decision-tree1

November 10, 2024

[ ]:
```
#Step-1: Begin the tree with the root node, says S, which contains the complete␣
↪dataset.
#Step-2: Find the best attribute in the dataset using Attribute Selection␣
↪Measure (ASM).
#Step-3: Divide the S into subsets that contains possible values for the best␣
↪attributes.
#Step-4: Generate the decision tree node, which contains the best attribute.
#Step-5: Recursively make new decision trees using the subsets of the dataset␣
↪created in step -3.
#Continue this process until a stage is reached where you cannot further␣
↪classify the nodes and
#called the final node as a leaf node.
```

[7]:
```
import pandas as pd
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import os
```

[8]:
```
df=pd.read_csv("E:\DATASET/daily_weather.csv")
df.head(10)
```

[8]:
```
   number  air_pressure_9am  air_temp_9am  avg_wind_direction_9am  \
0       0        918.060000     74.822000              271.100000
1       1        917.347688     71.403843              101.935179
2       2        923.040000     60.638000               51.000000
3       3        920.502751     70.138895              198.832133
4       4        921.160000     44.294000              277.800000
5       5        915.300000     78.404000              182.800000
6       6        915.598868     70.043304              177.875407
7       7        918.070000     51.710000              242.400000
8       8        920.080000     80.582000               40.700000
9       9        915.010000     47.498000              163.100000

   avg_wind_speed_9am  max_wind_direction_9am  max_wind_speed_9am  \
0            2.080354              295.400000            2.863283
1            2.443009              140.471548            3.533324
```

```
2         17.067852           63.700000          22.100967
3          4.337363          211.203341           5.190045
4          1.856660          136.500000           2.863283
5          9.932014          189.000000          10.983375
6          3.745587          186.606696           4.589632
7          2.527742          271.600000           3.646212
8          4.518619           63.000000           5.883152
9          4.943637          195.900000           6.576604

   rain_accumulation_9am  rain_duration_9am  relative_humidity_9am  \
0                   0.00                0.0              42.420000
1                   0.00                0.0              24.328697
2                   0.00               20.0               8.900000
3                   0.00                0.0              12.189102
4                   8.90            14730.0              92.410000
5                   0.02              170.0              35.130000
6                   0.00                0.0              10.657422
7                   0.00                0.0              80.470000
8                   0.00                0.0              29.580000
9                   0.00                0.0              88.600000

   relative_humidity_3pm
0              36.160000
1              19.426597
2              14.460000
3              12.742547
4              76.740000
5              33.930000
6              21.385657
7              74.920000
8              24.030000
9              68.050000
```

[9]:
```
df.columns
```

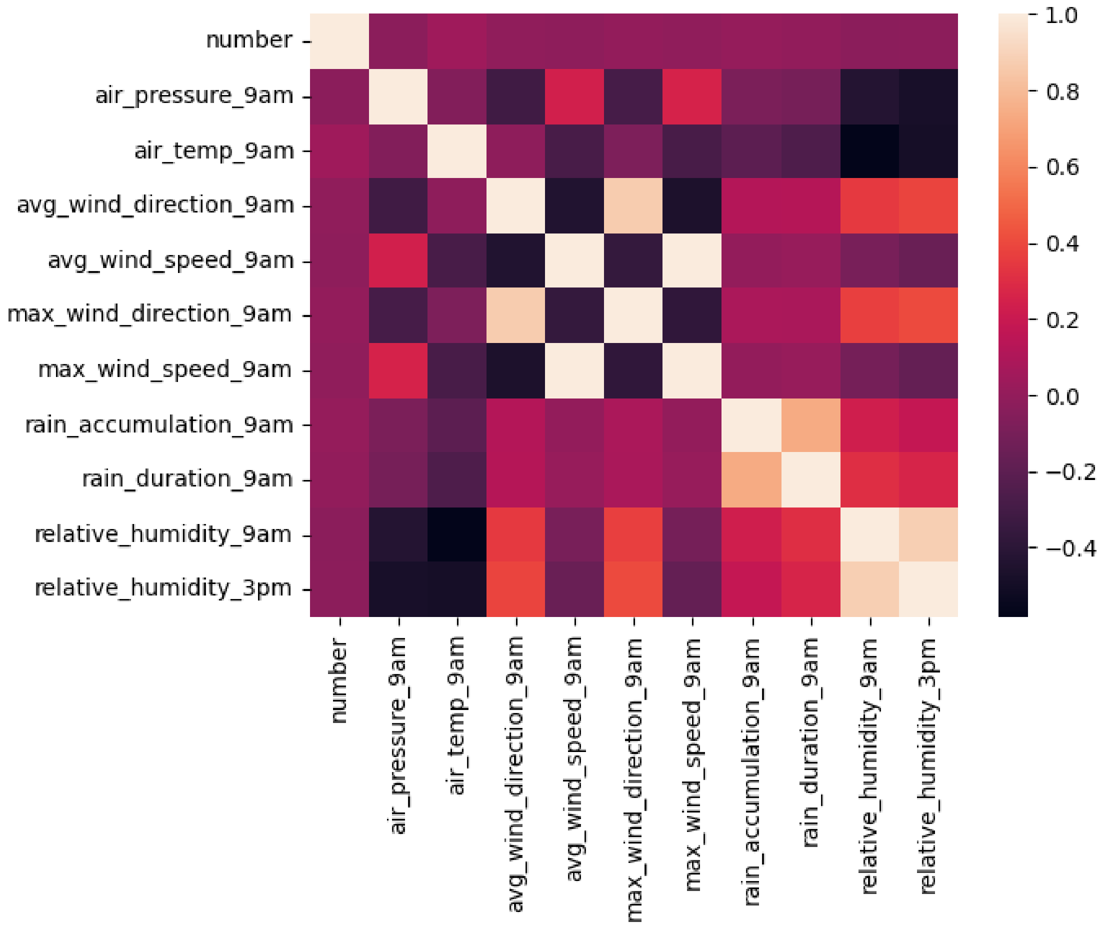
[9]:
```
Index(['number', 'air_pressure_9am', 'air_temp_9am', 'avg_wind_direction_9am',
       'avg_wind_speed_9am', 'max_wind_direction_9am', 'max_wind_speed_9am',
       'rain_accumulation_9am', 'rain_duration_9am', 'relative_humidity_9am',
       'relative_humidity_3pm'],
      dtype='object')
```

[10]:
```
df.shape
```

[10]:
```
(1095, 11)
```

[31]:
```
import seaborn as sns
sns.heatmap(df.corr())
```

[31]: <AxesSubplot:>



[11]:
```
df[df.isnull().any(axis=1)].head()
```

[11]:
```
     number  air_pressure_9am  air_temp_9am  avg_wind_direction_9am  \
16       16        917.890000           NaN              169.200000
111     111        915.290000     58.820000              182.600000
177     177        915.900000           NaN              183.300000
262     262        923.596607     58.380598               47.737753
277     277        920.480000     62.600000              194.400000

     avg_wind_speed_9am  max_wind_direction_9am  max_wind_speed_9am  \
16             2.192201              196.800000            2.930391
111           15.613841              189.000000                 NaN
177            4.719943              189.900000            5.346287
262           10.636273               67.145843           13.671423
277            2.751436                     NaN            3.869906
```

```
     rain_accumulation_9am  rain_duration_9am  relative_humidity_9am  \
16                     0.0                0.0              48.990000
111                    0.0                0.0              21.500000
177                    0.0                0.0              29.260000
262                    0.0                NaN              17.990876
277                    0.0                0.0              52.580000

     relative_humidity_3pm
16               51.190000
111              29.690000
177              46.500000
262              16.461685
277              54.030000
```

[12]:
```
df=df.dropna()
df.shape
```

[12]:
```
(1064, 11)
```

[13]:
```
clean_df=df.copy()
clean_df['high humadity label']=(clean_df['relative_humidity_3pm']>28)*1
clean_df['high humadity label'].head()
```

[13]:
```
0    1
1    0
2    0
3    0
4    1
Name: high humadity label, dtype: int32
```

[14]:
```
#target variable
y=clean_df[['high humadity label']].copy()
y.head()
```

[14]:
```
   high humadity label
0                    1
1                    0
2                    0
3                    0
4                    1
```

[15]:
```
df.columns
features=['air_pressure_9am', 'air_temp_9am', 'avg_wind_direction_9am',
       'avg_wind_speed_9am', 'max_wind_direction_9am', 'max_wind_speed_9am',
       'rain_accumulation_9am', 'rain_duration_9am', 'relative_humidity_9am',
       ]
```

[16]:
```
x=clean_df[features].copy()
x.columns
```

[16]:
```
Index(['air_pressure_9am', 'air_temp_9am', 'avg_wind_direction_9am',
       'avg_wind_speed_9am', 'max_wind_direction_9am', 'max_wind_speed_9am',
       'rain_accumulation_9am', 'rain_duration_9am', 'relative_humidity_9am'],
      dtype='object')
```

[17]:
```
y.columns
```

[17]:
```
Index(['high humadity label'], dtype='object')
```

[18]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪33,random_state=324)
```

[19]:
```
humidity_classifier=DecisionTreeClassifier(max_leaf_nodes=10,random_state=0)
humidity_classifier.fit(x_train,y_train)
```

[19]:
```
DecisionTreeClassifier(max_leaf_nodes=10, random_state=0)
```

[20]:
```
y_predicted=humidity_classifier.predict(x_test)
y_predicted
```

[20]:
```
array([0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0])
```

[21]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_predicted) * 100
print(f"Accuracy: {accuracy:.2f}%")
```

```
Accuracy: 88.92%
```

[22]:
```
confusion_matrix(y_test,y_predicted)
```

[22]: array([[166, 14],
       [25, 147]], dtype=int64)

[24]:
```
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

[27]:
```
import matplotlib.pyplot as plt
cm=confusion_matrix(y_test,y_predicted)
plt.figure(figsize=(8,6))
```

[27]: <Figure size 800x600 with 0 Axes>

<Figure size 800x600 with 0 Axes>

[30]:
```
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues',xticklabels=['Predicted␣
 ↪0','Predicted 1'],yticklabels=['Actual 0','Actual 1'])
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion matrix')
plt.show()
```