weather-predicting-system

July 26, 2023

1 WEATHER PREDICTING SYSTEM

2 IMPORTNG DATASET

[146]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
```

[147]:
```
dataset = pd.read_csv("seattle-weather.csv")
dataset.head(10)
```

[147]:

	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
5	2012-01-06	2.5	4.4	2.2	2.2	rain
6	2012-01-07	0.0	7.2	2.8	2.3	rain
7	2012-01-08	0.0	10.0	2.8	2.0	sun
8	2012-01-09	4.3	9.4	5.0	3.4	rain
9	2012-01-10	1.0	6.1	0.6	3.4	rain

3 DATA PREPROCESSING

Dimensional check

[148]:
```
dataset.shape
```

[148]:
```
(1461, 6)
```

[149]:
```
dataset.columns
```

[149]:
```
Index(['date', 'precipitation', 'temp_max', 'temp_min', 'wind', 'weather'],
dtype='object')
```

[150]:
```
dataset.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
```

4 Take care of missing data.

[151]:
```
dataset = dataset.dropna()
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 79.9+ KB
```

Here we have two temperature columns for the predection we had took avg of the both columns.

[152]:
```
avg_temp=dataset[["temp_max","temp_min"]].mean(axis=1)
dataset['avg_temp']=avg_temp
dataset.head()
```

[152]:

	date	precipitation	temp_max	temp_min	wind	weather	avg_temp
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle	8.90
1	2012-01-02	10.9	10.6	2.8	4.5	rain	6.70

```
2  2012-01-03           0.8      11.7       7.2   2.3     rain      9.45
3  2012-01-04          20.3      12.2       5.6   4.7     rain      8.90
4  2012-01-05           1.3       8.9       2.8   6.1     rain      5.85
```

[153]:
```
cols = ['temp_max', 'temp_min']
dataset = dataset.drop(cols, axis=1)
dataset.head()
```

[153]:
```
         date  precipitation  wind  weather  avg_temp
0  2012-01-01            0.0   4.7  drizzle      8.90
1  2012-01-02           10.9   4.5     rain      6.70
2  2012-01-03            0.8   2.3     rain      9.45
3  2012-01-04           20.3   4.7     rain      8.90
4  2012-01-05            1.3   6.1     rain      5.85
```

[154]:
```
dataset.columns
```

[154]:
```
Index(['date', 'precipitation', 'wind', 'weather', 'avg_temp'], dtype='object')
```

[155]:
```
dataset=dataset[["date" ,"precipitation", "wind","avg_temp","weather"]]
```

[156]:
```
dataset.head()
```

[156]:
```
         date  precipitation  wind  avg_temp  weather
0  2012-01-01            0.0   4.7      8.90  drizzle
1  2012-01-02           10.9   4.5      6.70     rain
2  2012-01-03            0.8   2.3      9.45     rain
3  2012-01-04           20.3   4.7      8.90     rain
4  2012-01-05            1.3   6.1      5.85     rain
```

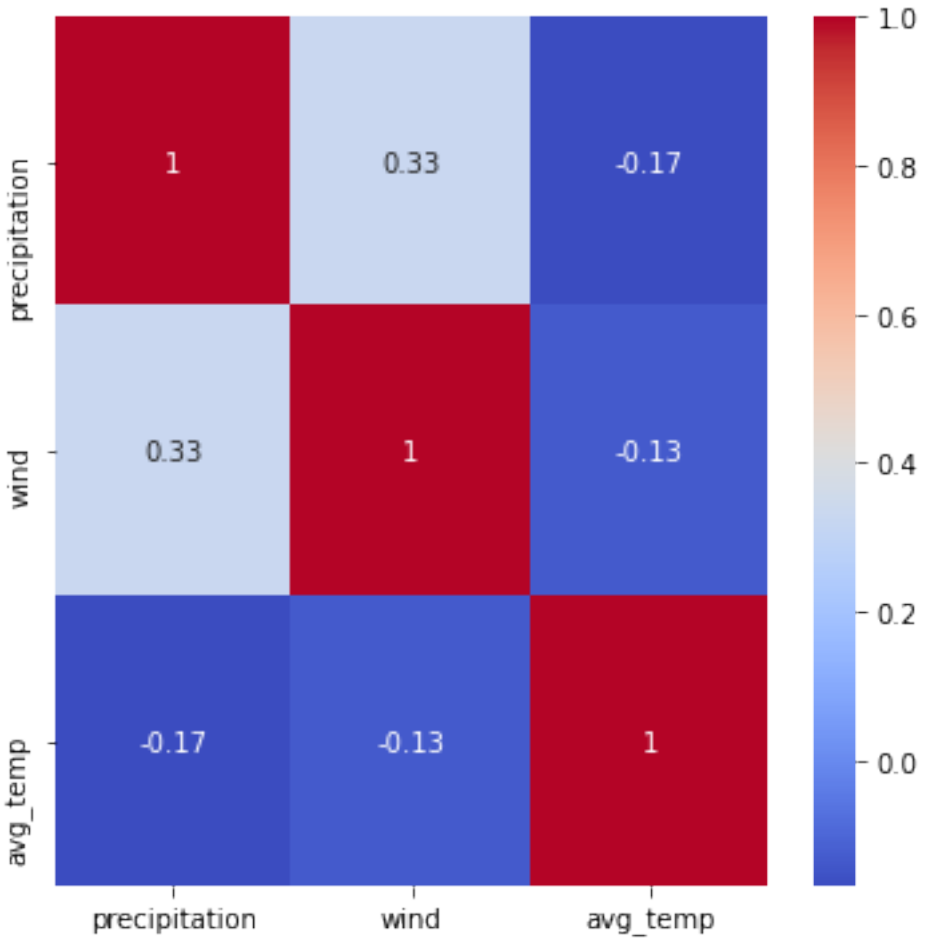
5 IDENTIFYING THE CORELATION USING HEAT MAP

[157]:
```
dataset.corr()
```

[157]:
```
               precipitation      wind  avg_temp
precipitation       1.000000  0.328045 -0.170465
wind                0.328045  1.000000 -0.132067
avg_temp           -0.170465 -0.132067  1.000000
```

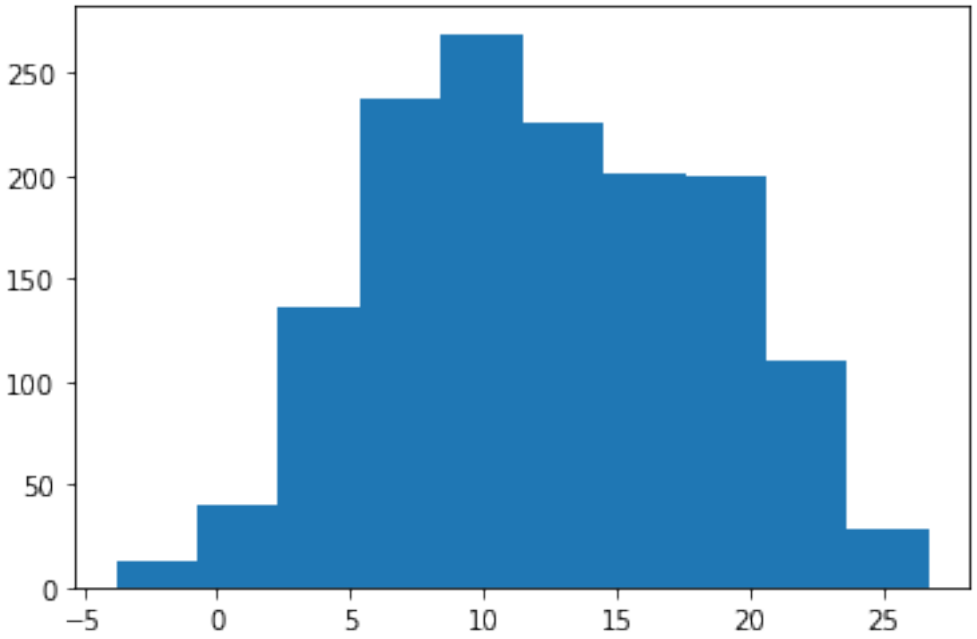
[158]:
```
plt.figure(figsize=(6,6))
sns.heatmap(dataset.corr(),annot=True,cmap="coolwarm")
```

[158]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf4bca55e0>
```



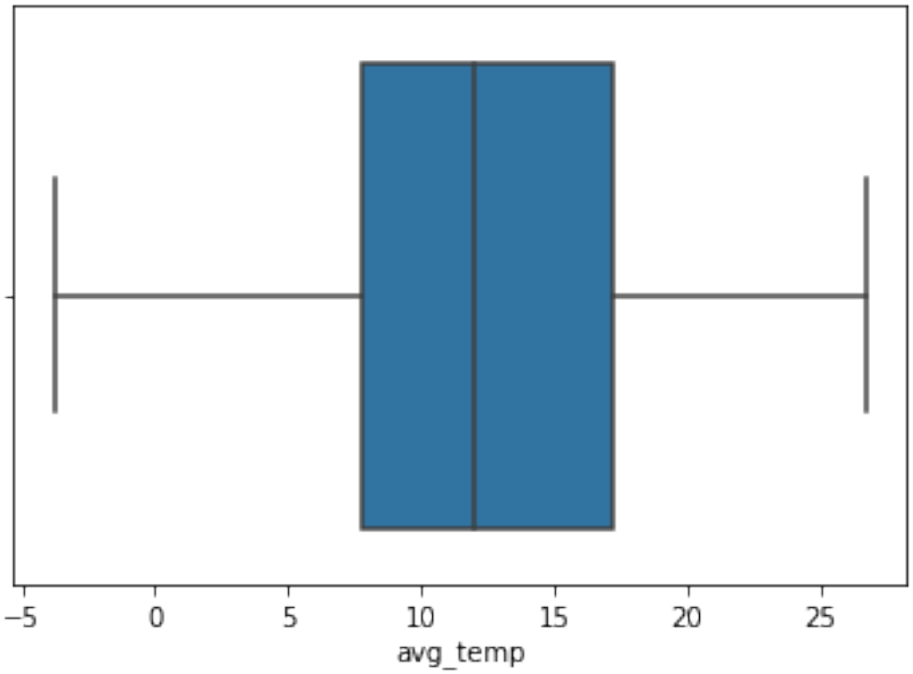
[159]:
```
plt.hist(dataset["avg_temp"]) #outliers
```

[159]:
```
(array([ 13.,  40., 136., 238., 269., 226., 201., 200., 110.,  28.]),
 array([-3.8 , -0.75,  2.3 ,  5.35,  8.4 , 11.45, 14.5 , 17.55, 20.6 ,
        23.65, 26.7 ]),
 <a list of 10 Patch objects>)
```



[160]:
```
sns.boxplot(x=dataset["avg_temp"])
```

[160]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf4ba16a90>
```



6 SPLITTING THE DATA

[161]:
```
X=dataset.iloc[:,1:4]
X.head()
```

[161]:

	precipitation	wind	avg_temp
0	0.0	4.7	8.90
1	10.9	4.5	6.70
2	0.8	2.3	9.45
3	20.3	4.7	8.90
4	1.3	6.1	5.85

[162]:
```
Y=dataset.iloc[:,-1:]
Y.head()
```

[162]:

	weather
0	drizzle
1	rain
2	rain
3	rain
4	rain

7 KNN ALGORITHM

[163]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20,␣
↪random_state = 0)
```

[164]:
```
#we are doing feature scaling to the training and test set of independent␣
↪variables for reducing the size to smaller values
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[165]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)␣
↪# n_neighbors is setting as 5, which means 5 neighborhood points are␣
↪required for classifying a given point.
classifier.fit(X_train, y_train)
#The Minkowski distance or Minkowski metric is a metric in a normed vector␣
↪space which can be considered as a generalization of both the Euclidean␣
↪distance and the Manhattan distance.
```

```
<ipython-input-165-abdf7f822748>:3: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(X_train, y_train)
```

[165]:
```
KNeighborsClassifier()
```

[166]:
```
y_pred = classifier.predict(X_test)
y_pred
```

[166]:
```
array(['sun', 'fog', 'rain', 'sun', 'drizzle', 'rain', 'rain', 'rain',
       'sun', 'sun', 'sun', 'rain', 'sun', 'sun', 'sun', 'sun', 'sun',
       'rain', 'rain', 'sun', 'rain', 'rain', 'rain', 'sun', 'sun', 'sun',
       'sun', 'rain', 'sun', 'sun', 'rain', 'rain', 'rain', 'rain', 'sun',
       'rain', 'sun', 'rain', 'sun', 'rain', 'sun', 'rain', 'rain',
       'rain', 'sun', 'sun', 'drizzle', 'sun', 'sun', 'sun', 'sun', 'sun',
       'rain', 'rain', 'rain', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun',
       'sun', 'sun', 'sun', 'sun', 'rain', 'sun', 'sun', 'sun', 'rain',
       'rain', 'sun', 'rain', 'fog', 'fog', 'sun', 'sun', 'sun', 'rain',
       'rain', 'rain', 'sun', 'rain', 'rain', 'rain', 'sun', 'fog',
       'rain', 'sun', 'rain', 'sun', 'sun', 'rain', 'sun', 'rain', 'rain',
       'sun', 'sun', 'sun', 'rain', 'rain', 'sun', 'sun', 'rain', 'rain',
       'sun', 'rain', 'rain', 'fog', 'sun', 'rain', 'sun', 'rain', 'rain',
       'sun', 'rain', 'sun', 'sun', 'sun', 'rain', 'rain', 'rain', 'rain',
       'rain', 'rain', 'rain', 'sun', 'drizzle', 'sun', 'sun', 'rain',
       'rain', 'sun', 'sun', 'rain', 'fog', 'sun', 'sun', 'sun', 'rain',
       'rain', 'rain', 'sun', 'sun', 'sun', 'sun', 'rain', 'sun', 'rain',
       'sun', 'rain', 'sun', 'sun', 'fog', 'fog', 'sun', 'sun', 'sun',
       'sun', 'sun', 'sun', 'fog', 'sun', 'sun', 'rain', 'rain', 'rain',
       'rain', 'rain', 'rain', 'sun', 'rain', 'sun', 'sun', 'rain',
       'rain', 'sun', 'fog', 'sun', 'sun', 'sun', 'sun', 'sun', 'sun',
       'rain', 'sun', 'sun', 'rain', 'fog', 'sun', 'sun', 'sun', 'sun',
       'drizzle', 'rain', 'rain', 'sun', 'sun', 'sun', 'rain', 'rain',
       'sun', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'sun', 'sun',
       'rain', 'rain', 'rain', 'rain', 'sun', 'sun', 'sun', 'rain',
       'rain', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun', 'sun', 'sun',
       'rain', 'rain', 'sun', 'drizzle', 'rain', 'rain', 'rain', 'rain',
       'fog', 'sun', 'sun', 'rain', 'rain', 'sun', 'sun', 'sun', 'sun',
       'rain', 'sun', 'sun', 'sun', 'sun', 'sun', 'sun', 'sun', 'rain',
       'rain', 'sun', 'fog', 'sun', 'sun', 'sun', 'rain', 'sun', 'fog',
       'fog', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'sun', 'rain',
       'sun', 'rain', 'rain', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun',
       'rain', 'rain', 'rain', 'sun', 'sun', 'sun', 'sun', 'sun', 'rain',
       'sun', 'rain', 'rain', 'sun'], dtype=object)
```

[167]:
```
y_test
```

[167]:

	weather
530	sun
657	sun
459	rain
279	sun

```
656     sun
..      …
440    rain
634     sun
494     sun
61     rain
517     sun

[293 rows x 1 columns]
```

[168]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac1 = accuracy_score(y_test,y_pred)
```

[169]:
```
ac1
```

[169]:
```
0.689419795221843
```

[170]:
```
cm
```

[170]:
```
array([[  2,   0,   2,   0,   6],
       [  1,   3,   3,   0,  22],
       [  0,   3,  95,   0,  25],
       [  0,   0,   5,   0,   1],
       [  2,   9,  12,   0, 102]], dtype=int64)
```

8 NAVIE BAYES

[171]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\utils\validation.py:73:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
```

[171]:
```
GaussianNB()
```

[172]:
```
y_pred  =  classifier.predict(X_test)
y_pred
```

[172]:
```
array(['sun', 'sun', 'rain', 'sun', 'sun', 'rain', 'rain', 'rain', 'sun',
       'rain', 'sun', 'rain', 'sun', 'sun', 'sun', 'sun', 'sun', 'rain',
       'rain', 'rain', 'rain', 'rain', 'rain', 'sun', 'sun', 'sun', 'sun',
```

```
       'rain', 'sun', 'sun', 'rain', 'sun', 'rain', 'rain', 'sun', 'rain',
       'sun', 'rain', 'sun', 'rain', 'sun', 'rain', 'rain', 'rain', 'sun',
       'sun', 'sun', 'sun', 'sun', 'sun', 'sun', 'sun', 'sun', 'rain',
       'rain', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'sun', 'sun',
       'sun', 'sun', 'rain', 'sun', 'sun', 'sun', 'rain', 'rain', 'rain',
       'rain', 'sun', 'sun', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun',
       'sun', 'rain', 'rain', 'rain', 'sun', 'sun', 'rain', 'sun', 'rain',
       'sun', 'sun', 'rain', 'sun', 'rain', 'rain', 'sun', 'sun', 'sun',
       'sun', 'rain', 'sun', 'sun', 'rain', 'rain', 'sun', 'rain', 'rain',
       'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'sun', 'sun', 'sun',
       'sun', 'rain', 'rain', 'sun', 'rain', 'rain', 'sun', 'rain', 'sun',
       'sun', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'rain', 'rain',
       'rain', 'sun', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun', 'sun',
       'sun', 'rain', 'rain', 'sun', 'sun', 'sun', 'sun', 'sun', 'sun',
       'rain', 'sun', 'rain', 'sun', 'sun', 'rain', 'sun', 'sun', 'sun',
       'rain', 'sun', 'rain', 'rain', 'sun', 'rain', 'rain', 'rain',
       'sun', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun', 'sun', 'sun',
       'sun', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'sun', 'sun',
       'rain', 'sun', 'sun', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun',
       'sun', 'sun', 'sun', 'rain', 'sun', 'sun', 'sun', 'rain', 'rain',
       'rain', 'sun', 'sun', 'sun', 'rain', 'rain', 'rain', 'rain', 'sun',
       'sun', 'sun', 'rain', 'rain', 'rain', 'sun', 'sun', 'sun', 'rain',
       'sun', 'sun', 'sun', 'sun', 'rain', 'sun', 'sun', 'rain', 'rain',
       'rain', 'rain', 'sun', 'rain', 'sun', 'rain', 'rain', 'sun',
       'rain', 'sun', 'rain', 'rain', 'sun', 'rain', 'sun', 'sun', 'sun',
       'sun', 'sun', 'rain', 'rain', 'sun', 'sun', 'sun', 'sun', 'sun',
       'sun', 'rain', 'sun', 'sun', 'sun', 'sun', 'rain', 'sun', 'rain',
       'sun', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun', 'sun', 'rain',
       'sun', 'rain', 'sun', 'rain', 'rain', 'rain', 'sun', 'sun', 'sun',
       'rain', 'sun', 'rain', 'sun', 'sun', 'rain', 'sun'], dtype='<U7')
```

[173]:
```
y_test
```

[173]:
```
    weather
530     sun
657     sun
459    rain
279     sun
656     sun
..      …
440    rain
634     sun
494     sun
61     rain
517     sun

[293 rows x 1 columns]
```

[174]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac2 = accuracy_score(y_test,y_pred)
```

[175]:
```
ac2
```

[175]:
```
0.8054607508532423
```

[176]:
```
cm
```

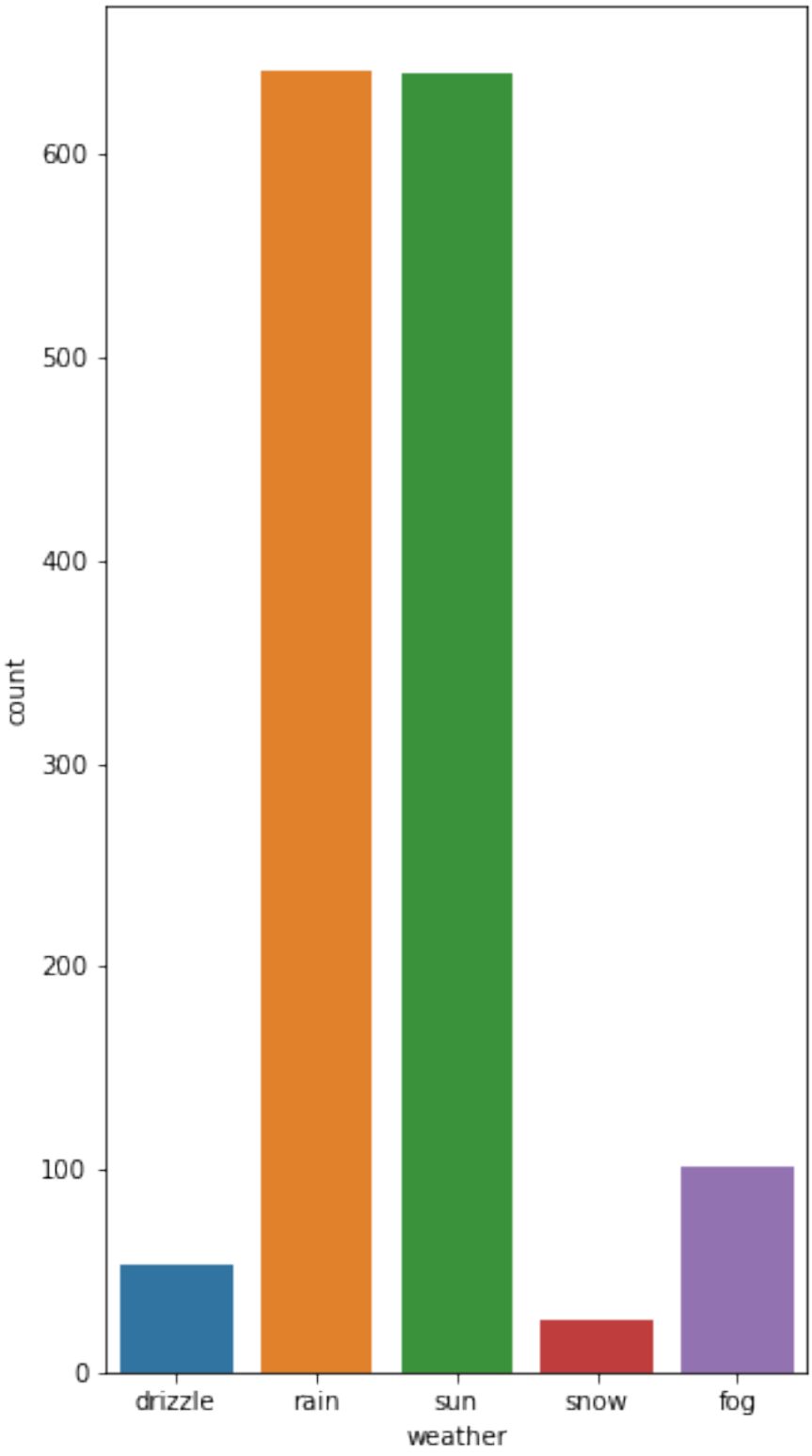
[176]:
```
array([[  0,   0,   0,   0,  10],
       [  0,   0,   0,   0,  29],
       [  0,   0, 111,   0,  12],
       [  0,   0,   6,   0,   0],
       [  0,   0,   0,   0, 125]], dtype=int64)
```

9 DECISION TREE

[177]:
```
dataset['weather'].value_counts()
```

[177]:
```
rain       641
sun        640
fog        101
drizzle     53
snow        26
Name: weather, dtype: int64
```

[178]:
```
plt.figure(figsize=(5,10));
sns.countplot(dataset['weather']);
```


600
500
400
300
200
100
0
count
drizzle
rain
sun
snow
fog
weather

[179]:
```
from sklearn.tree import DecisionTreeClassifier
```

[180]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
```

[181]:
```
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
```

[182]:
```
#Predict the response for test dataset
y_pred = clf.predict(X_test)
y_pred
```

[182]:
```
array(['sun', 'fog', 'rain', 'sun', 'sun', 'rain', 'rain', 'rain',
       'drizzle', 'rain', 'sun', 'rain', 'sun', 'sun', 'sun', 'sun',
       'sun', 'rain', 'rain', 'rain', 'rain', 'rain', 'rain', 'sun',
       'sun', 'sun', 'sun', 'rain', 'sun', 'sun', 'rain', 'sun', 'rain',
       'rain', 'drizzle', 'rain', 'sun', 'rain', 'sun', 'rain', 'sun',
       'rain', 'rain', 'rain', 'drizzle', 'fog', 'drizzle', 'sun', 'sun',
       'sun', 'sun', 'sun', 'rain', 'rain', 'rain', 'sun', 'sun', 'rain',
       'sun', 'rain', 'sun', 'sun', 'sun', 'sun', 'fog', 'rain', 'sun',
       'sun', 'sun', 'rain', 'snow', 'snow', 'rain', 'sun', 'fog', 'sun',
       'sun', 'drizzle', 'rain', 'rain', 'sun', 'sun', 'rain', 'rain',
       'rain', 'sun', 'fog', 'rain', 'sun', 'rain', 'sun', 'sun', 'rain',
       'sun', 'rain', 'rain', 'sun', 'drizzle', 'sun', 'sun', 'rain',
       'sun', 'sun', 'rain', 'rain', 'drizzle', 'rain', 'rain', 'fog',
       'fog', 'rain', 'sun', 'rain', 'fog', 'sun', 'sun', 'sun', 'sun',
       'rain', 'rain', 'rain', 'rain', 'rain', 'sun', 'rain', 'rain',
       'sun', 'sun', 'sun', 'rain', 'sun', 'rain', 'sun', 'rain', 'rain',
       'rain', 'drizzle', 'fog', 'sun', 'fog', 'rain', 'rain', 'sun',
       'sun', 'sun', 'rain', 'rain', 'drizzle', 'fog', 'sun', 'sun',
       'sun', 'sun', 'rain', 'drizzle', 'rain', 'sun', 'sun', 'rain',
       'sun', 'sun', 'sun', 'rain', 'sun', 'rain', 'rain', 'fog', 'rain',
       'rain', 'rain', 'sun', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun',
       'fog', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun', 'rain', 'sun',
       'sun', 'sun', 'rain', 'sun', 'sun', 'sun', 'sun', 'drizzle',
       'rain', 'rain', 'sun', 'drizzle', 'rain', 'rain', 'rain', 'sun',
       'sun', 'drizzle', 'rain', 'rain', 'rain', 'sun', 'sun', 'sun',
       'rain', 'rain', 'rain', 'rain', 'fog', 'sun', 'sun', 'rain',
       'rain', 'rain', 'sun', 'sun', 'sun', 'rain', 'drizzle', 'sun',
       'sun', 'sun', 'rain', 'drizzle', 'rain', 'rain', 'rain', 'rain',
       'rain', 'drizzle', 'rain', 'sun', 'rain', 'rain', 'rain', 'rain',
       'sun', 'rain', 'rain', 'sun', 'rain', 'sun', 'drizzle', 'sun',
       'sun', 'sun', 'rain', 'snow', 'drizzle', 'fog', 'sun', 'sun',
       'sun', 'rain', 'rain', 'fog', 'sun', 'sun', 'sun', 'rain', 'fog',
       'rain', 'drizzle', 'sun', 'sun', 'sun', 'rain', 'rain', 'sun',
       'sun', 'rain', 'drizzle', 'rain', 'sun', 'rain', 'rain', 'rain',
       'sun', 'fog', 'drizzle', 'rain', 'sun', 'rain', 'sun', 'fog',
```

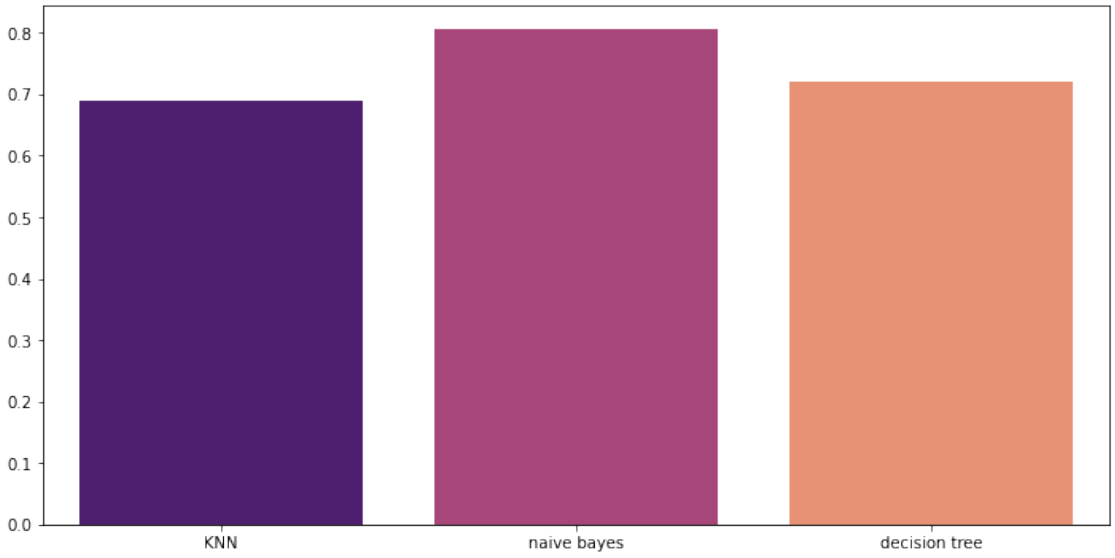
```
       'rain', 'sun'], dtype=object)
```

[183]:
```
from sklearn import metrics
ac3= metrics.accuracy_score(y_test, y_pred)
ac3
```

[183]:
```
0.7201365187713311
```

[184]:
```
plt.figure(figsize=(12,6))
model_acc = [ac1, ac2, ac3]
model_name = [ 'KNN', 'naive bayes', 'decision tree']
sns.barplot(x= model_name, y=model_acc, palette='magma')
```

[184]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf4bd16340>
```



[ ]: