Weather Prediction

In [140…
```
from IPython.display import Image
Image('https://t4.ftcdn.net/jpg/02/66/38/15/360_F_266381525_alVrbw15u5EjhIpoqqa1eI5ghSf7hpz7.jpg')
```

Out[140…

COMPLETE EDA (EXPLORATORY DATA ANALYSIS)

meaning of eda: adv data analysis to convert raw data into clean for perfectvisualiation with the hep of EDA Technonogies.

EDA Technonogies 1.var identification 2.univariate 3.bivariate 4.Outlier Treatment 5.missing value treatment 6.varoable transformation

PROCESS 1.identify the var predictor(input) and target(output),and type of categorical 2.check whether the data is raw or clean 3.if data is raw,then clean the data ,apply missing value ,remove outlier 4.handling the missing value by mean for numerical data and medaian for categirical. 5.converting categorical to numerical

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
df=pd.read_csv(r"C:\Users\shaba\OneDrive\data scinece 2\seattle-weather.csv")
```

In [4]:
```
df
```

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
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

1461 rows × 6 columns

In [5]:
```
df.info
```

Out[5]:
```
<bound method DataFrame.info of            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
...          ...            ...       ...       ...   ...      ...
1456  2015-12-27            8.6       4.4       1.7   2.9     rain
1457  2015-12-28            1.5       5.0       1.7   1.3     rain
1458  2015-12-29            0.0       7.2       0.6   2.6      fog
1459  2015-12-30            0.0       5.6      -1.0   3.4      sun
1460  2015-12-31            0.0       5.6      -2.1   3.5      sun

[1461 rows x 6 columns]>
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['date', 'precipitation', 'temp_max', 'temp_min', 'wind', 'weather'], dtype='object')
```

In [7]:
```
df.head()
```

Out[7]:

	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 [8]:
```
df.tail()
```

Out[8]:

	date	precipitation	temp_max	temp_min	wind	weather
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

In [9]:
```
df.describe
```

Out[9]:
```
<bound method NDFrame.describe of            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
...          ...            ...       ...       ...   ...      ...
1456  2015-12-27            8.6       4.4       1.7   2.9     rain
1457  2015-12-28            1.5       5.0       1.7   1.3     rain
1458  2015-12-29            0.0       7.2       0.6   2.6      fog
1459  2015-12-30            0.0       5.6      -1.0   3.4      sun
1460  2015-12-31            0.0       5.6      -2.1   3.5      sun

[1461 rows x 6 columns]>
```

In [10]:
```
df.describe()
```

Out[10]:

	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 [11]:
```
df.columns
```

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

In [12]:
```
df.isnull().sum()
```

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

1.variable identificatiion

Variable identification in the dataset is very important to choose right machine learning algorithm. Variable or attribute or feature are split into 2 types ✉ 🎵 Independent Variable | Non target variable | Non predicted variable 🎵 Dependent Variable | target variable | Predicted variable In seattle-weather ,it has 6 columns and in that weather is a dependent var and other are independent var.

In [14]:
```
df.head()
```

Out[14]:

	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

2.Univariate Analysis

Visualize the graph using one variable is called Univariate Analysis

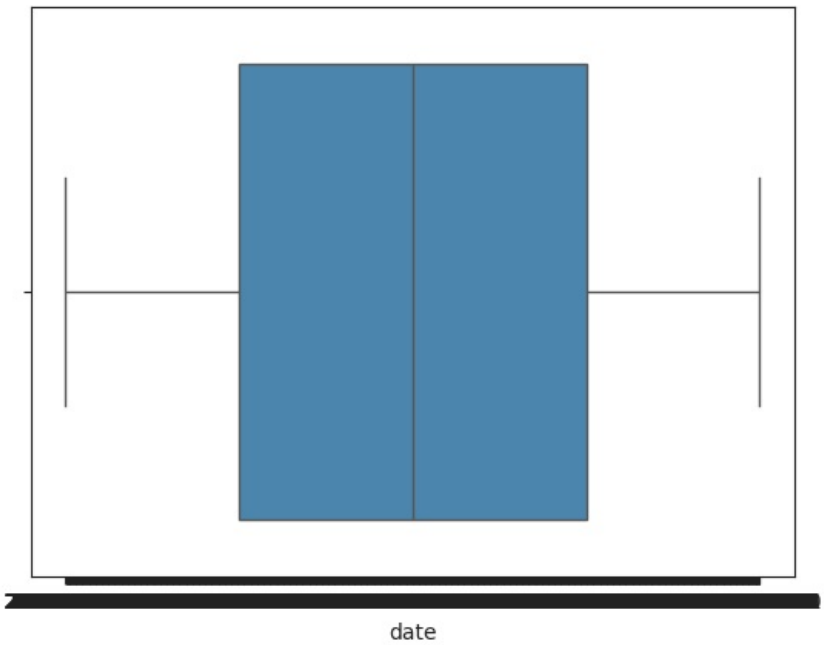
In [16]:
```
import seaborn as sns
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['date', 'precipitation', 'temp_max', 'temp_min', 'wind', 'weather'], dtype='object')
```

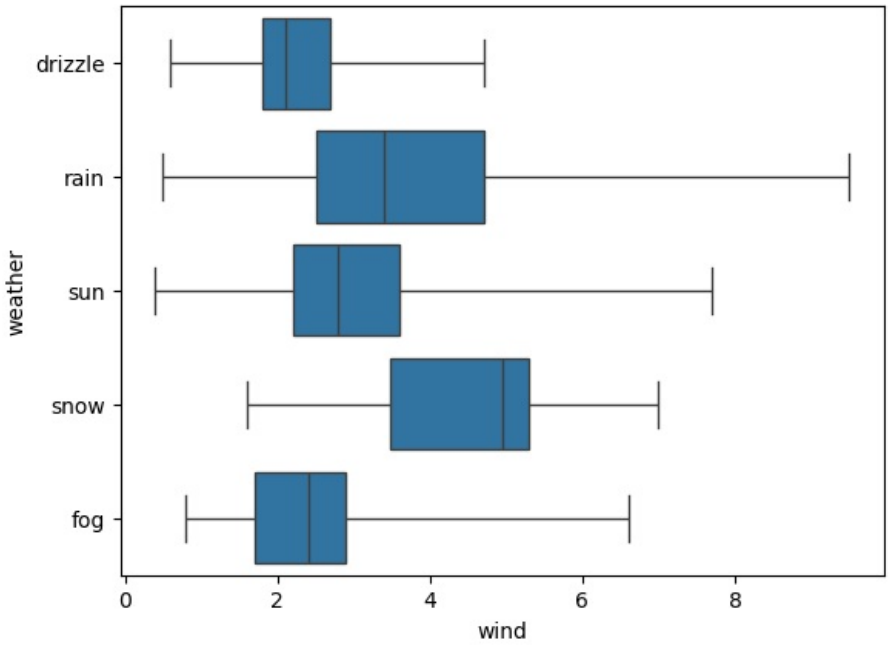
In [18]:
```
sns.boxplot(x=df['date'])
```

Out[18]:
```
<Axes: xlabel='date'>
```



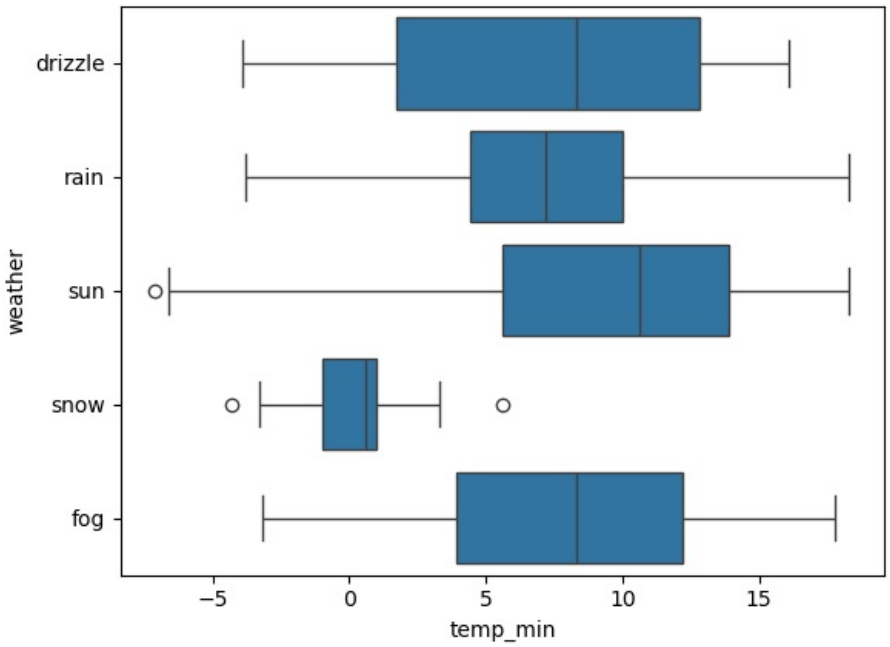
In [19]:
```
sns.boxplot(data=df, x="wind", y="weather", whis=(0, 100))
```

Out[19]:
```
<Axes: xlabel='wind', ylabel='weather'>
```



In [20]:
```
sns.boxplot(data=df, x="temp_min", y="weather")
```

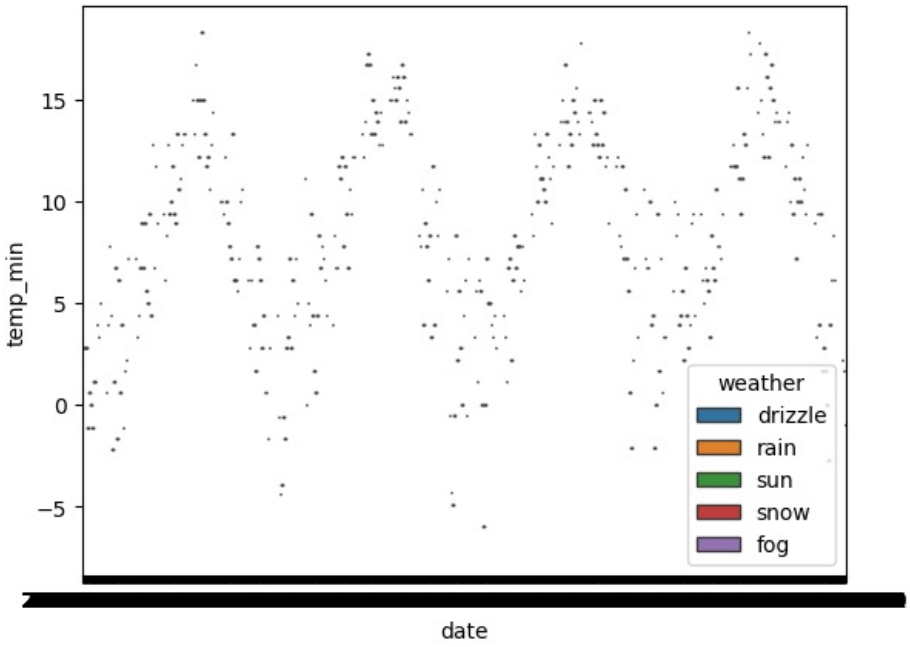
Out[20]: <Axes: xlabel='temp_min', ylabel='weather'>



In [21]:
```
import warnings
warnings.filterwarnings('ignore')
```

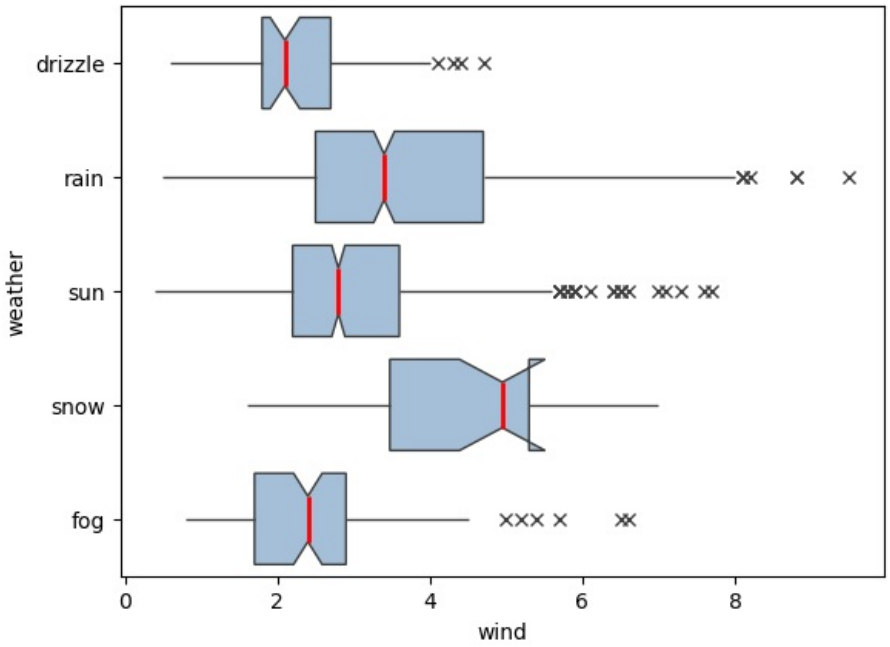
In [22]:
```
sns.boxplot(data=df, x="date", y="temp_min",hue='weather')
```

Out[22]: <Axes: xlabel='date', ylabel='temp_min'>



In [23]:
```
sns.boxplot(
    data=df, x="wind", y="weather",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.3, .5, .7, .5)},
    medianprops={"color": "r", "linewidth": 2},
)
```

Out[23]: <Axes: xlabel='wind', ylabel='weather'>



sns.boxplot(data=df, x="precipitation", y="weather",fill=False, gap=.1)

3.Bivariate Analysis

Visualize the graph using 2 variable is Bivariate Analysis Visualize the graph using more than 2 variable or many variables is Multivariate analysis Relation Between 2 variable – CORELATION Below is the pattern of corelation. Corelation is ranging from -1 to 1 0 to 1 ✉ Positive

corelation -1 to 0 ✉ Negative Corelation 0 ✉ No Corelationcorrelation statistical measures that describes the relationship between two variable fun= df.corr()

In [25]:
```
df.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 [26]:
```
df
```

Out[26]:

	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
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

1461 rows × 6 columns

In [27]:
```
df.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
```

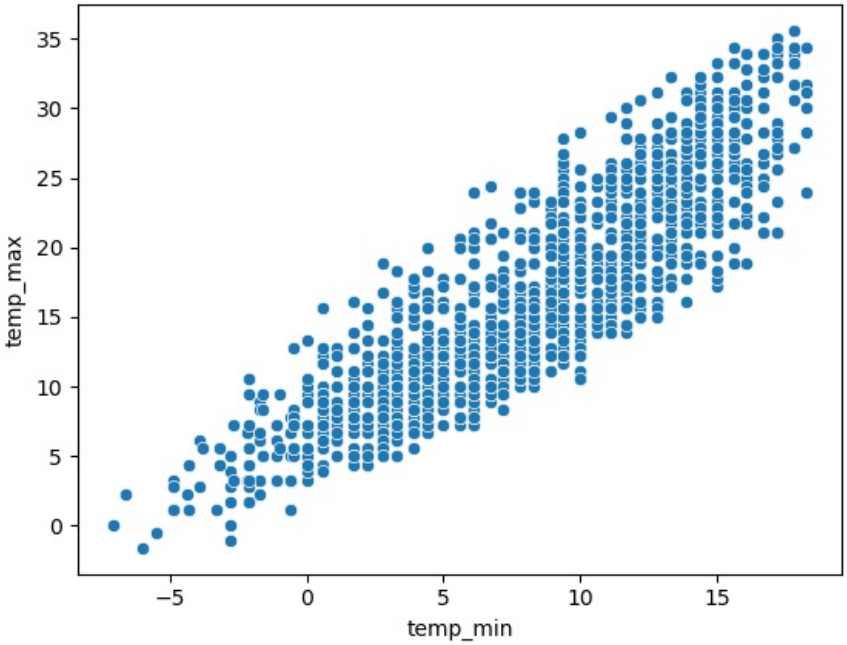
0 to 1 ✉ Positive corelation -1 to 0 ✉ Negative Corelation 0 ✉ No CorelationTYPES OF RELATIION 1.strong positive correlation 2.moderate positive correlation 3.no relation 4.strong negative correlaton 5.moderate negative correlation 6.curvilinear relation

In [28]:
```
correlation = df['temp_min'].corr(df['temp_max'])
print(correlation)
```

```
0.8756866637108167
```

In [29]:
```
sns.scatterplot(data=df, x="temp_min", y="temp_max")  #moderate positive  correlation
```

Out[29]:
```
<Axes: xlabel='temp_min', ylabel='temp_max'>
```

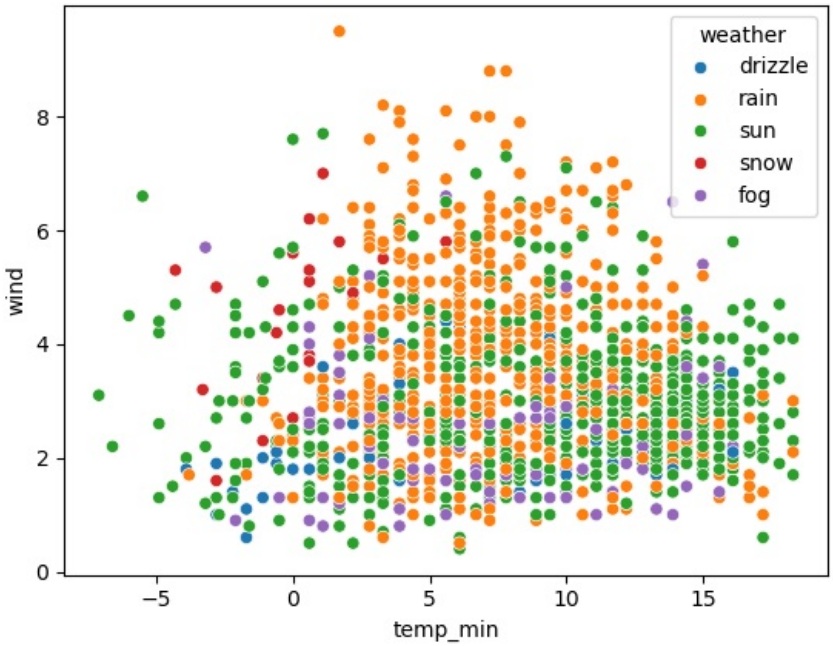


In [30]:
```
correlation = df['temp_min'].corr(df['wind'])
print(correlation)
```

```
-0.074185225373253
```

In [31]:
```
sns.scatterplot(
    data=df, x="temp_min", y="wind", hue="weather",
    sizes=(20, 200), legend="full"
)
```

Out[31]: <Axes: xlabel='temp_min', ylabel='wind'>

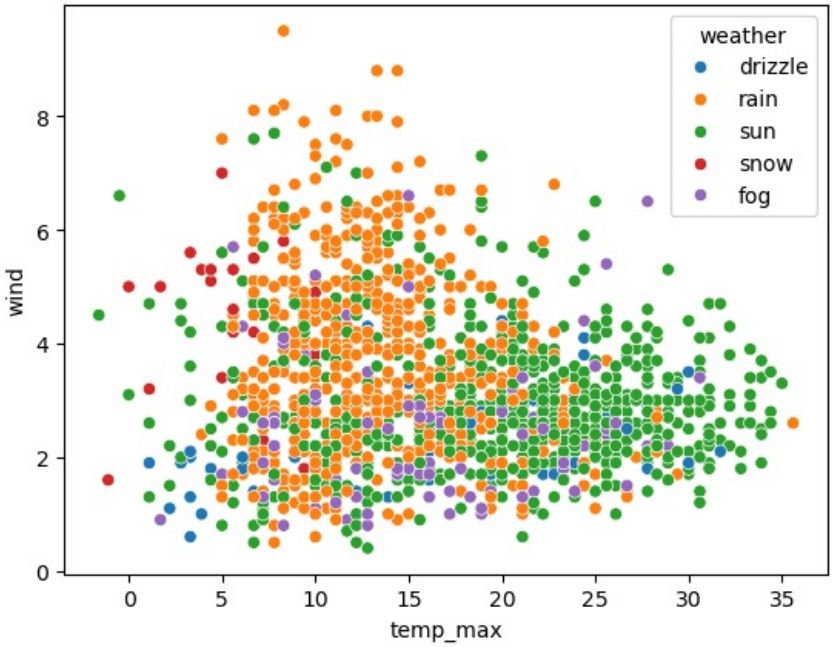


In [32]:
```
correlation = df['temp_max'].corr(df['wind'])
print(correlation)
```

```
-0.1648566348749546
```

In [33]:
```
sns.scatterplot(data=df, x="temp_max", y="wind", hue="weather")
```

Out[33]:
```
<Axes: xlabel='temp_max', ylabel='wind'>
```



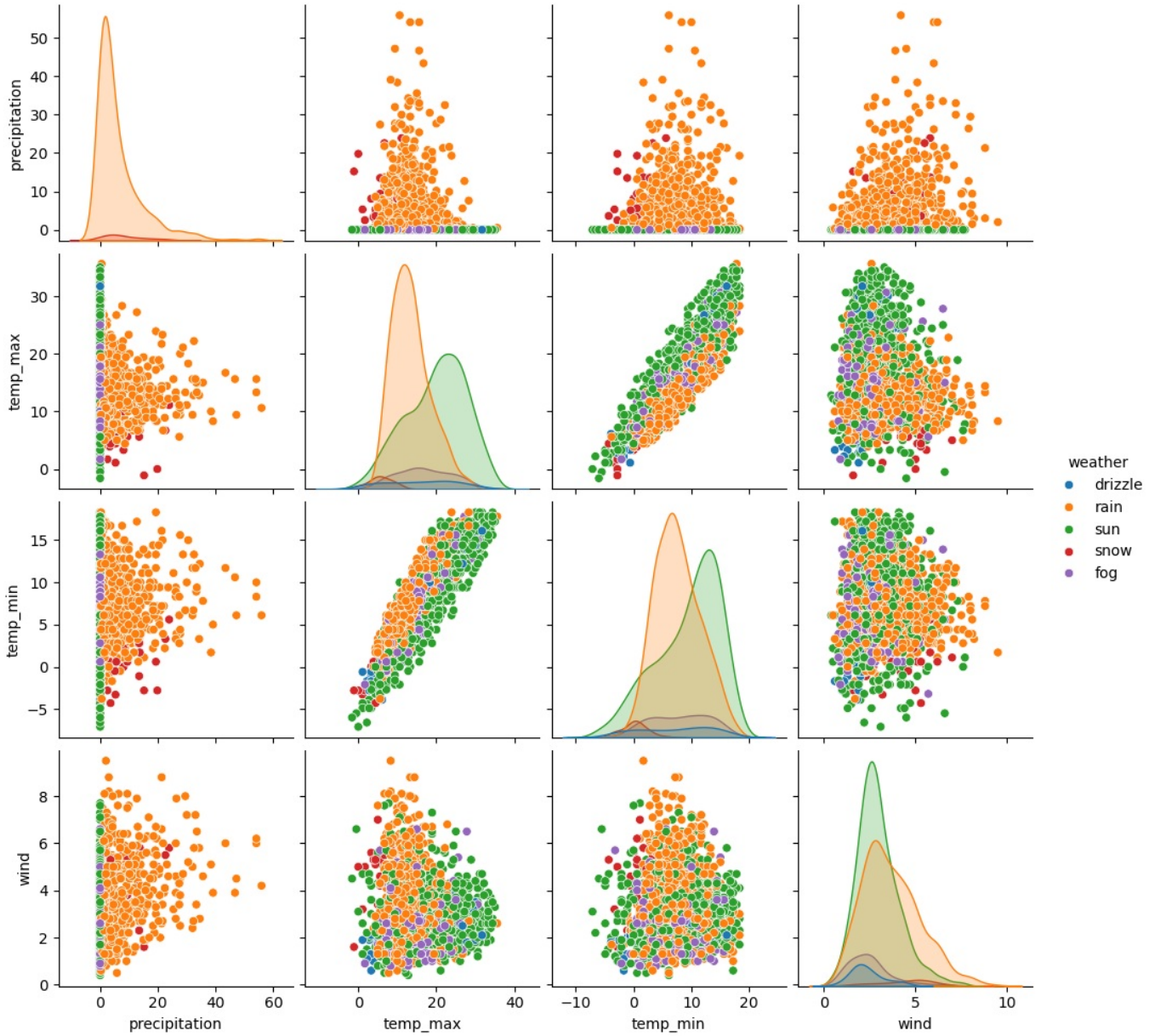
In [34]:
```
df
```

Out[34]:

	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
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

1461 rows × 6 columns

In [35]:
```
sns.pairplot(df,hue='weather')
plt.show()
```



4. Outlier detection

oulier means data which is far from others observation.

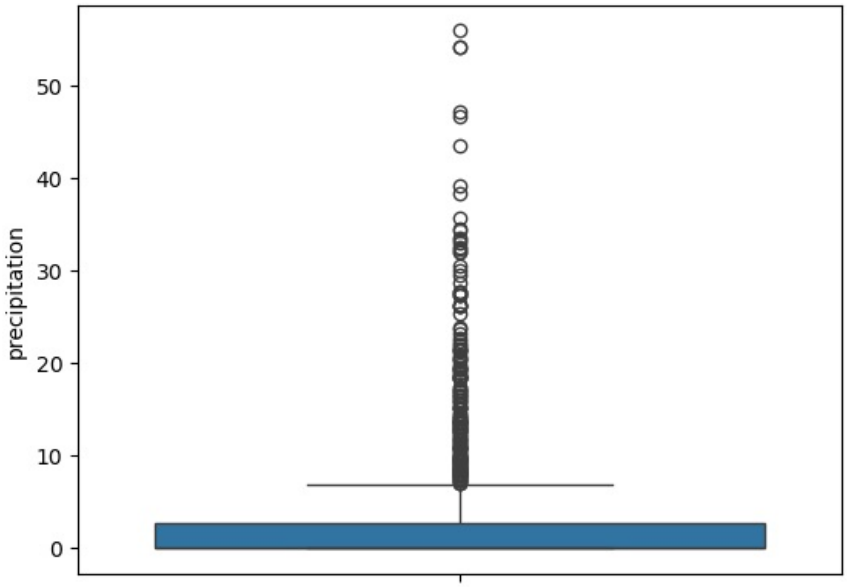
In [37]:
```
df.info()
```

```
<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
```

#first method to remove outlier #z-score method #find lowerlimit and upper limit if data goes out of these two that data will be outlier

In [38]:
```
sns.boxplot(df['precipitation'])  #it has lots of outliers
```

Out[38]:
```
<Axes: ylabel='precipitation'>
```



In [39]:
```
lower_limit=df['precipitation'].mean()-3*df['precipitation'].std()
upper_limit=df['precipitation'].mean()+3*df['precipitation'].std()
print(lower_limit,upper_limit)
```

```
-17.01115107098252 23.070014862905865
```

In [40]:
```
df.loc[(df['precipitation']>upper_limit)|(df['precipitation']<lower_limit)]
```

Out[40]:

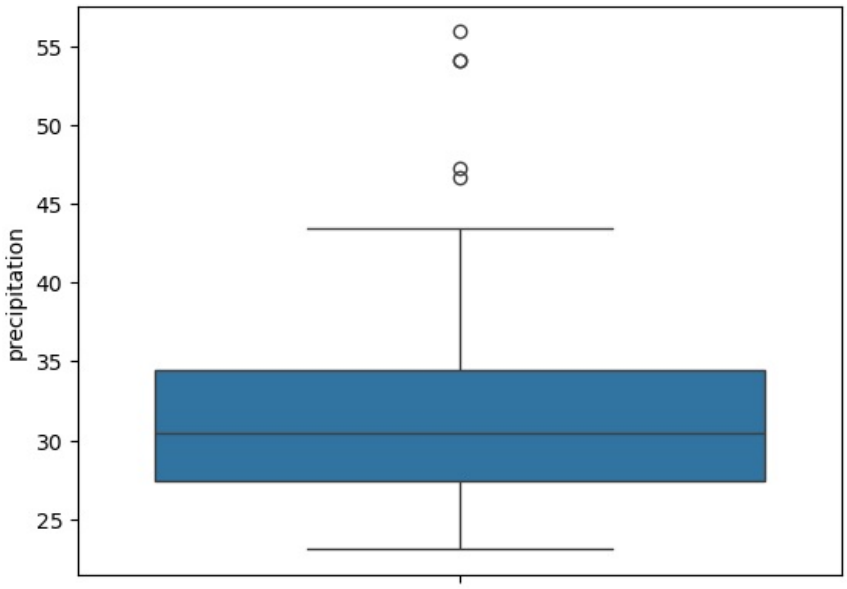
	date	precipitation	temp_max	temp_min	wind	weather
28	2012-01-29	27.7	9.4	3.9	4.5	rain
74	2012-03-15	23.9	11.1	5.6	5.8	snow
88	2012-03-29	27.4	10.0	6.1	4.4	rain
300	2012-10-27	23.1	14.4	9.4	5.1	rain
303	2012-10-30	34.5	15.0	12.2	2.8	rain
323	2012-11-19	54.1	13.3	8.3	6.0	rain
327	2012-11-23	32.0	9.4	6.1	2.4	rain
334	2012-11-30	35.6	15.0	7.8	4.6	rain
374	2013-01-09	38.4	10.0	1.7	5.1	rain
462	2013-04-07	39.1	8.3	5.0	3.9	rain
613	2013-09-05	27.7	20.0	15.6	2.5	rain
636	2013-09-28	43.4	16.7	11.7	6.0	rain
676	2013-11-07	30.0	11.1	10.0	7.2	rain
687	2013-11-18	26.2	12.8	9.4	3.9	rain
777	2014-02-16	26.4	9.4	3.9	7.9	rain
794	2014-03-05	46.7	15.6	10.6	3.9	rain
797	2014-03-08	32.3	12.8	6.7	2.7	rain
805	2014-03-16	27.7	10.6	4.4	3.8	rain
853	2014-05-03	33.3	15.0	8.9	3.4	rain
1025	2014-10-22	32.0	15.6	11.7	5.0	rain
1033	2014-10-30	25.4	15.6	11.1	3.2	rain
1062	2014-11-28	34.3	12.8	3.3	5.8	rain
1112	2015-01-17	26.2	13.3	3.3	2.8	rain
1131	2015-02-05	26.2	13.3	8.3	4.6	rain
1133	2015-02-07	23.6	12.2	9.4	4.6	rain
1169	2015-03-15	55.9	10.6	6.1	4.2	rain
1321	2015-08-14	30.5	18.3	15.0	5.2	rain
1336	2015-08-29	32.5	22.2	13.3	5.8	rain
1378	2015-10-10	28.7	21.1	13.3	4.7	rain
1399	2015-10-31	33.0	15.6	11.7	7.2	rain
1400	2015-11-01	26.2	12.2	8.9	6.0	rain
1412	2015-11-13	33.5	13.3	9.4	6.5	rain
1413	2015-11-14	47.2	9.4	6.1	4.5	rain
1416	2015-11-17	29.5	13.3	6.7	8.0	rain
1436	2015-12-07	27.4	11.1	8.3	3.4	rain
1437	2015-12-08	54.1	15.6	10.0	6.2	rain
1450	2015-12-21	27.4	5.6	2.8	4.3	rain

In [41]:
```
new_df=df.loc[(df['precipitation']>upper_limit)|(df['precipitation']<lower_limit)]
print('length old data',len(df))
print('length new data',len(new_df))
```

```
length old data 1461
length new data 37
```

In [42]:
```
sns.boxplot(new_df['precipitation'])  #it has lots of outliers
```

Out[42]:
```
<Axes: ylabel='precipitation'>
```



In [ ]:

In [43]:
```
lower_limit=df['temp_min'].mean()-3*df['temp_min'].std()
upper_limit=df['temp_min'].mean()+3*df['temp_min'].std()
print(lower_limit,upper_limit)
```

```
-6.834241834887223 23.30378324488038
```

In [44]:
```
df.loc[(df['temp_min']>upper_limit)|(df['temp_min']<lower_limit)]
```

Out[44]:

	date	precipitation	temp_max	temp_min	wind	weather
706	2013-12-07	0.0	0.0	-7.1	3.1	sun

In [45]:
```
new_df=df.loc[(df['temp_min']>upper_limit)|(df['temp_min']<lower_limit)]
print('length old data',len(df))
print('length new data',len(new_df))
```

```
length old data 1461
length new data 1
```

In [46]:
```
lower_limit=df['temp_max'].mean()-3*df['temp_max'].std()
upper_limit=df['temp_max'].mean()+3*df['temp_max'].std()
print(lower_limit,upper_limit)
```

```
-5.610191472094243 38.488357112066865
```

In [47]:
```
df.loc[(df['temp_max']>upper_limit)|(df['temp_max']<lower_limit)]
```

Out[47]:

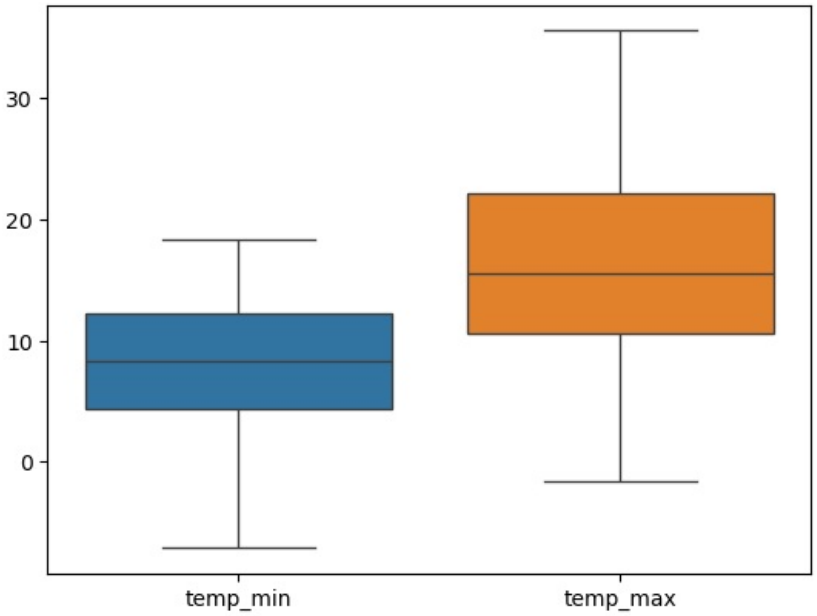
	date	precipitation	temp_max	temp_min	wind	weather

In [48]:
```
new_df=df.loc[(df['temp_max']>upper_limit)|(df['temp_max']<lower_limit)]
print('length old data',len(df))
print('length new data',len(new_df))
```

```
length old data 1461
length new data 0
```

In [49]:
```
sns.boxplot(df[['temp_min','temp_max']])
```

Out[49]:
```
<Axes: >
```



In [50]:
```
lower_limit=df['wind'].mean()-3*df['wind'].std()
upper_limit=df['wind'].mean()+3*df['wind'].std()
print(lower_limit,upper_limit)
```

```
-1.0723389685472133 7.554611384700533
```

In [51]:
```
df.loc[(df['temp_min']>upper_limit)|(df['temp_min']<lower_limit)]
```

Out[51]:

	date	precipitation	temp_max	temp_min	wind	weather
10	2012-01-11	0.0	6.1	-1.1	5.1	sun
11	2012-01-12	0.0	6.1	-1.7	1.9	sun
12	2012-01-13	0.0	5.0	-2.8	1.3	sun
14	2012-01-15	5.3	1.1	-3.3	3.2	snow
15	2012-01-16	2.5	1.7	-2.8	5.0	snow
...	...	...	...	...	...	...
1432	2015-12-03	12.7	15.6	7.8	5.9	rain
1436	2015-12-07	27.4	11.1	8.3	3.4	rain
1437	2015-12-08	54.1	15.6	10.0	6.2	rain
1438	2015-12-09	13.5	12.2	7.8	6.3	rain
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

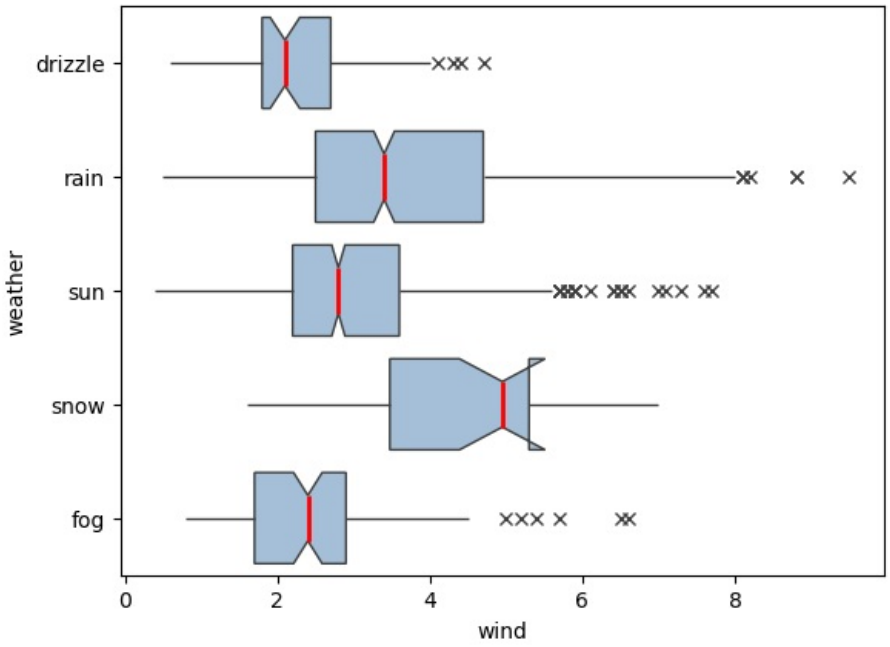
866 rows × 6 columns

In [52]:
```
new_df=df.loc[(df['wind']>upper_limit)|(df['wind']<lower_limit)]
print('length old data',len(df))
print('length new data',len(new_df))
```

```
length old data 1461
length new data 15
```

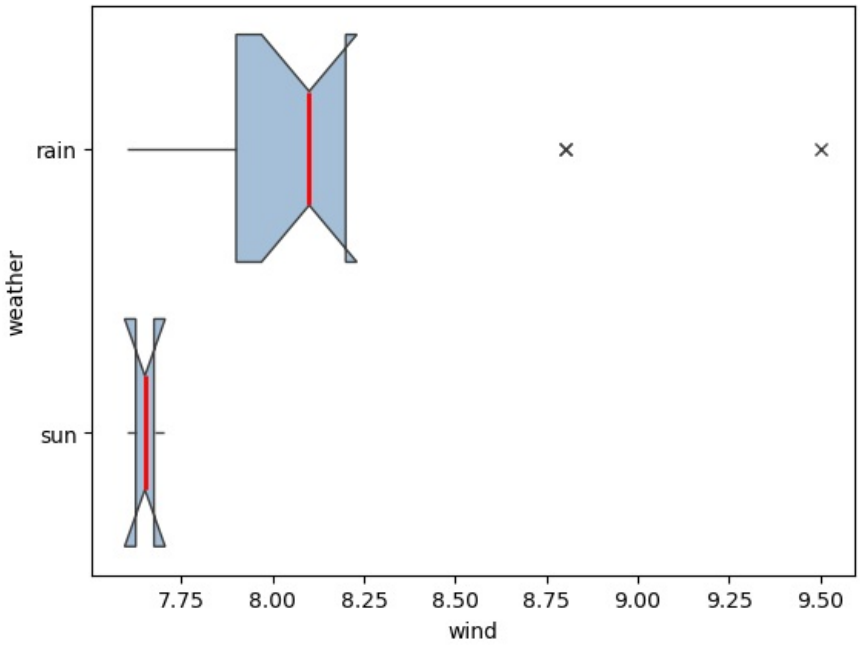
In [53]:
```
sns.boxplot(
    data=df, x="wind", y="weather",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.3, .5, .7, .5)},
    medianprops={"color": "r", "linewidth": 2},
)
```

Out[53]:
```
<Axes: xlabel='wind', ylabel='weather'>
```



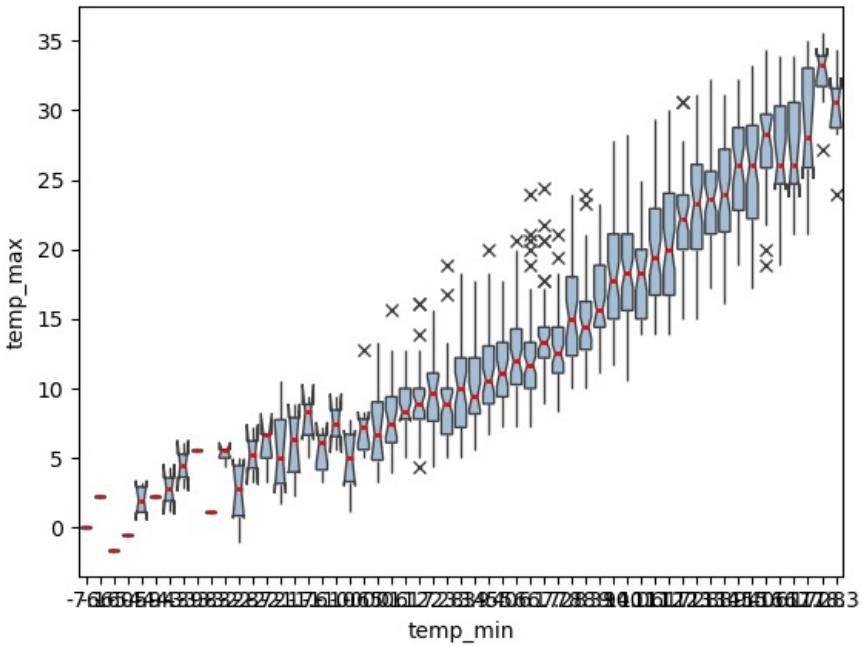
In [54]:
```
sns.boxplot(
    data=new_df, x="wind", y="weather",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.3, .5, .7, .5)},
    medianprops={"color": "r", "linewidth": 2},
)
```

Out[54]: <Axes: xlabel='wind', ylabel='weather'>



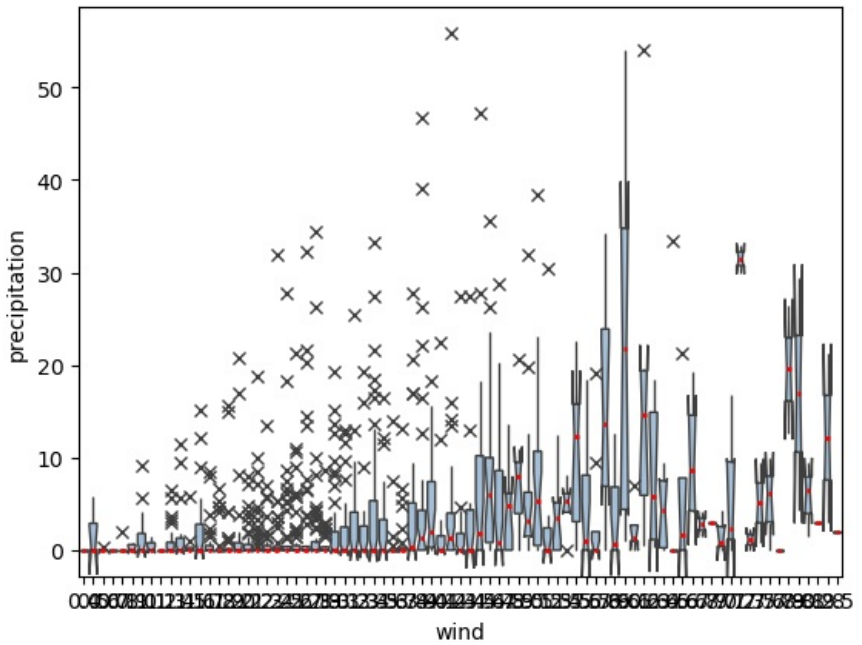
In [55]:
```
sns.boxplot(        # 'x' is outlier
    data=df, x="temp_min", y="temp_max",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.3, .5, .7, .5)},
    medianprops={"color": "r", "linewidth": 2},
)
```

Out[55]: <Axes: xlabel='temp_min', ylabel='temp_max'>



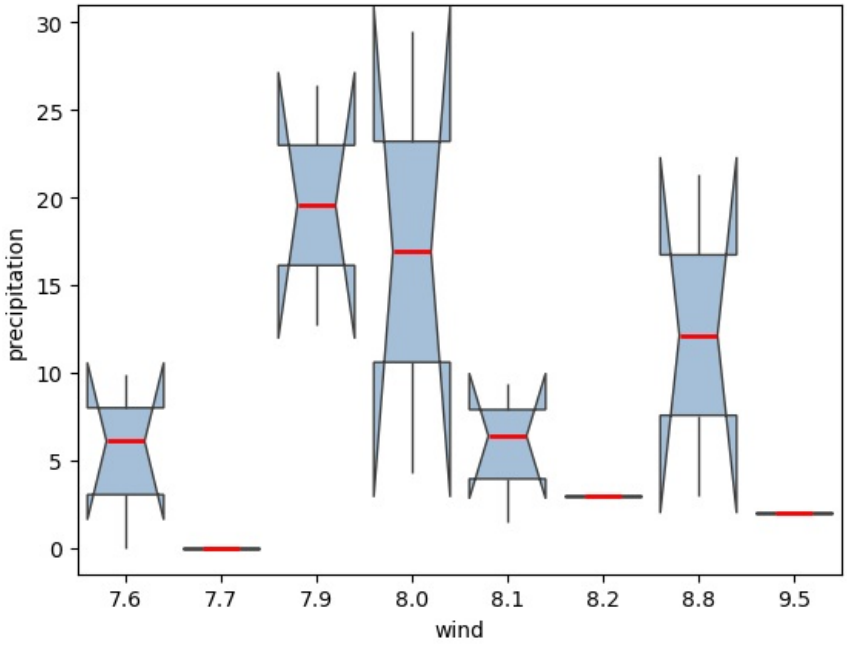
In [56]:
```
sns.boxplot(
    data=df, x="wind", y="precipitation",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.3, .5, .7, .5)},
    medianprops={"color": "r", "linewidth": 2},
)
```

Out[56]: <Axes: xlabel='wind', ylabel='precipitation'>



In [57]:
```
sns.boxplot(
    data=new_df, x="wind", y="precipitation",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.3, .5, .7, .5)},
    medianprops={"color": "r", "linewidth": 2},
)
```

Out[57]: <Axes: xlabel='wind', ylabel='precipitation'>



5.Missing value treatmenet

data is clean alredy

6.Variable Transformation

In [60]:
```
df['date']=df['date'].astype('category')
df['precipitation']=df['precipitation'].astype(int)
df['temp_max']=df['temp_max'].astype(int)
df['temp_min']=df['temp_min'].astype(int)
df['wind']=df['wind'].astype(int)
df['weather']=df['weather'].astype('category')
```

In [61]:
```
df.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   category
 1   precipitation  1461 non-null   int32
 2   temp_max       1461 non-null   int32
 3   temp_min       1461 non-null   int32
 4   wind           1461 non-null   int32
 5   weather        1461 non-null   category
dtypes: category(2), int32(4)
memory usage: 71.1 KB
```

A change in the weather is sufficient to recreate the world and ourselves.alina