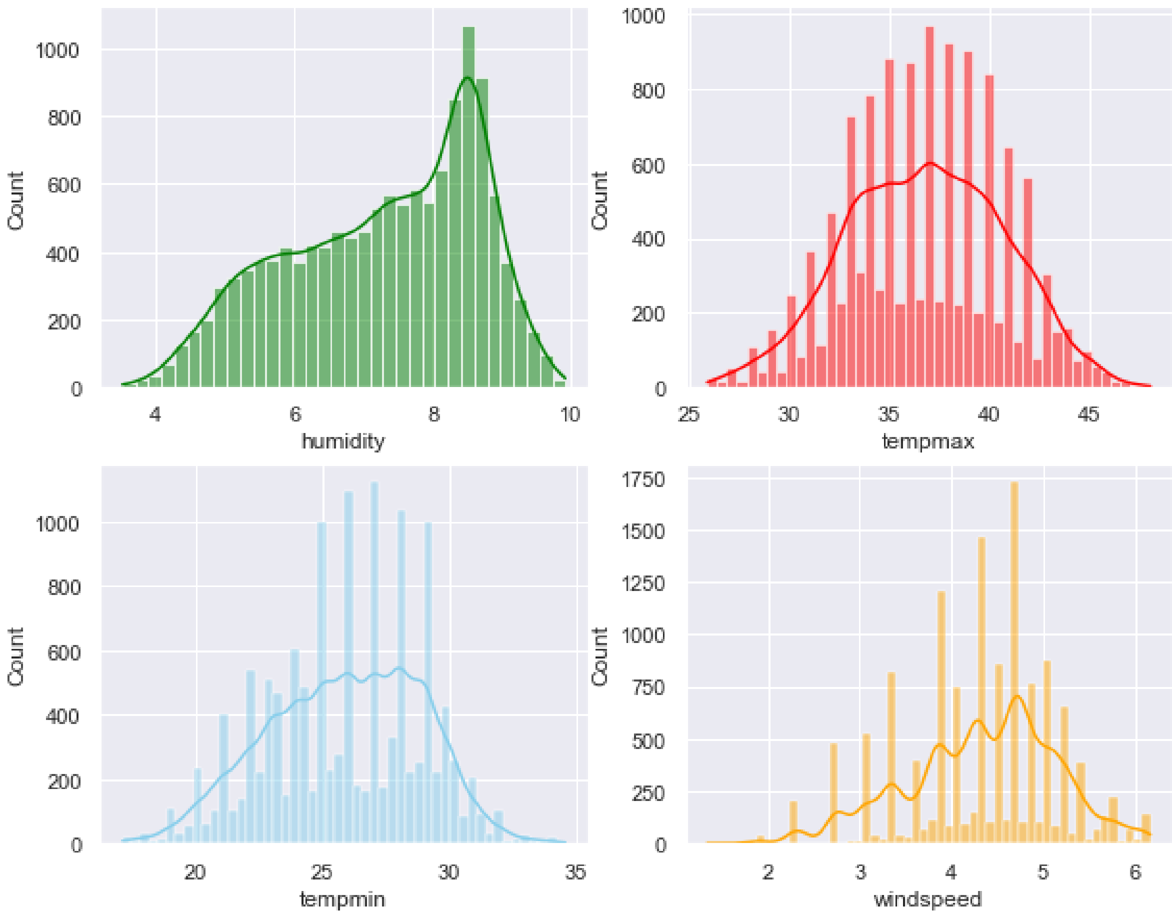
```
df.humidity=np.sqrt(df.humidity)
df.windspeed=np.sqrt(df.windspeed)
```

In [47]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=df,x="humidity",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=df,x="tempmax",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=df,x="tempmin",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=df,x="windspeed",kde=True,ax=axs[1,1],color='orange')
```

Out[47]:

```
<AxesSubplot:xlabel='windspeed', ylabel='Count'>
```



In [48]:

```
df.head()
```

Out[48]:

	City	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	windsp
0	New Delhi	34.0	19.0	27.1	31.6	19.0	26.1	3.1	4.753946	4.774
4	New Delhi	38.8	21.0	29.9	37.1	21.0	28.9	8.1	5.371220	3.674
5	New Delhi	38.0	22.6	30.4	37.2	22.6	29.5	10.2	5.523586	3.847
6	New Delhi	36.0	23.4	29.6	34.6	23.4	28.7	9.7	5.629387	4.289
7	New Delhi	34.9	20.9	27.6	32.6	20.9	26.7	4.4	5.144900	3.987

In [49]:

```
df1 = df.drop(['City'], axis = 1)
```

In [50]:

```
df1.head()
```

Out[50]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	windspeed	w
0	34.0	19.0	27.1	31.6	19.0	26.1	3.1	4.753946	4.774935	
4	38.8	21.0	29.9	37.1	21.0	28.9	8.1	5.371220	3.674235	
5	38.0	22.6	30.4	37.2	22.6	29.5	10.2	5.523586	3.847077	
6	36.0	23.4	29.6	34.6	23.4	28.7	9.7	5.629387	4.289522	
7	34.9	20.9	27.6	32.6	20.9	26.7	4.4	5.144900	3.987480	

In [52]:

```
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [53]:

```
lc=LabelEncoder()
df1["conditions"]=lc.fit_transform(df1["conditions"])
```

In [54]:

```
df1.head()
```

Out[54]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	windspeed	w
0	34.0	19.0	27.1	31.6	19.0	26.1	3.1	4.753946	4.774935	
4	38.8	21.0	29.9	37.1	21.0	28.9	8.1	5.371220	3.674235	
5	38.0	22.6	30.4	37.2	22.6	29.5	10.2	5.523586	3.847077	
6	36.0	23.4	29.6	34.6	23.4	28.7	9.7	5.629387	4.289522	
7	34.9	20.9	27.6	32.6	20.9	26.7	4.4	5.144900	3.987480	

In [56]:

```
x=((df1.loc[:,df1.columns!="conditions"]).astype(int)).values[:,0:]
y=df1["conditions"].values
```

In [57]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [58]:

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

```
KNN Accuracy:85.39%
```

In [59]:

```
svm=SVC()
svm.fit(x_train,y_train)
print("SVM Accuracy:{:.2f}%".format(svm.score(x_test,y_test)*100))
```

```
SVM Accuracy:85.01%
```

In [60]:

```
gbc=GradientBoostingClassifier(subsample=0.5,n_estimators=450,max_depth=5,max_leaf_node
gbc.fit(x_train,y_train)
print("Gradient Boosting Accuracy:{:.2f}%".format(gbc.score(x_test,y_test)*100))
```

```
Gradient Boosting Accuracy:88.83%
```