logistic-regression-assignment

September 26, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
data = pd.read_csv(r"C:\\Users\\prana\\Documents\\Python data\\SeattleWeather.
↪csv")
data
```

[2]:
```
             DATE  PRCP  TMAX  TMIN   RAIN
0      01-01-1948  0.47    51    42   True
1      02-01-1948  0.59    45    36   True
2      03-01-1948  0.42    45    35   True
3      04-01-1948  0.31    45    34   True
4      05-01-1948  0.17    45    32   True
…             …   …     …     …     …
25546  10-12-2017  0.00    49    34  False
25547  11-12-2017  0.00    49    29  False
25548  12-12-2017  0.00    46    32  False
25549  13-12-2017  0.00    48    34  False
25550  14-12-2017  0.00    50    36  False

[25551 rows x 5 columns]
```

[3]:
```
data.head(5)
```

[3]:
```
         DATE  PRCP  TMAX  TMIN  RAIN
0  01-01-1948  0.47    51    42  True
1  02-01-1948  0.59    45    36  True
2  03-01-1948  0.42    45    35  True
3  04-01-1948  0.31    45    34  True
4  05-01-1948  0.17    45    32  True
```

[4]:
```
data.isna().sum()
```

[4]:
```
DATE    0
PRCP    3
TMAX    0
```

```
TMIN    0
RAIN    3
dtype: int64
```

[5]:
```
data[data["PRCP"].isna()]
```

[5]:
```
             DATE  PRCP  TMAX  TMIN RAIN
18415  02-06-1998   NaN    72    52  NaN
18416  03-06-1998   NaN    66    51  NaN
21067  05-09-2005   NaN    70    52  NaN
```

[6]:
```
data[data["RAIN"].isna()]
```

[6]:
```
             DATE  PRCP  TMAX  TMIN RAIN
18415  02-06-1998   NaN    72    52  NaN
18416  03-06-1998   NaN    66    51  NaN
21067  05-09-2005   NaN    70    52  NaN
```

[7]:
```
data = data.dropna()
```

[8]:
```
data.isna().sum()
```

[8]:
```
DATE    0
PRCP    0
TMAX    0
TMIN    0
RAIN    0
dtype: int64
```

[9]:
```
data.RAIN.value_counts()
```

[9]:
```
RAIN
False    14648
True     10900
Name: count, dtype: int64
```

[10]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[11]:
```
data["RAIN"] = le.fit_transform(data["RAIN"])
```

```
C:\Users\prana\AppData\Local\Temp\ipykernel_14568\66160035.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data["RAIN"] = le.fit_transform(data["RAIN"])
```

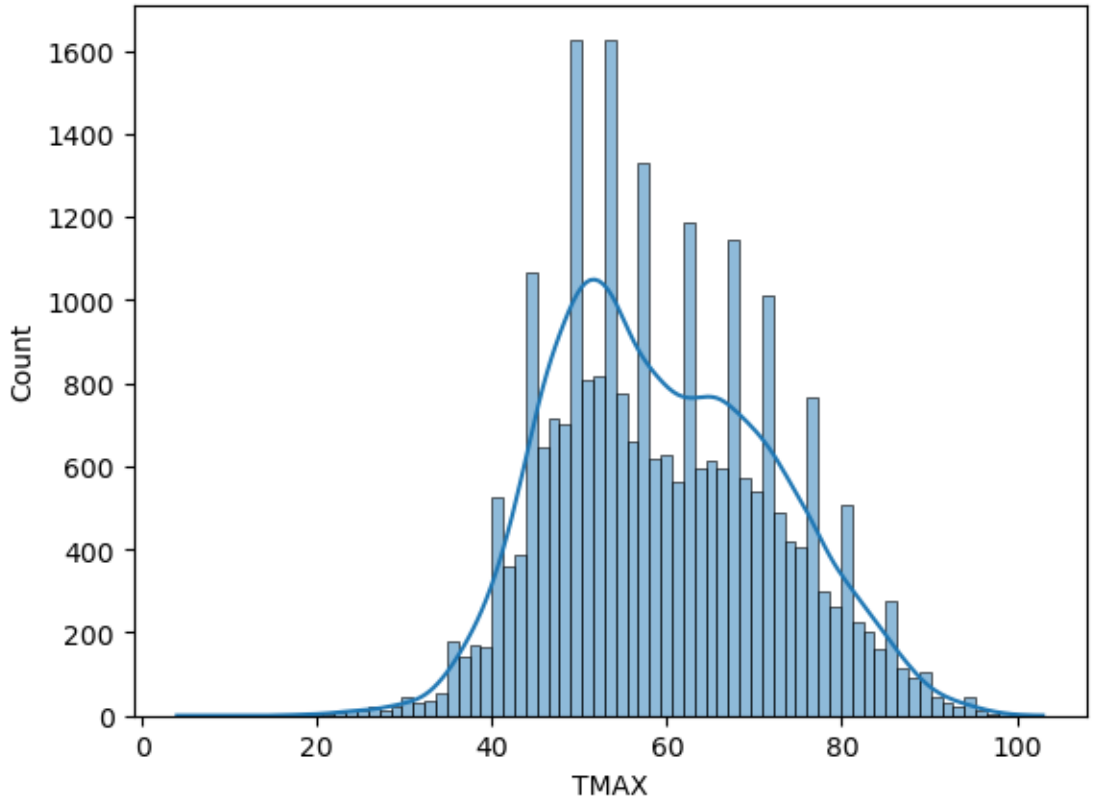
[12]:
```
data["RAIN"].dtype
```

[12]:
```
dtype('int32')
```

[13]:
```
import seaborn as sns
```

[14]:
```
#pip install seaborn --upgrade
```

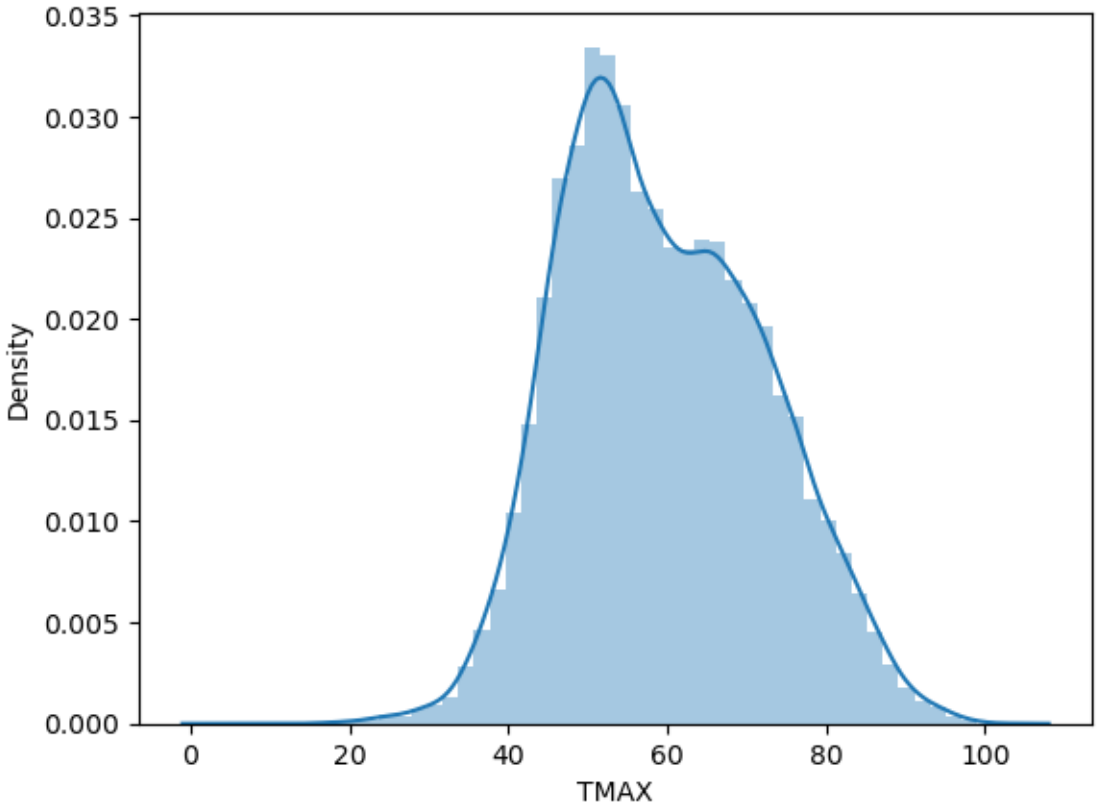
[15]:
```
sns.histplot(data["TMAX"], kde = True)
plt.show()
```



[16]:
```
sns.distplot(data["TMAX"], kde = True)
plt.show()
```

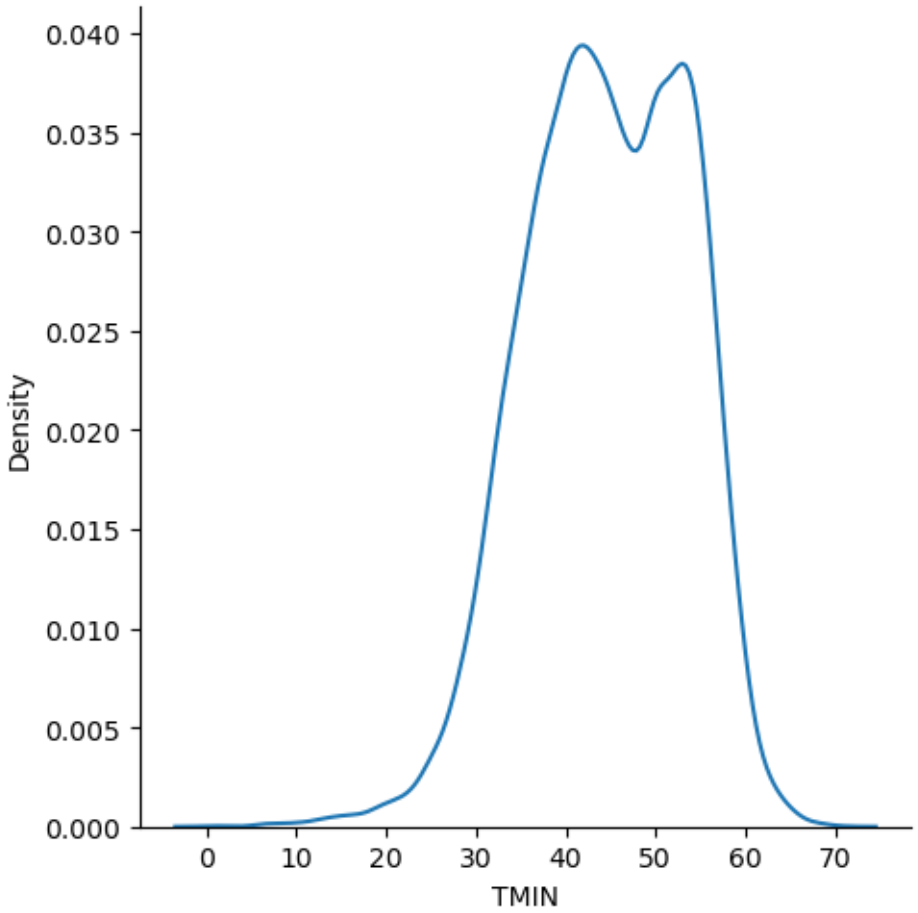
```
C:\Users\prana\AppData\Local\Temp\ipykernel_14568\2605151143.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

```
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data["TMAX"], kde = True)
```

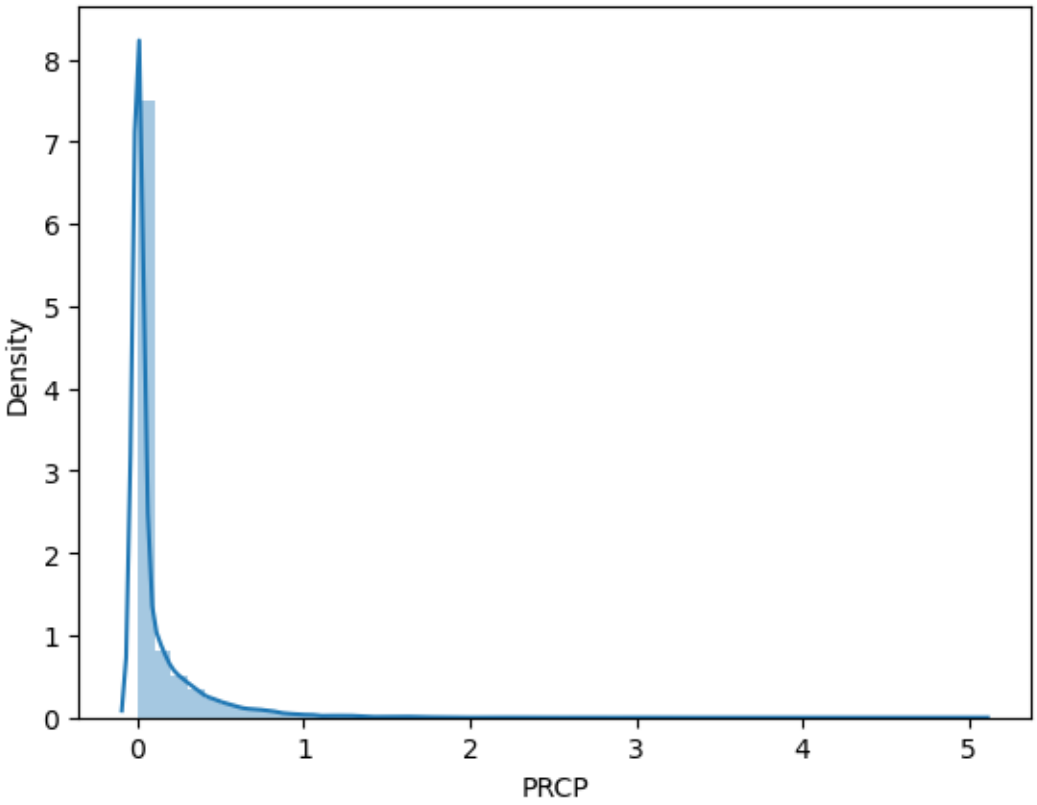


[17]:
```
sns.displot(data["TMIN"], kind ="kde" )
plt.show()
```

```
C:\Users\prana\anaconda3\New folder\Lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



[18]:
```
import warnings
warnings.filterwarnings("ignore")

sns.distplot(data["PRCP"])
plt.show()
```

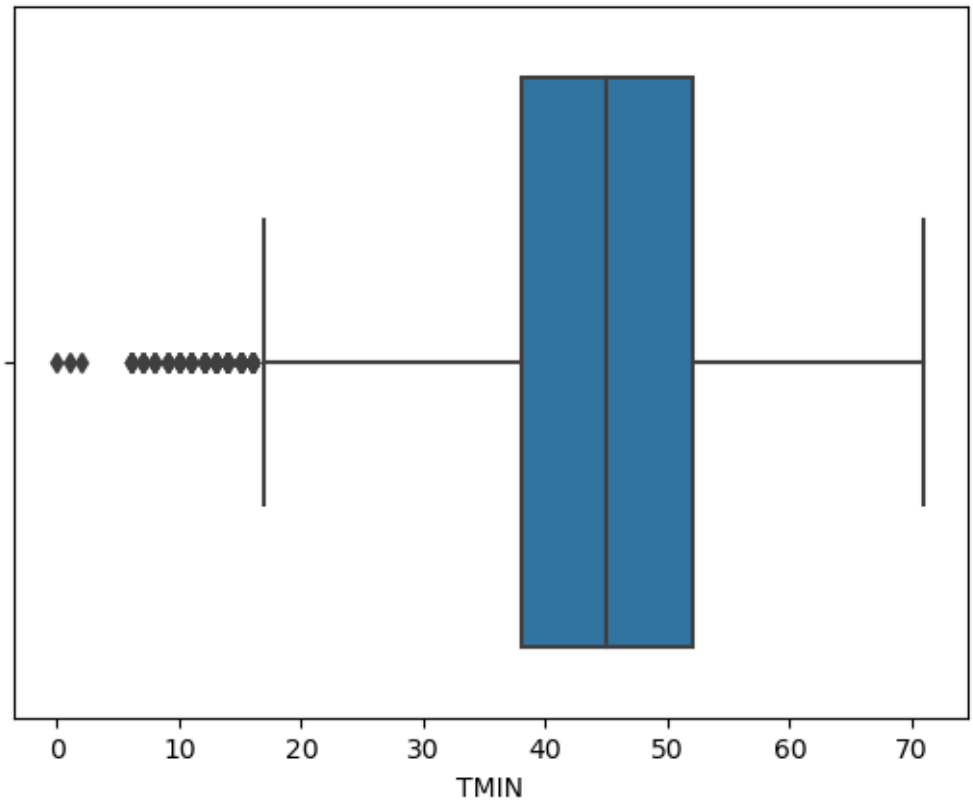


[19]:
```
data.describe()
```

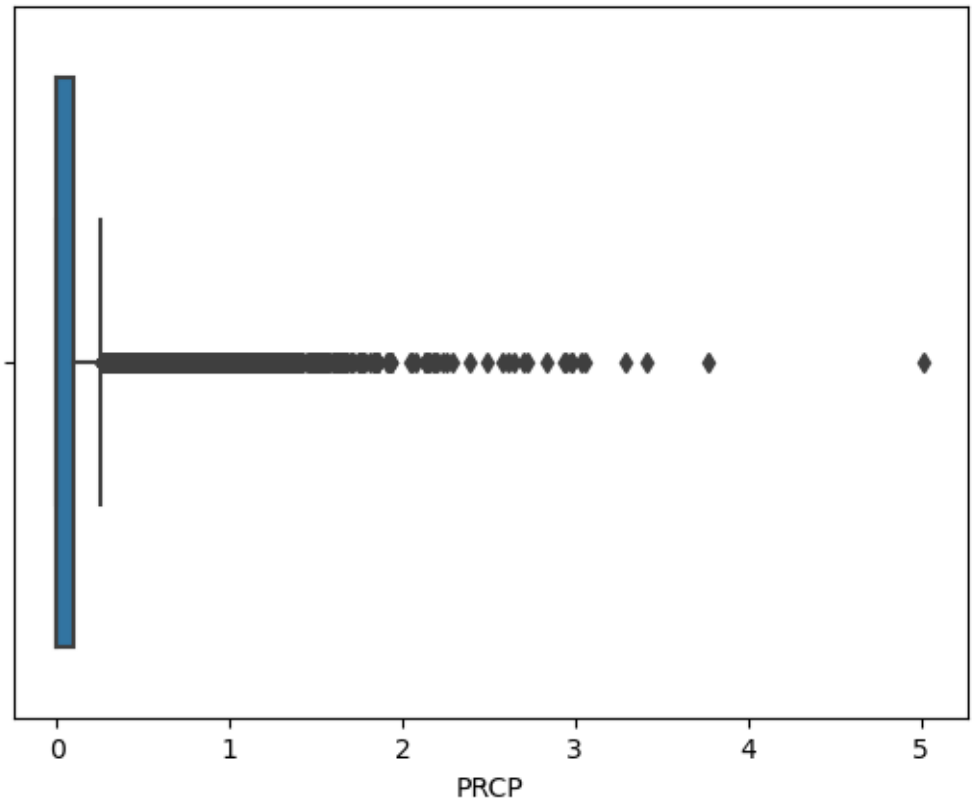
[19]:

	PRCP	TMAX	TMIN	RAIN
count	25548.000000	25548.000000	25548.000000	25548.000000
mean	0.106222	59.543056	44.513387	0.426648
std	0.239031	12.773265	8.893019	0.494600
min	0.000000	4.000000	0.000000	0.000000
25%	0.000000	50.000000	38.000000	0.000000
50%	0.000000	58.000000	45.000000	0.000000
75%	0.100000	69.000000	52.000000	1.000000
max	5.020000	103.000000	71.000000	1.000000

[22]:
```
sns.boxplot(x=data["TMIN"])
plt.show()
```



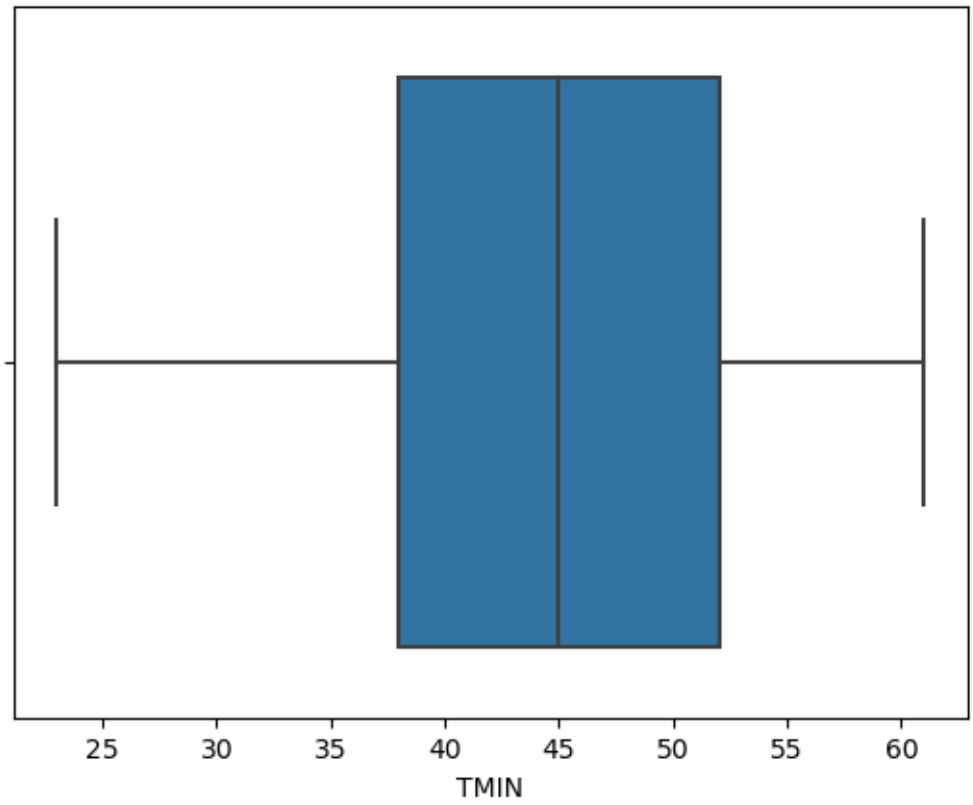
[21]:
```
sns.boxplot(x=data["PRCP"])
plt.show()
```



[23]:
```
upper_limit = data["TMIN"].quantile(0.99)
lower_limit = data["TMIN"].quantile(0.01)
```

[24]:
```
data["TMIN"] = np.where(data["TMIN"] < lower_limit, lower_limit, data["TMIN"])
data["TMIN"] = np.where(data["TMIN"] > upper_limit, upper_limit, data["TMIN"])
```

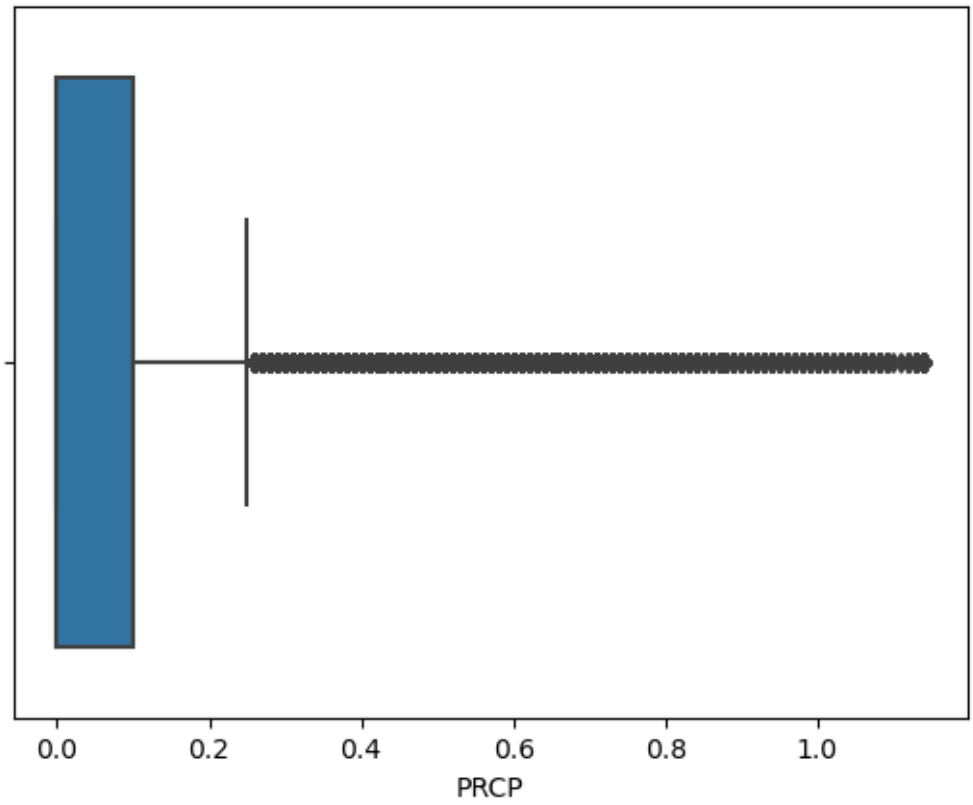
[25]:
```
sns.boxplot(x=data["TMIN"])
plt.show()
```



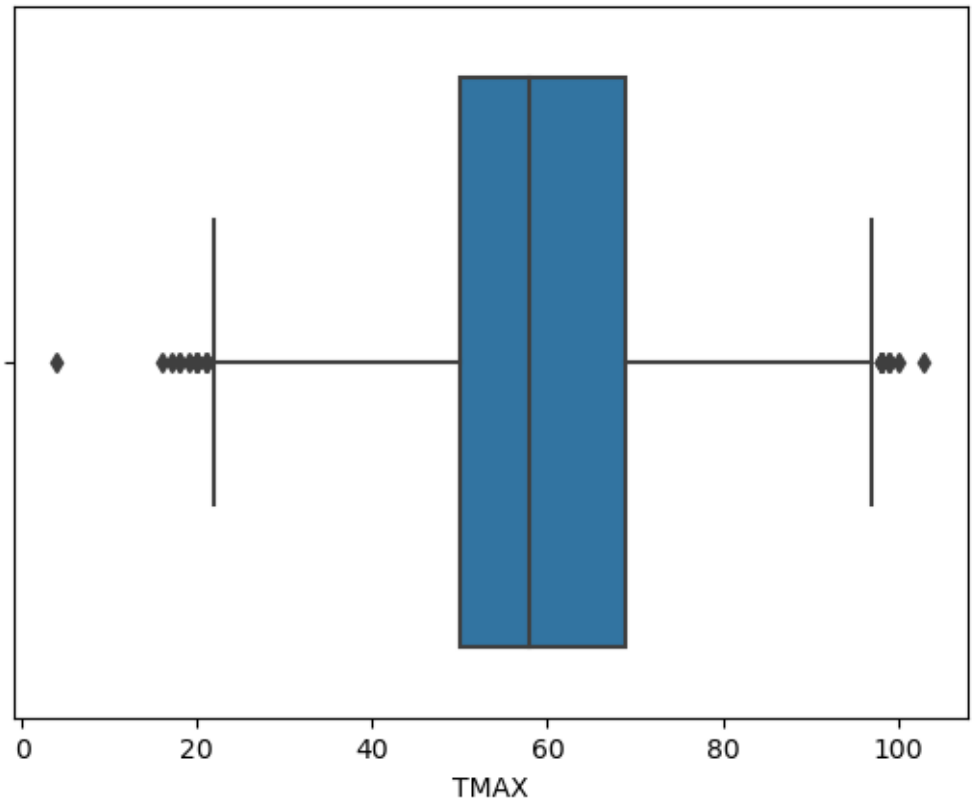
[26]:
```
upper_limit = data["PRCP"].quantile(0.99)
lower_limit = data["PRCP"].quantile(0.01)
```

[27]:
```
data["PRCP"] = np.where(data["PRCP"] < lower_limit, lower_limit, data["PRCP"])
data["PRCP"] = np.where(data["PRCP"] > upper_limit, upper_limit, data["PRCP"])
```

[28]:
```
sns.boxplot(x=data["PRCP"])
plt.show()
```



[29]:
```
sns.boxplot(x=data["TMAX"])
plt.show()
```



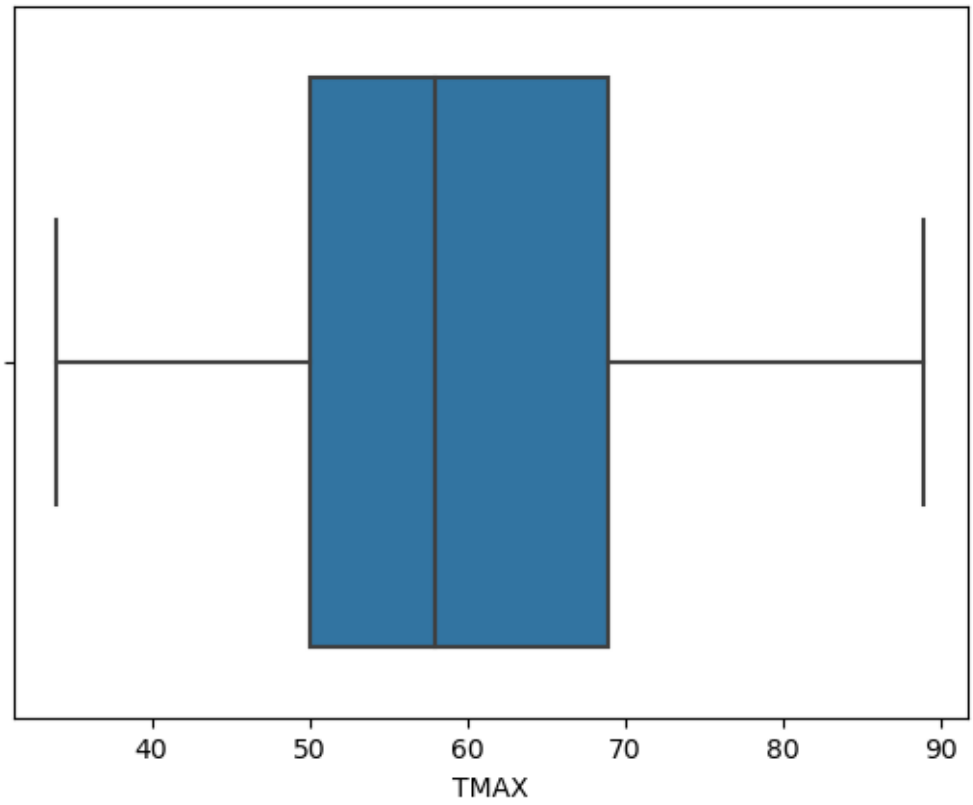
[37]:
```
data.shape
```

[37]: (25548, 5)

[30]:
```
upper_limit = data["TMAX"].quantile(0.99)
lower_limit = data["TMAX"].quantile(0.01)
```

[31]:
```
data["TMAX"] = np.where(data["TMAX"] < lower_limit, lower_limit, data["TMAX"])
data["TMAX"] = np.where(data["TMAX"] > upper_limit, upper_limit, data["TMAX"])
```

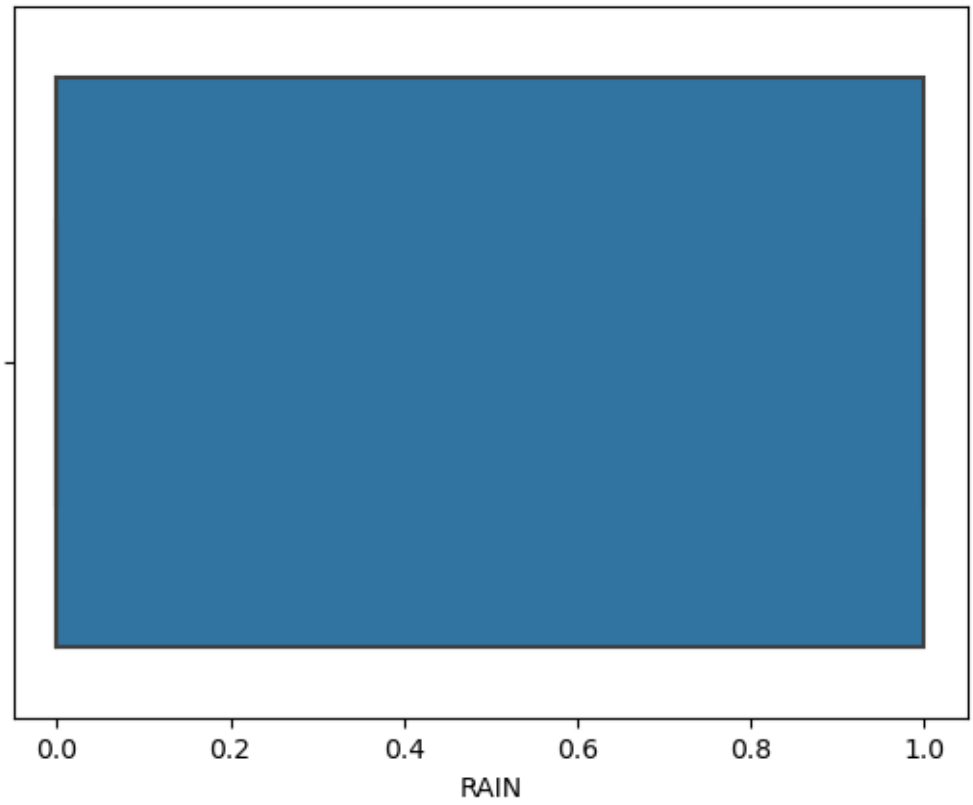
[32]:
```
sns.boxplot(x=data["TMAX"])
plt.show()
```



[36]:
```
data.shape
```

[36]: (25548, 5)

[33]:
```
sns.boxplot(x=data["RAIN"])
plt.show()
```



[35]:
```
data.shape
```

[35]:
```
(25548, 5)
```

[39]:
```
data = data.drop(data[(data["TMAX"] > 97)|(data["TMAX"] < 21)].index)
```

[41]:
```
data = data.drop(data[data["TMIN"] < 17].index)
```

[42]:
```
data = data.drop(data[data["PRCP"] > 0.2].index)
```

[44]:
```
data.shape
```

[44]:
```
(21306, 5)
```

[45]:
```
X = data.drop(["RAIN","DATE"], axis=1)
y = data["RAIN"]
```

[46]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.70,␣
 ↪random_state=42)
```

[47]:
```
#Creation of model
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

[48]:
```
#Model has learned from the training data
lr.fit(X_train,y_train)
```

[48]:
```
LogisticRegression()
```

[49]:
```
pred = lr.predict(X_test)
```

[50]:
```
X_test.head()
```

[50]:

	PRCP	TMAX	TMIN
13132	0.00	43.0	35.0
25234	0.00	43.0	29.0
11960	0.20	63.0	56.0
23422	0.07	45.0	38.0
16679	0.00	79.0	53.0

[51]:
```
pred
```

[51]:
```
array([0, 0, 1, …, 0, 1, 0])
```

[52]:
```
#Actual value
y_test
```

[52]:
```
13132    0
25234    0
11960    1
23422    1
16679    0
        ..
23717    1
6203     1
16128    0
25353    1
10251    0
Name: RAIN, Length: 6392, dtype: int32
```

[53]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,pred)
accuracy
```

[53]:
```
0.905819774718398
```

[54]:
```
lr.predict_proba(X_test)
```

[54]:
```
array([[6.72968884e-01, 3.27031116e-01],
       [8.56642894e-01, 1.43357106e-01],
       [4.30985713e-04, 9.99569014e-01],
       …,
       [9.78767931e-01, 2.12320687e-02],
       [1.28297250e-01, 8.71702750e-01],
       [7.13186420e-01, 2.86813580e-01]])
```

[55]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,pred)
cm
```

[55]:
```
array([[4367,   19],
       [ 583, 1423]], dtype=int64)
```

[56]:
```
#TP=4367, TN=1423, FP=19, FN=545
```

[58]:
```
#Prediction on new data
X_new=[[2.2,45,20]]
pred_new=lr.predict(X_new)
pred_new
```

[58]:
```
array([1])
```

[59]:
```
lr.predict_proba(X_new)
```

[59]:
```
array([[0., 1.]])
```

[ ]:
```

```