Classification of Weather Data using scikit-learn

Daily Weather Data Analysis

In this notebook, we will use scikit-learn to perform a decision tree based classification of weather data.

Importing the Necessary Libraries

In [1]:

```
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

Creating a Pandas DataFrame from a CSV file

In [2]:

```
data = pd.read_csv('./weather/daily_weather.csv')
```

Daily Weather Data Description

The file **daily_weather.csv** is a comma-separated file that contains weather data. This data comes from a weather station located in San Diego, California. The weather station is equipped with sensors that capture weather-related measurements such as air temperature, air pressure, and relative humidity. Data was collected for a period of three years, from September 2011 to September 2014, to ensure that sufficient data for different seasons and weather conditions is captured.

Let's now check all the columns in the data.

In [3]:

```
data.columns
```

Out[3]:

```
Index(['number', 'air_pressure_9am', 'air_temp_9am', 'avg_wind_directi
on_9am',
       'avg_wind_speed_9am', 'max_wind_direction_9am', 'max_wind_speed
_9am',
       'rain_accumulation_9am', 'rain_duration_9am', 'relative_humidit
y_9am',
       'relative_humidity_3pm'],
      dtype='object')
```

Each row in daily_weather.csv captures weather data for a separate day.

Sensor measurements from the weather station were captured at one-minute intervals. These measurements were then processed to generate values to describe daily weather. Since this dataset was created to classify low-humidity days vs. non-low-humidity days (that is, days with normal or high humidity), the variables included are weather measurements in the morning, with one measurement, namely relatively humidity, in the afternoon. The idea is to use the morning weather values to predict whether the day will be low-humidity or not based on the afternoon measurement of relative humidity.

Each row, or sample, consists of the following variables:

- **number:** unique number for each row
- **air_pressure_9am:** air pressure averaged over a period from 8:55am to 9:04am (*Unit: hectopascals*)
- **air_temp_9am:** air temperature averaged over a period from 8:55am to 9:04am (*Unit: degrees Fahrenheit*)
- **air_wind_direction_9am:** wind direction averaged over a period from 8:55am to 9:04am (*Unit: degrees, with 0 means coming from the North, and increasing clockwise*)
- **air_wind_speed_9am:** wind speed averaged over a period from 8:55am to 9:04am (*Unit: miles per hour*)
- **max_wind_direction_9am:** wind gust direction averaged over a period from 8:55am to 9:10am (*Unit: degrees, with 0 being North and increasing clockwise*)
- **max_wind_speed_9am:** wind gust speed averaged over a period from 8:55am to 9:04am (*Unit: miles per hour*)
- **rain_accumulation_9am:** amount of rain accumulated in the 24 hours prior to 9am (*Unit: millimeters*)
- **rain_duration_9am:** amount of time rain was recorded in the 24 hours prior to 9am (*Unit: seconds*)
- **relative_humidity_9am:** relative humidity averaged over a period from 8:55am to 9:04am (*Unit: percent*)
- **relative_humidity_3pm:** relative humidity averaged over a period from 2:55pm to 3:04pm (*Unit: percent*)

In [4]:

```
data
```

Out[4]:

	number	air_pressure_9am	air_temp_9am	avg_wind_direction_9am	avg_wind_speed_9am	max_wind_direc
0	0	918.060000	74.822000	271.100000	2.080354	2
1	1	917.347688	71.403843	101.935179	2.443009	1
2	2	923.040000	60.638000	51.000000	17.067852	
3	3	920.502751	70.138895	198.832133	4.337363	2
4	4	921.160000	44.294000	277.800000	1.856660	1
5	5	915.300000	78.404000	182.800000	9.932014	1
6	6	915.598868	70.043304	177.875407	3.745587	1
7	7	918.070000	51.710000	242.400000	2.527742	2
8	8	920.080000	80.582000	40.700000	4.518619	

In [5]:

```
data[data.isnull().any(axis = 1)]
```

Out[5]:

	number	air_pressure_9am	air_temp_9am	avg_wind_direction_9am	avg_wind_speed_9am	max_wind_direc
16	16	917.890000	NaN	169.200000	2.192201	1
111	111	915.290000	58.820000	182.600000	15.613841	1
177	177	915.900000	NaN	183.300000	4.719943	1
262	262	923.596607	58.380598	47.737753	10.636273	
277	277	920.480000	62.600000	194.400000	2.751436	
334	334	916.230000	75.740000	149.100000	2.751436	1
358	358	917.440000	58.514000	55.100000	10.021491	
361	361	920.444946	65.801845	49.823346	21.520177	
381	381	918.480000	66.542000	90.900000	3.467257	

Data Cleaning Steps

We will not need to number for each row so we can clean it.

In [6]:

```
del data['number']
```

Now let's drop null values using the *pandas dropna* function.

In [7]:

```
before_rows = data.shape[0] # get the number of rows in a data set
print(before_rows)
```

```
1095
```

In [8]:

```
data = data.dropna()
```

In [9]:

```
after_rows = data.shape[0]
print(after_rows)
```

```
1064
```

How many rows dropped due to cleaning?

In [10]:

```
before_rows - after_rows
```

Out[10]:

```
31
```

Convert to a Classification Task

Binarize the relative_humidity_3pm to 0 or 1.

In [11]:

```
clean_data = data.copy()
clean_data['high_humidity_label'] = (clean_data['relative_humidity_3pm'] > 24.99)*1
print(clean_data['high_humidity_label'])
```

```
0       1
1       0
2       0
3       0
4       1
5       1
6       0
7       1
8       0
9       1
10      1
11      1
12      1
13      1
14      0
15      0
17      0
18      1
19      0
20      0
21      1
22      0
23      1
24      0
25      1
26      1
27      1
28      1
29      1
30      1
       ..
1064    1
1065    1
1067    1
1068    1
1069    1
1070    1
1071    1
1072    0
1073    1
1074    1
1075    0
1076    0
1077    1
1078    0
1079    1
1080    0
1081    0
1082    1
1083    1
1084    1
1085    1
1086    1
1087    1
1088    1
```

```
1089    1
1090    1
1091    1
1092    1
1093    1
1094    0
Name: high_humidity_label, Length: 1064, dtype: int64
```

Target is stored in 'y'.

In [12]:

```
# Store high humidity label column into y
y = clean_data[["high_humidity_label"]].copy()
y
```

Out[12]:

	high_humidity_label
0	1
1	0
2	0
3	0
4	1
5	1
6	0
7	1
8	0
9	1
10	1
11	1
12	1
13	1
14	0
15	0
17	0
18	1
19	0
20	0
21	1
22	0
23	1
24	0
25	1
26	1
27	1
28	1
29	1
30	1
...	...
1064	1

	high_humidity_label
1065	1
1067	1
1068	1
1069	1
1070	1
1071	1
1072	0
1073	1
1074	1
1075	0
1076	0
1077	1
1078	0
1079	1
1080	0
1081	0
1082	1
1083	1
1084	1
1085	1
1086	1
1087	1
1088	1
1089	1
1090	1
1091	1
1092	1
1093	1
1094	0

1064 rows × 1 columns

In [13]:

```
clean_data['relative_humidity_3pm'].head()
```

Out[13]:

```
0    36.160000
1    19.426597
2    14.460000
3    12.742547
4    76.740000
Name: relative_humidity_3pm, dtype: float64
```

In [14]:

```
y.head() # over 24.99 data are turned into 1
```

Out[14]:

	high_humidity_label
0	1
1	0
2	0
3	0
4	1

Use 9am Sensor Signals as Features to Predict Humidity at 3pm

In [15]:

```
morning_features = ['air_pressure_9am','air_temp_9am','avg_wind_direction_9am','avg_
        'max_wind_direction_9am','max_wind_speed_9am','rain_accumulation_9am',
        'rain_duration_9am']
```

In [16]:

```
X = clean_data[morning_features].copy()
```

In [17]:

```
X.columns
```

Out[17]:

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

In [18]:

```
y.columns
```

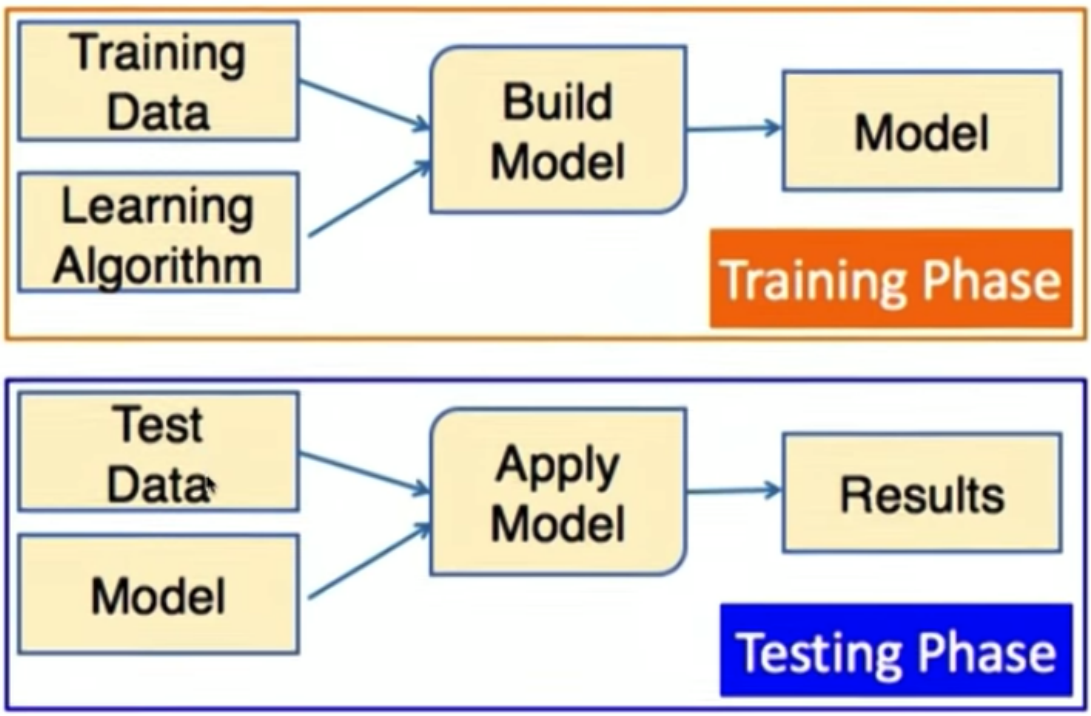
Out[18]:

```
Index(['high_humidity_label'], dtype='object')
```

Perform Test and Train split

REMINDER: Training Phase

In the **training phase**, the learning algorithm uses the training data to adjust the model's parameters to minimize errors. At the end of the training phase, you get the trained model.



In the **testing phase**, the trained model is applied to test data. Test data is separate from the training data, and is previously unseen by the model. The model is then evaluated on how it performs on the test data. The goal in building a classifier model is to have the model perform well on training as well as test data.

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_sta
```

In [20]:

```
type(X_train)
```

Out[20]:

```
pandas.core.frame.DataFrame
```

In [21]:

```
type(X_test)
```

Out[21]:

```
pandas.core.frame.DataFrame
```

In [22]:

```
type(y_train)
```

Out[22]:

```
pandas.core.frame.DataFrame
```

In [23]:

```
type(y_test)
```

Out[23]:

```
pandas.core.frame.DataFrame
```

In [24]:

```
X_train.head()
```

Out[24]:

	air_pressure_9am	air_temp_9am	avg_wind_direction_9am	avg_wind_speed_9am	max_wind_
841	918.370000	72.932000	184.500000	2.013246	
75	920.100000	53.492000	186.100000	13.444009	
95	927.610000	54.896000	55.000000	4.988376	
895	919.235153	65.951112	194.343333	2.942019	
699	919.888128	68.687822	228.517730	3.960858	

In [25]:

```
y_train.head()
```

Out[25]:

	high_humidity_label
841	0
75	1
95	0
895	0
699	0

Fit on Train Set

In [26]:

```
humidity_clf = DecisionTreeClassifier(max_leaf_nodes = 10, random_state = 0)
humidity_clf.fit(X_train, y_train)
```

Out[26]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
None,
            max_features=None, max_leaf_nodes=10,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=
0,
            splitter='best')
```

In [27]:

```
type(humidity_clf)
```

Out[27]:

```
sklearn.tree.tree.DecisionTreeClassifier
```

Predict on Test Set

In [28]:

```
pred = humidity_clf.predict(X_test)
```

In [29]:

```
pred[:10]
```

Out[29]:

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

In [30]:

```
y_test['high_humidity_label'][:10]
```

Out[30]:

```
456     0
845     0
693     1
259     1
723     1
224     1
300     1
442     0
585     1
1057    1
Name: high_humidity_label, dtype: int64
```

Measure Accuracy of the Classifier

In [31]:

```
accuracy_score(y_true = y_test, y_pred = pred)
```

Out[31]:

```
0.8153409090909091
```

In [ ]:

```

```