Chapter 6 - Exercise 4: Weather

Cho dữ liệu weather.csv

Sử dụng thuật toán Decision Tree để dự đoán nhiệt độ (Temperature_c) dựa trên các thông tin được cung cấp.

1. Đọc dữ liệu và gán cho biến data. Xem thông tin data: shape, type, head(), tail(), info. Tiền xử lý dữ liệu (nếu cần)
2. Từ inputs data và outputs data => Tạo X_train, X_test, y_train, y_test với tỷ lệ 80:20
3. Thực hiện Decision Tree với X_train, y_train
4. Dự đoán y từ X_test => so sánh với y_test
5. Xem kết quả => Nhận xét model
6. Ghi model nếu model phù hợp

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:
```
# import some data to play with
data = pd.read_csv("weather.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
Temperature_c           10000 non-null float64
Humidity                10000 non-null float64
Wind_Speed_kmh          10000 non-null float64
Wind_Bearing_degrees    10000 non-null int64
Visibility_km           10000 non-null float64
Pressure_millibars      10000 non-null float64
Rain                    10000 non-null int64
Description             10000 non-null object
dtypes: float64(5), int64(2), object(1)
memory usage: 625.1+ KB
```

In [3]:
```
data.shape
```

Out[3]:
```
(10000, 8)
```

In [4]:
```
# Kiểm tra dữ liệu null
print(data.isnull().sum())
# => Không có dữ liệu null
```

```
Temperature_c           0
Humidity                0
Wind_Speed_kmh          0
Wind_Bearing_degrees    0
Visibility_km           0
Pressure_millibars      0
Rain                    0
Description             0
dtype: int64
```

In [5]:
```
# HV tự tìm cách fill dữ liệu thiếu/drop dựa trên các kiến thức đã học
#data = data.dropna()
```

In [6]:
```
data.head()
```

Out[6]:

	Temperature_c	Humidity	Wind_Speed_kmh	Wind_Bearing_degrees	Visibility_km	Pressure_millil
0	-0.555556	0.92	11.2700	130	8.0500	102
1	21.111111	0.73	20.9300	330	16.1000	101
2	16.600000	0.97	5.9731	193	14.9086	101
3	1.600000	0.82	3.2200	300	16.1000	103
4	2.194444	0.60	10.8836	116	9.9820	102

In [7]:
```
data.tail()
```

Out[7]:

	Temperature_c	Humidity	Wind_Speed_kmh	Wind_Bearing_degrees	Visibility_km	Pressure_n
9995	10.022222	0.95	10.2396	20	4.0089	
9996	8.633333	0.64	11.0446	80	9.9820	
9997	5.977778	0.93	11.0446	269	14.9086	
9998	9.788889	0.78	8.1788	231	7.8246	
9999	11.138889	0.79	14.2485	131	10.2557	

In [8]:
```
# The columns that we will be making predictions with.
inputs = data.drop(["Temperature_c"], axis=1)
inputs.shape
```

Out[8]: (10000, 7)

In [9]:
```
inputs.head()
```

Out[9]:

	Humidity	Wind_Speed_kmh	Wind_Bearing_degrees	Visibility_km	Pressure_millibars	Rain	Des
0	0.92	11.2700	130	8.0500	1021.60	0	
1	0.73	20.9300	330	16.1000	1017.00	1	
2	0.97	5.9731	193	14.9086	1013.99	1	
3	0.82	3.2200	300	16.1000	1031.59	1	
4	0.60	10.8836	116	9.9820	1020.88	1	

In [10]:
```
inputs = pd.get_dummies(inputs)
inputs.head()
```

Out[10]:

	Humidity	Wind_Speed_kmh	Wind_Bearing_degrees	Visibility_km	Pressure_millibars	Rain	Des
0	0.92	11.2700	130	8.0500	1021.60	0	
1	0.73	20.9300	330	16.1000	1017.00	1	
2	0.97	5.9731	193	14.9086	1013.99	1	
3	0.82	3.2200	300	16.1000	1031.59	1	
4	0.60	10.8836	116	9.9820	1020.88	1	

In [11]:
```
#inputs.info()
```

In [12]:
```
# The column that we want to predict.
outputs = data["Temperature_c"]
outputs = np.array(outputs)
outputs.shape
```

Out[12]:
```
(10000,)
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
                                                    test_size=0.3,
                                                    random_state=42)
```

In [14]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score
```

In [15]:
```
# Create decision tree regressor object
model = DecisionTreeRegressor()
# Train model
model.fit(X_train, y_train)
```

Out[15]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, 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=None, splitter='best')
```

In [16]:
```
# Kiểm tra độ chính xác
print("The Train/ Score is: ", model.score(X_train,y_train)*100,"%")
print("The Test/ Score is: ", model.score(X_test,y_test)*100,"%")
```

```
The Train/ Score is:  100.0 %
The Test/ Score is:  78.42765805520914 %
```

In [30]:
```
# Tính MSE
from sklearn import metrics
y_pred = model.predict(X_test)
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Squared Error: 18.866824794381582
Mean Absolute Error: 3.2400592592696666
```

Nhận xét:

- Training và Testing chênh nhau ~22% => có hiện tượng overfitting
- Mô hình trên cho R^2 khá ~ 0.78, cho thấy nó fit 78% dữ liệu
- MSE ~ 19 & MAE ~ 3.3 => mô hình chưa ổn lắm, cần tìm cách giải quyết overfitting

In [18]:
```
df = pd.DataFrame({'Actual': pd.DataFrame(y_test)[0].values,
                   'Prediction': pd.DataFrame(y_pred)[0].values})
df.head(10)
```

Out[18]:

	Actual	Prediction
0	-2.727778	-0.038889
1	11.094444	16.288889
2	1.122222	0.583333
3	-2.850000	-7.188889
4	7.777778	6.594444
5	2.105556	3.722222
6	19.877778	19.977778
7	6.066667	13.888889
8	1.111111	1.138889
9	30.111111	28.905556

In [19]:
```
# Xuất model
import pickle
# Save to file in the current working directory
pkl_filename = "weather.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(model, file)
```

In [20]:
```
with open(pkl_filename, 'rb') as file:
    w_model = pickle.load(file)
```

In [21]:
```
w_model
```

Out[21]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, 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=None, splitter='best')
```

In [22]:
```
# Có giải pháp nào tốt hơn không?
```

In [23]:
```
from sklearn.preprocessing import StandardScaler
```

In [24]:
```
sc = StandardScaler()
sc.fit(X_train)
X_train_new = sc.transform(X_train)
X_test_new = sc.transform(X_test)
```

In [25]:
```
model_new = DecisionTreeRegressor()
model_new.fit(X_train_new, y_train)
```

Out[25]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, 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=None, splitter='best')
```

In [26]:
```
model_new.score(X_train_new, y_train)
```

Out[26]:
```
1.0
```

In [27]:
```
model_new.score(X_test_new, y_test)
```

Out[27]:
```
0.7827732115777397
```

In [28]:
```
# Tính MSE
from sklearn import metrics
y_pred_new = model.predict(X_test_new)
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred_new))
```

```
Mean Squared Error: 87.16790886765631
```

In [29]:
```
# Scaler không đem lại kết quả tốt hơn
```