TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

KHOA TOÁN – TIN HỌC

….📖….

BÁO CÁO CUỐI KỲ

Môn Nhập môn Khoa học dữ liệu – MTH10171

Học kỳ II (2021 - 2022)

TÌM HIỂU, XÂY DỰNG VÀ KIỂM THỬ CÁC MÔ HÌNH MÁY HỌC TRONG LĨNH VỰC KHOA HỌC DỮ LIỆU

Tên đề tài: Xây dựng mô hình dự đoán thời tiết – Weather Prediction.

Nhóm: 15

Tên thành viên:

Hồ Ngọc Ân – 20280001.

Trần Tuấn Thái – 20280082.

Huỳnh Quang Trung – 20280108.

Hỏa Ngọc Tú – 20280111.

Giảng viên hướng dẫn: ThS. Hà Văn Thảo.

TP. Hồ Chí Minh, tháng 06 năm 2022

MỤC LỤC

MỞ ĐẦU

Học phần Nhập môn Khoa học dữ liệu – MTH10171 nhằm cung cấp cho sinh viên các kiến thức cơ bản trong lĩnh vực Khoa học dữ liệu, một lĩnh vực liên ngành về các phương pháp, các quá trình, và các hệ thống có khả năng học và phát hiện tri thức từ lượng dữ liệu ban đầu. Các phương pháp và mô hình trong Khoa học dữ liệu sẽ giúp con người, máy tính đưa ra các quyết định và phán đoán tốt hơn trong thực tế. Trong môn học này, các phương pháp, mô hình từ Học máy (Machine Learning), Khai phá dữ liệu (Data Mining), và Thống kê (Statistics) cũng sẽ được giới thiệu nhằm cung cấp cho sinh viên những kiến thức, kỹ năng cơ bản trong lĩnh vực khoa học dữ liệu.

Đề tài kết thúc môn học lần này nhằm giúp sinh viên nắm được và vận dụng các bước chính khi đào tạo một mô hình học máy bao gồm phân tích dữ liệu, tạo giả thuyết, lấy dữ liệu, tiền xử lý, phân tích, đánh giá chất lượng, đưa ra các phán đoán. Với đề tài "Xây dựng mô hình dự đoán thời tiết – Weather Prediction" nhóm chúng em mong rằng sẽ mang đến được những kết quả tốt nhất có thể để chia sẻ những thông tin hữu ích, đóng góp vào bộ tài nguyên cho ngành học của chúng ta thêm phong phú hơn, làm nguồn tham khảo cho tất cả mọi người.

Nhóm sinh viên thực hiện đề tài.

NỘI DUNG

1. GIỚI THIỆU.

1.1. Giới thiệu chung về đề tài.

Trong bài báo cáo này, nhóm sẽ sử dụng bộ dữ liệu "seattle-weather.csv" từ Kaggle để dùng trong việc phân tích và xây dựng các mô hình dự đoán tình trạng thời tiết dựa trên các điều kiện. Đây là một bộ dữ liệu bao gồm các điều kiện thời tiết dựa trên các mẫu và tình trạng thời tiết gắn với các điều kiện đó. Liên kết đến bộ dữ liệu: https://www.kaggle.com/datasets/ananthr1/weather-prediction.

Các mô hình được sử dụng trong bài báo cáo: Logistic Regression, Decision Tree, K-Nearest Neighbor Classifier (KNN).

1.2. Mục tiêu của đề tài.

Sau khi thực hiện xong đề tài, sinh viên biết được quy trình, cách thức để xây dựng được một mô hình Machine Learning để có thể ứng dụng vào giải quyết các bài toán phù hợp.

Rèn luyện thêm kỹ năng làm sạch và phân tích dữ liệu, tiền xử lý và hiểu được bộ dữ liệu trước khi tiến hành đào tạo một mô hình.

Bài báo cáo được hoàn thành dựa trên quan điểm tăng độ chính xác của kết quả bài toán nhờ vào việc xây dựng nhiều mô hình Machine Learning khác nhau.

2. NỘI DUNG THỰC HIỆN.

2.1. Các thư viện cần thiết trong quá trình xử lý bài toán.

Việc gọi các thư viện cần sử dụng đến trong quá trình xử lý dữ liệu và đào tạo mô hình là một bước cực kỳ quan trọng cần phải được thực hiện đầu tiên trước khi xử lý các vấn đề khác. Ở bài toán này, chúng ta sẽ cần đến các thư viện tiêu biểu như: Numpy, Pandas, Matplotlib… và đặc biệt là thư viện Sklearn chứa các mô hình mà chúng ta sẽ sử dụng đến.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import re
import itertools
import missingno as mso
from scipy import stats
from scipy.stats import ttest_ind
from scipy.stats import pearsonr
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,cla
ssification_report
```

2.2. Tìm hiểu và khám phá bộ dữ liệu.

2.2.1. Thông tin về bộ dữ liệu.

Để đọc bộ dữ liệu "seattle-weather.csv" vào không gian làm việc trong Notebook, chúng ta sẽ sử dụng thư viện pandas bằng cách gọi hàm read csv(). Bộ dữ liệu đã được đọc vào bằng URL giúp linh động trong việc thực hiện giữa các người dùng khác nhau mà không cần bận tâm đến việc phải có file mềm của dữ liệu.

```
# Đọc bộ dữ liệu ở dạng RAW thông qua URL với thư viện pandas
data=pd.read_csv("https://raw.githubusercontent.com/trunghq0205
/Dataset/main/seattle-weather.csv")
data.head()
```

Output[1]:

	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

```
data.shape
```

Output[2]:

```
(1461, 6)
```

Quan sát bộ dữ liệu thu thập được ở trên, ta có thể thấy có 6 cột với tổng cộng 1461 dòng theo các quan sát trong tập dữ liệu. Trong đó, 4 biến liên tục bao gồm: “precipitation”, “temp_max”, “temp_min”, “wind”; một biến ghi nhận thông tin ngày: “date” có dạng YYYY-MM-DD; một biến chỉ tình trạng thời tiết “weather”.

Giải thích rõ ràng hơn về ý nghĩa của các biến: biến “precipitation” chỉ thông tin lượng mưa của tất cả các dạng nước rơi xuống mặt đất như mưa, mưa đá, tuyết rơi hoặc mưa phùn; biến “temp_max” chỉ nhiệt độ cao nhất trong ngày; biến “temp_min” chỉ nhiệt độ thấp nhất trong ngày; biến “wind” lưu thông tin tốc độ gió trong ngày; biến “weather” là biến chỉ tình trạng thời tiết với các điều kiện mẫu.

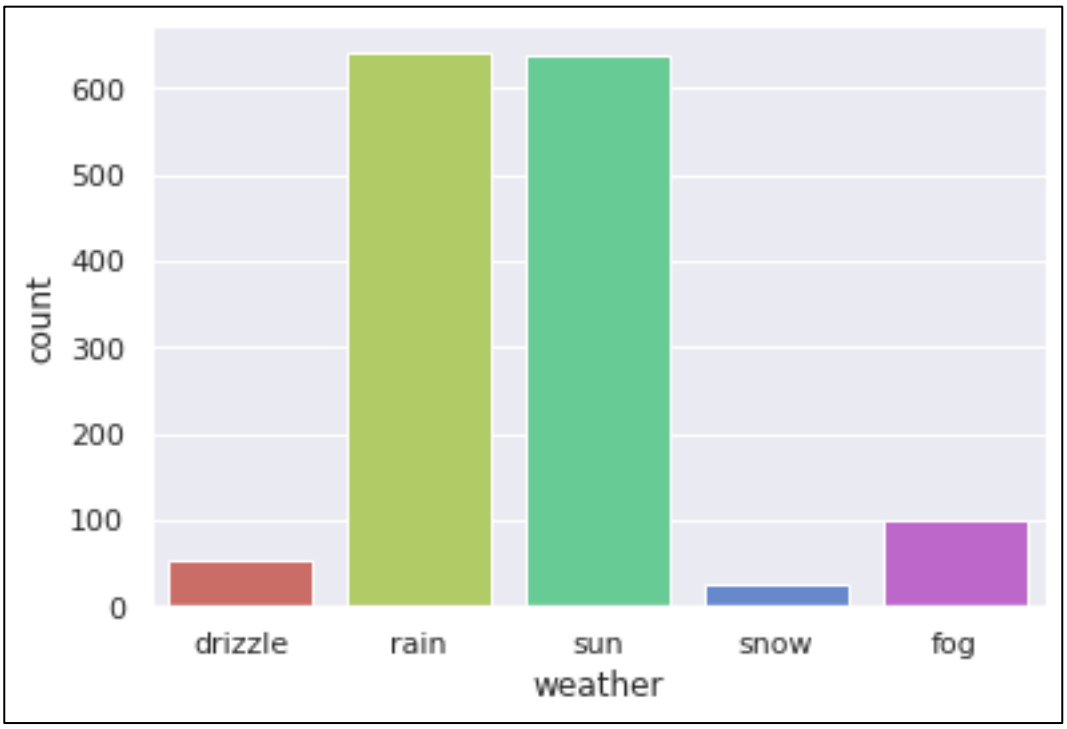
2.2.2. Khám phá các biến dữ liệu.

Ở bước này, chúng ta sẽ tiến hành phân tích các biến trong bộ dữ liệu mà chúng ta đã thu được phía trên. Trước tiên, chúng ta sẽ bắt đầu từ các biến phân loại.

Input[3]:

```
import warnings
warnings.filterwarnings('ignore')
sns.countplot("weather",data=data,palette="hls");
```

Output[3]:



Input[4]:

```
countrain=len(data[data.weather=="rain"])
countsun=len(data[data.weather=="sun"])
countdrizzle=len(data[data.weather=="drizzle"])
countsnow=len(data[data.weather=="snow"])
countfog=len(data[data.weather=="fog"])
print("Percent of Rain:{:2f}%".format((countrain/(len(data
.weather))*100)))
print("Percent of Sun:{:2f}%".format((countsun/(len(data.w
eather))*100)))
print("Percent of Drizzle:{:2f}%".format((countdrizzle/(le
n(data.weather))*100)))
print("Percent of Snow:{:2f}%".format((countsnow/(len(data
.weather))*100)))
print("Percent of Fog:{:2f}%".format((countfog/(len(data.w
eather))*100)))
```

Output[4]:

```
Percent of Rain:43.874059%
Percent of Sun:43.805613%
Percent of Drizzle:3.627652%
Percent of Snow:1.779603%
Percent of Fog:6.913073%
```

Từ đồ thị và các phân tích trên, ta có thể thấy tập dữ liệu chứa phần lớn là tình trạng thời tiết rain và sun với hơn 600 dòng dữ liệu và xấp xỉ nhau khi chiếm 43.3% bộ dữ liệu. Đối với các tình trạng thời tiết như snow, fog và drizzle thì có khoảng dưới 100 dòng dữ liệu với dưới 10% bộ dữ liệu.

Nhận xét chung: Từ việc có ít dữ liệu đề cập đến các tình trạng snow, fog và drizzle thì có thể ảnh hưởng đến độ chính xác của mô hình khi dự đoán các tình trạng thời tiết như snow, fog và drizzle khi quá quá ít dữ liệu để đào tạo cho mô hình.

2.2.3. Tìm hiểu về phân phối của các biến liên tục.

Input[5]:

```
data[["precipitation","temp_max","temp_min","wind"]].de
scribe()
```

Output[5]:

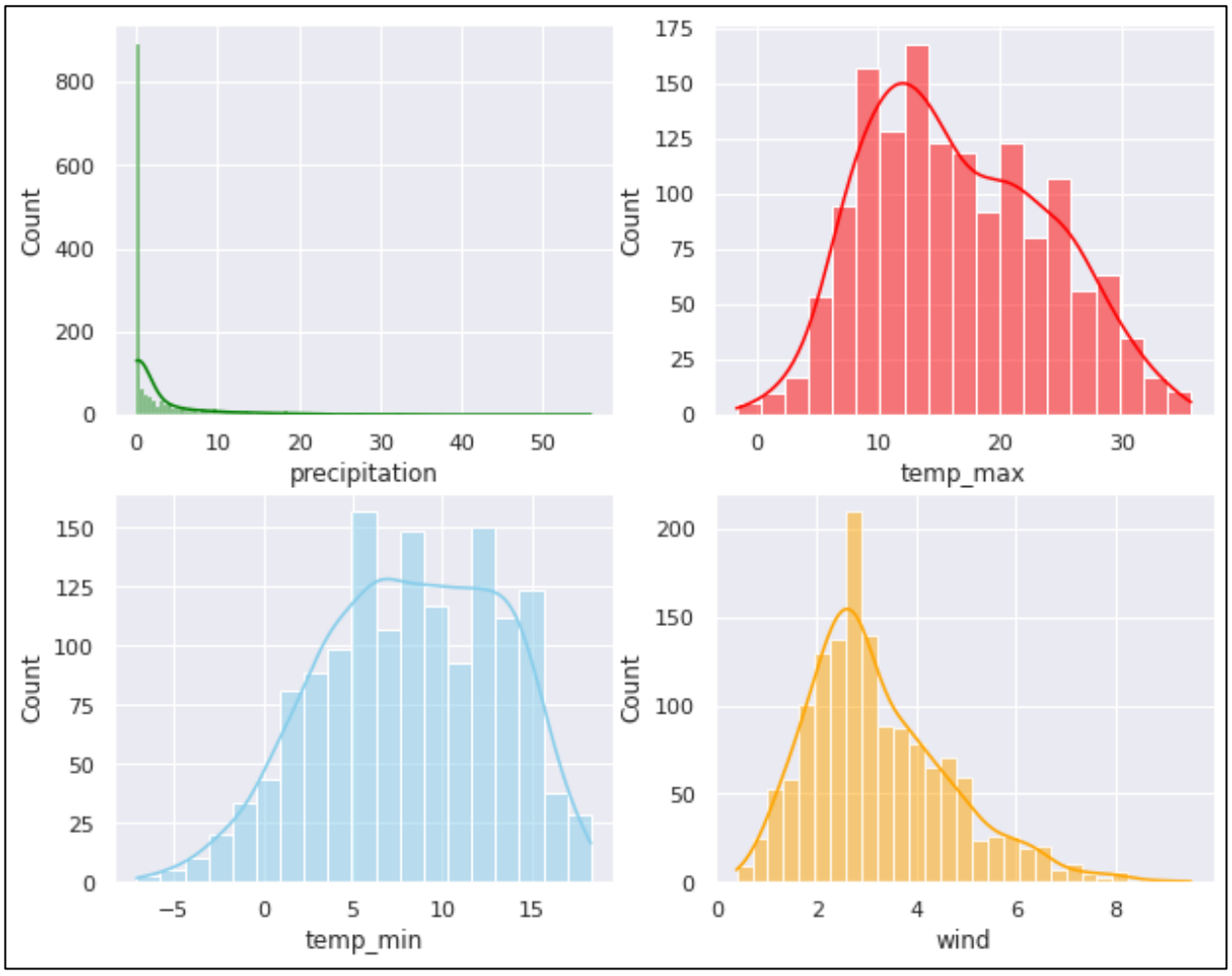
	precipitation	temp_max	temp_min	wind
count	1461.000000	1461.000000	1461.000000	1461.000000
mean	3.029432	16.439083	8.234771	3.241136
std	6.680194	7.349758	5.023004	1.437825
min	0.000000	-1.600000	-7.100000	0.400000
25%	0.000000	10.600000	4.400000	2.200000
50%	0.000000	15.600000	8.300000	3.000000
75%	2.800000	22.200000	12.200000	4.000000
max	55.900000	35.600000	18.300000	9.500000

Để xem và dự đoán được phân phối của các biến dữ liệu liên tục trên, ta sử dụng đồ thị Histogram với hàm vẽ histplot trong thư viện Seaborn.

Input[6]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=data,x="precipitation",kde=True,ax=axs[0,0],color='green');
sns.histplot(data=data,x="temp_max",kde=True,ax=axs[0,1],color='red');
sns.histplot(data=data,x="temp_min",kde=True,ax=axs[1,0],color='skyblue');
sns.histplot(data=data,x="wind",kde=True,ax=axs[1,1],color='orange');
```

Output[6]:



Từ các đồ thị phía trên, ta thấy rõ ràng phân phối của “precipitation”, “wind” và có độ lệch tích cực (lệch phải), đuôi bên phải dài hơn đuôi bên trái. Phân phối của “temp_min” có độ lệch tiêu cực (lệch trái). Đối với phân phối của biến “temp_max” cũng có xu hướng lệch phải. Và cả bốn biến đều có một số giá trị ngoại lai (Outlier).

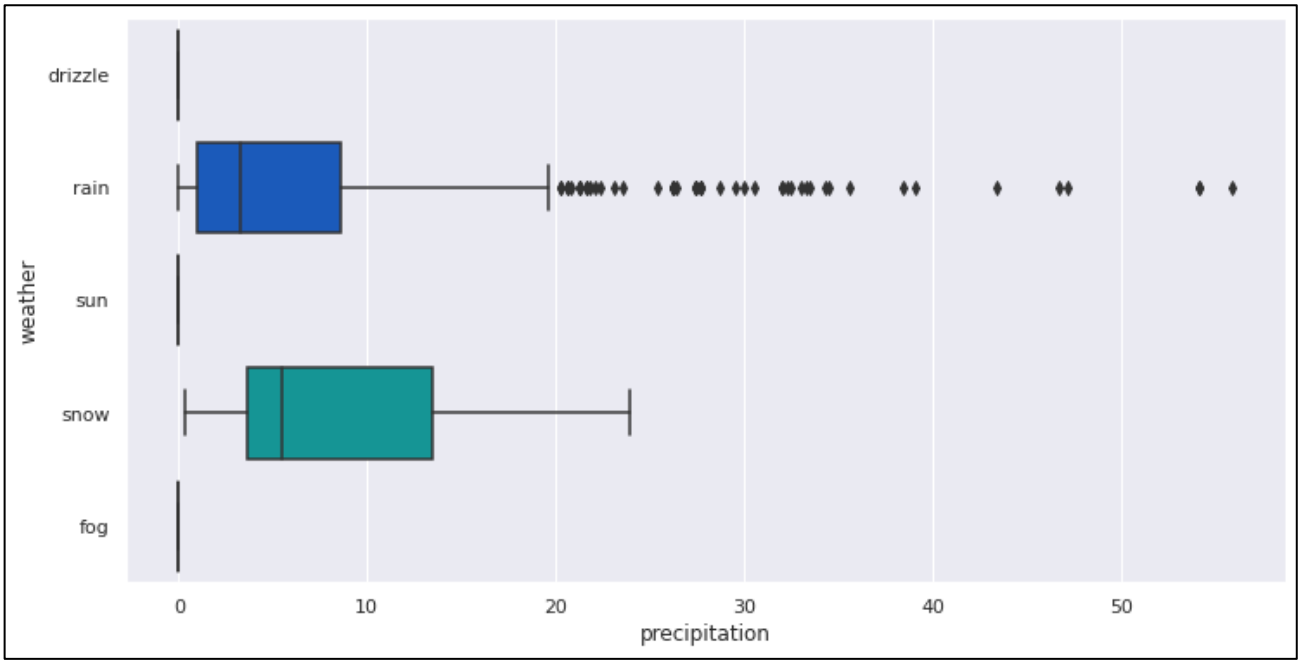
2.2.4. Xác định các điểm ngoại lai (Outlier) và tính lệch của các biến liên tục.

Nhận diện các điểm ngoại lai bằng đồ thị Boxplot là một phương pháp được sử dụng phổ biến dựa trên đặc điểm phân phối chuẩn dữ liệu trong phân tích thống kê. Ở bước này, ta sẽ sử dụng hàm boxplot trong thư viện Seaborn để vẽ đồ thị histogram tương quan giữa các biến “precipitation”, “wind”, “temp_max”, “temp_min” đối với “weather”.

Input[7]:

```
plt.figure(figsize=(12,6))
sns.boxplot("precipitation","weather",data=data,palette="winter")
;
```

Output[7]:

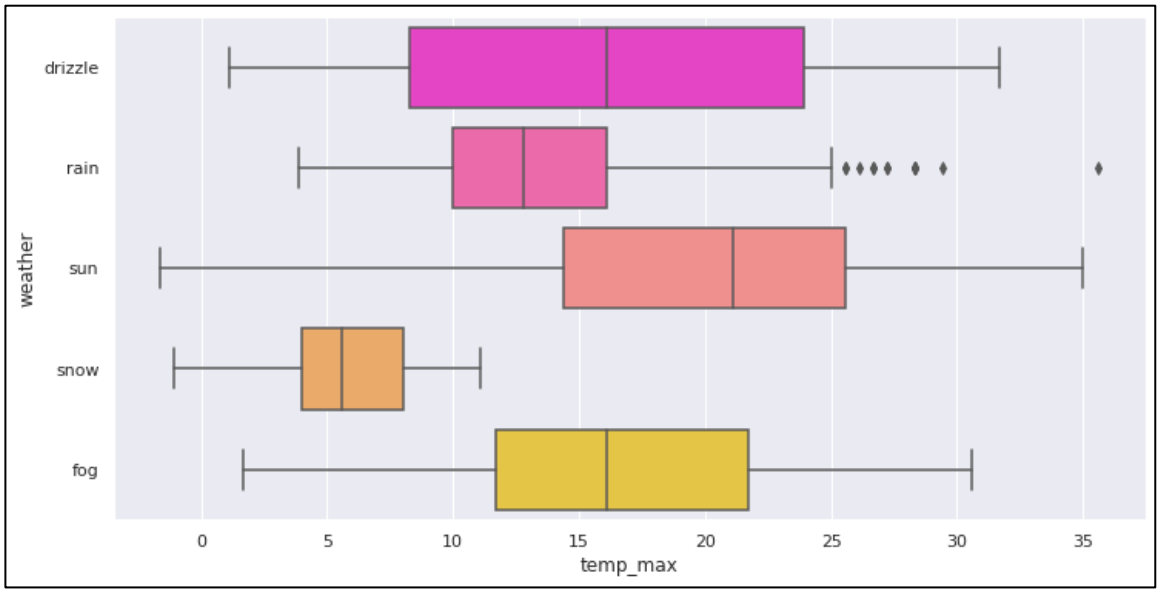


Từ đồ thị boxplot giữa Weather và Precipitation phía trên, giá trị của Rain có nhiều dữ liệu ngoại lai dương và cả Rain và Snow đều bị lệch phải/có độ lệch dương.

Input[8]:

```
plt.figure(figsize=(12,6))
sns.boxplot("temp_max","weather",data=data,palett
e="spring");
```

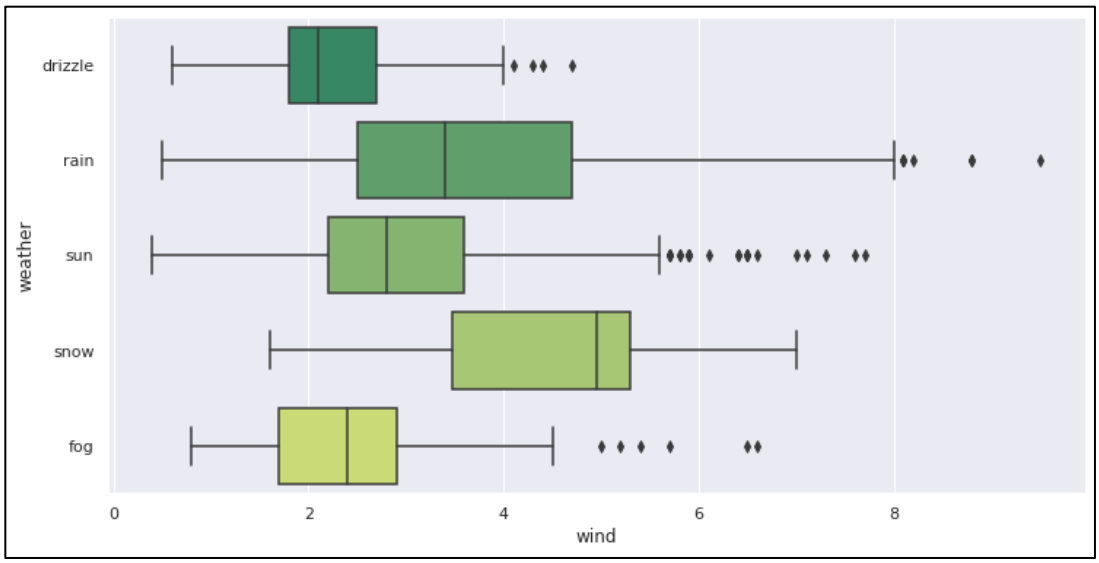
Output[8]:



Input[9]:

```
plt.figure(figsize=(12,6))
sns.boxplot("wind","weather",data=data,palette="s
ummer");
```

Output[9]:

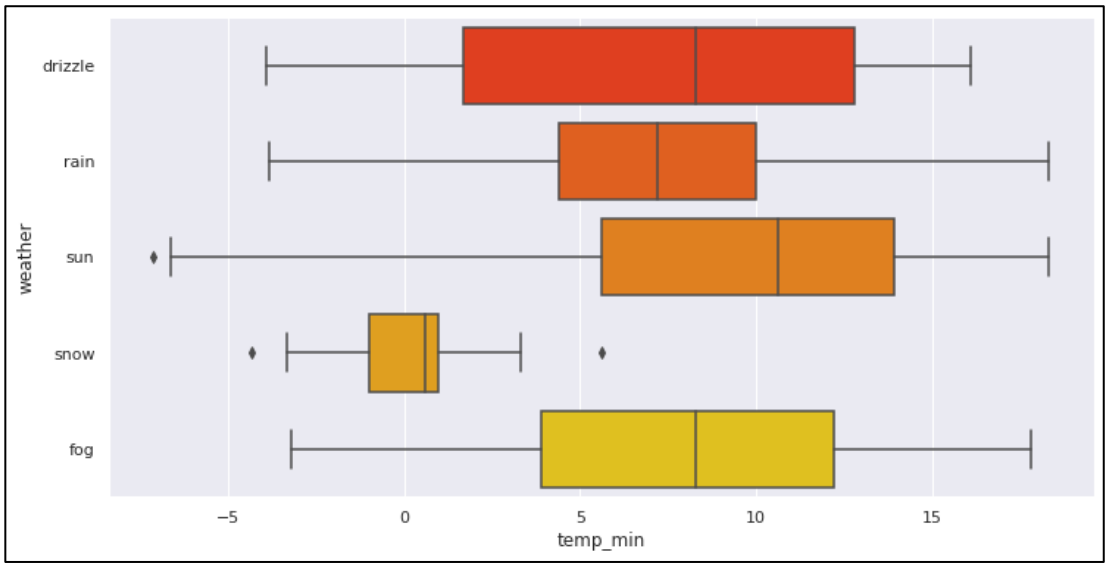


Từ các đồ thị boxplot phía trên, chúng ta thấy rằng mỗi thuộc tính của "weather" có một vài dữ liệu ngoại lai dương và bao gồm cả hai kiểu lệch trái và lệch phải.

Input[10]:

```
plt.figure(figsize=(12,6))
sns.boxplot("temp_min","weather",data=data,palette="aut
umn");
```

Output[10]:



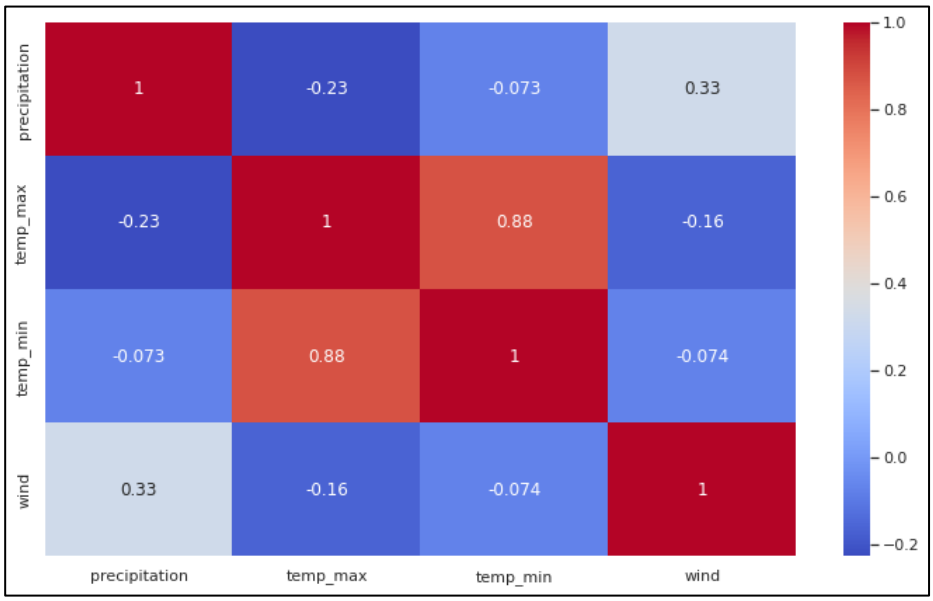
Quan sát được từ đồ thị boxplot giữa “weather” và “temp_min”, một vài dữ liệu có giá trị âm và một vài dữ liệu có cả hai loại giá trị ngoại lai âm và dương, trong đó “snow" bị lệch trái.

2.2.5. Heatmap.

Input[11]:

```
plt.figure(figsize=(12,7))
sns.heatmap(data.corr(),annot=True,cmap='coolwarm');
```

Output[11]:



Dựa vào Heatmap ta thấy có một mối tương quan đồng biến giữa “temp_max” và “temp_min”.

2.2.6. Sự tương quan, ý nghĩa thống kê giữa các biến.

Để kiểm tra ý nghĩa thống kế giữa hai biến trong bộ dữ liệu, ta sử dụng hàm ttest_ind(), đây là hàm có sẵn trong thư viện scipy.stats(). Nó hỗ trợ thực kiểm kiểm định giả thuyết thống kê với giả thuyết H0 rằng hai mẫu độc lập có giá trị trung bình (kỳ vọng) giống hệt nhau và đối thuyết rằng giá trị trung bình là khác nhau (kiểm định 2 phía).

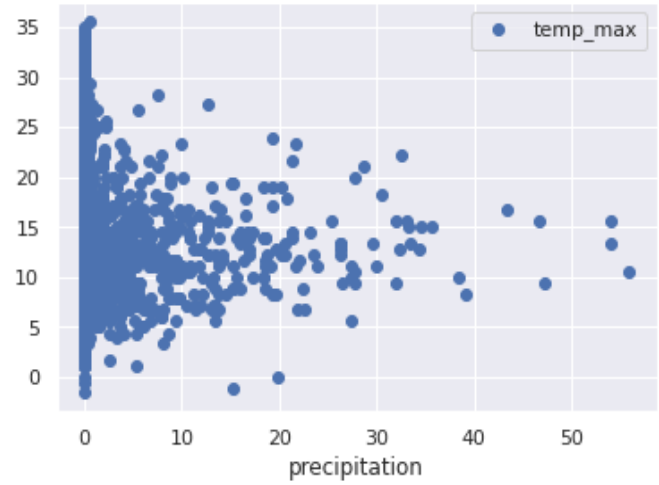
Nếu 2 biến đều có ý nghĩa thống kê, tức là chúng thực sự có trung bình khác nhau có nghĩa là sự hiện diện của nó trong bộ dữ liệu sẽ ảnh hưởng đến kết quả sau này. Ở đây chính là kết quả dự đoán tình trạng thời tiết dựa trên các biến này.

Input[12]:

```
data.plot("precipitation","temp_max",style='o')
print("Pearson correlation:",data["precipitation"].corr(data
["temp_max"]))
print("T Test and P value:",stats.ttest_ind(data["precipitat
ion"],data["temp_max"]))
```

Output[12]:

```
Pearson correlation: -0.22855481643297046
T Test and P value: Ttest_indResult(statistic=-51.60685279531918, pvalue=0.0)
```

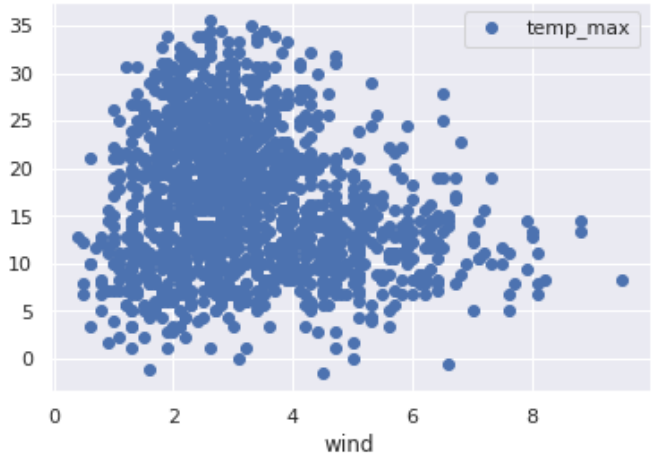


Input[13]:

```
data.plot("wind","temp_max",style='o')
print("Pearson correlation:",data["wind"].corr(data["temp_ma
x"]))
print("T Test and P value:",stats.ttest_ind(data["wind"],dat
a["temp_max"]))
```

Output[13]:

```
Pearson correlation: -0.16485663487495486
T Test and P value: Ttest_indResult(statistic=-67.3601643301846, pvalue=0.0)
```



Theo kết quả của ttest và p-value tính được bằng 0 từ trên thì chứng tỏ rằng giả thuyết H0 trong các cột tương ứng bị bác bỏ và các cột đều có ý nghĩa thống kê và đều có ảnh hưởng đến kế quả dự đoán.

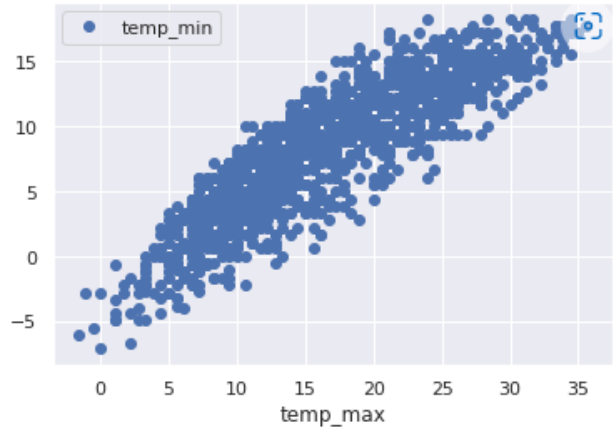
Đồng thời, ta cũng thấy hệ số tương quan giữa các cặp biến trên đều nằm trong khoảng -1 < r < 0, điều này có nghĩa là chúng có mối tương quan yếu với nhau hay có hệ số tương quan âm. Nghĩa là giá trị biến x tăng thì giá trị biến y giảm và ngược lại, giá trị biến y tăng thì giá trị biến x giảm.

Input[14]:

```
data.plot("temp_max","temp_min",style='o')
print("Pearson correlation:",data["temp_max"].corr(data["t
emp_min"]))
print("T Test and P value:",stats.ttest_ind(data["temp_max
"],data["temp_min"]))
```

Output[14]:

```
Pearson correlation: 0.8756866637108172
T Test and P value: Ttest_indResult(statistic=35.22645781239284, pvalue=7.526905087245157e-227)
```



Dựa vào đồ thị Output[14], ta có thể nhận xét rằng là biến "temp_min" và biến "temp_max" có mối quan hệ tương quan dương với nhau và mối quan hệ tuyến tính này khá mạnh. Nghĩa là giá trị biến x tăng thì giá trị biến y tăng và ngược lại, giá trị biến y tăng thì giá trị biến x cũng tăng.

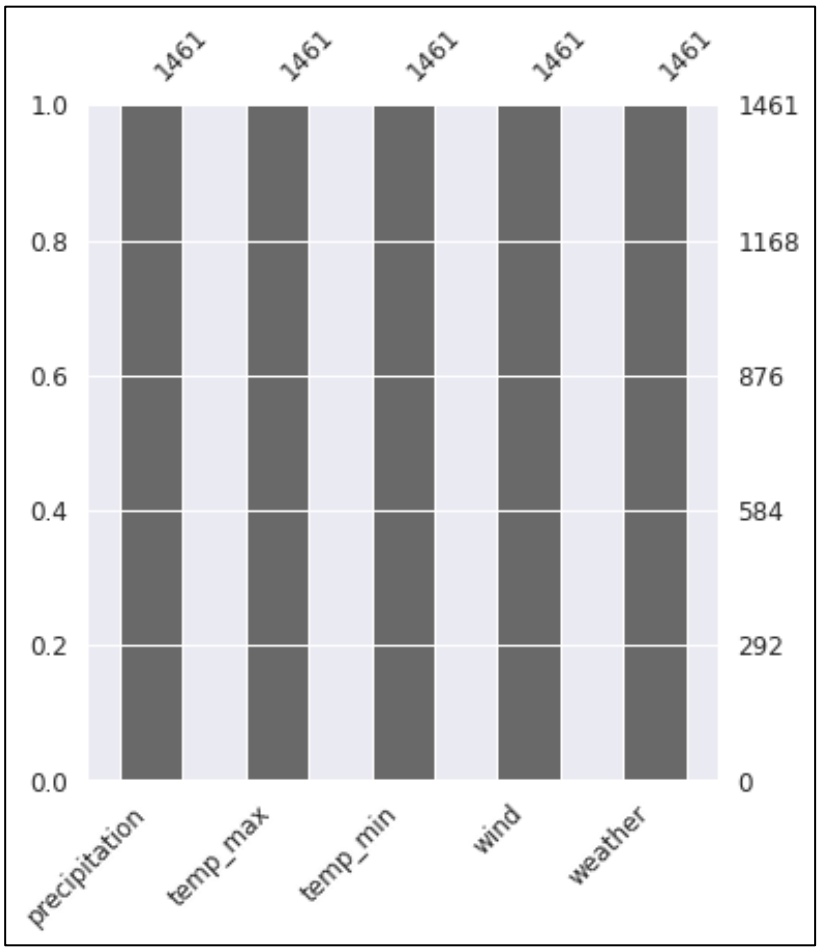
2.2.7. Giá trị NULL.

Tiếp theo, ta tiến hành kiểm tra xem trong bộ dữ liệu có tồn tại giá trị là NULL hay không.

Input[15]:

```
plt.figure(figsize=(12,6))
axz=plt.subplot(1,2,2)
mso.bar(data.drop(["date"],axis=1),ax=axz,fontsize=12
);
```

Output[15]:



Bằng cách quan sát biểu đồ ở phía trên, ta có thể kết luận rằng không tồn tại các giá trị NULL ở các biến bởi vì các cột đều có 1461 quan sát bằng đúng với số dòng của dữ liệu.

2.3. Tiền xử lý và làm sạch dữ liệu.

Đây là bước rất quan trọng trước khi đào tạo một mô hình Machine Learning, nó bao gồm nhiều bước thực hiện chẳng hạn như loại bỏ các điểm ngoại lai, loại các biến không cần thiết, không được sử dụng khi xây dựng mô hình hay xử lý các phân phối lệch. Đồng thời cũng thực hiện gắn nhãn dữ liệu và phân tách bộ dữ liệu thành các tập dữ liệu đào tạo (x_train, y_train) và các tập dữ liệu kiểm thử (x_test, y_test).

2.3.1. Loại bỏ các biến không cần thiết.

Trong bộ dữ liệu này, biến "date" chính là biến dữ liệu không cần thiết, không cần sử dụng đến trong quá trình xây dựng mô hình dự báo của chúng ta. Vì vậy ta tiến hành loại bỏ biến này ra khỏi dữ liệu sử dụng hàm drop().

Input[16]

```
df=data.drop(["date"],axis=1)
df.head()
```

Output[16]:

	precipitation	temp_max	temp_min	wind	weather
0	0.0	12.8	5.0	4.7	drizzle
1	10.9	10.6	2.8	4.5	rain
2	0.8	11.7	7.2	2.3	rain
3	20.3	12.2	5.6	4.7	rain
4	1.3	8.9	2.8	6.1	rain

2.3.2. Loại bỏ các điểm Outlier và các giá trị vô hạn.

Vì trong bộ dữ liệu trên có chứa các giá trị ngoại lai (Outlier), chúng ta sẽ loại bỏ chúng để làm cho bộ dữ liệu đồng đều hơn.

Input[17]:

```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
df=df[~((df<(Q1-
1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
```

Chúng ta loại bỏ đi những điểm Outlier bằng cách tính khoảng tứ phân vị IQR, sau đó loại đi những giá trị nằm ngoài khoảng (`Q1-1.5*IQR, Q3+1.5*IQR`). Những điểm nằm ngoài khoảng này được gọi là những điểm ngoại lai (Outlier).

2.3.3. Xử lý các phân phối lệch.

Ta xử lý hai biến có phân phối lệch là "precipition" và "wind" bằng cách lấy căn bậc 2 của chúng.

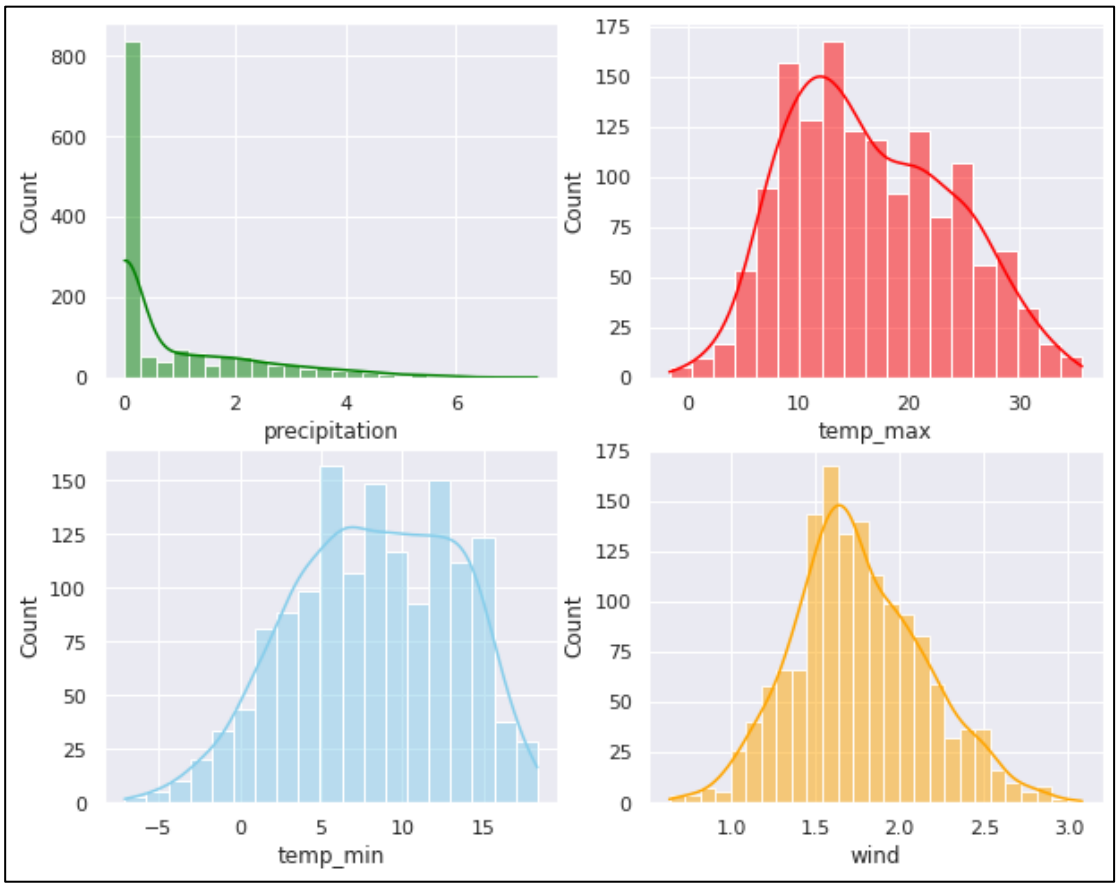
Input[18]:

```
df.precipitation=np.sqrt(df.precipitation)
df.wind=np.sqrt(df.wind)
```

Input[19]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=df,x="precipitation",kde=True,
ax=axs[0,0],color='green')
sns.histplot(data=df,x="temp_max",kde=True,ax=ax
s[0,1],color='red')
sns.histplot(data=df,x="temp_min",kde=True,ax=ax
s[1,0],color='skyblue')
sns.histplot(data=df,x="wind",kde=True,ax=axs[1,
1],color='orange')
```

Output[19]:



2.3.4. Mã hóa dữ liệu ở biến weather.

Mã hóa các nhãn dữ liệu ở biến weather thành các giá trị từ 0 - 4 sử dụng hàm LabelEncoder(). Việc này là cần thiết để xây dựng mô hình dự đoán trong bài toán này.

Input[20]:

```
lc=LabelEncoder()
df["weather"]=lc.fit_transform(df["weather"])
```

Output[20]:

	precipitation	temp_max	temp_min	wind	weather
0	0.000000	12.8	5.0	2.167948	0
2	0.894427	11.7	7.2	1.516575	2
4	1.140175	8.9	2.8	2.469818	2
5	1.581139	4.4	2.2	1.483240	2
6	0.000000	7.2	2.8	1.516575	2

Ta có thể thấy các nhãn dữ liệu trong biến "weather" đã được mã hóa như sau: giá trị 0 tương ứng với Dizzle, giá trị 1 tương ứng với Fog, giá trị 2 tương ứng với rain, giá trị 3 tương ứng với Snow, giá trị 4 tương ứng với Sun.

2.3.5. Phân tách tập dữ liệu thành biến phụ thuộc và biến độc lập.

Tiếp theo, ta tiến hành phân tách giá trị các "precipitation", "wind", "temp_max", "temp_min" thành một biến x độc lập với biến "weather" để chuẩn bị cho công đoạn tách bộ dữ liệu thành các tập dữ liệu đào tạo và kiểm thử.

Input[21]:

```
x=((df.loc[:,df.columns!="weather"]).astype(int)).values[:,0:
]
y=df["weather"].values
```

Sau đó, ta chia bộ dữ liệu thành hai tập dữ liệu riêng biệt bao gồm tập dữ liệu đào tạo và tập dữ liệu kiểm thử với tỷ lệ tương ứng là 9:1. Sử dụng hàm train_test_split(), đây là một hàm có sẵn trong thư viện Sklearn.

Input[22]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=
0.1,random_state=2)
```

2.4. Đào tạo mô hình.

2.4.1. K-Nearest Neighbor Classifier (KNN).

K-Nearest Neighbor Classifer là một bộ phân loại học có giám sát, phi tham số, sử dụng khoảng cách gần nhất để thực hiện phân loại hoặc dự đoán về việc nhóm một điểm dữ liệu riêng lẻ. Được sử dụng trong nhiều ứng dụng khác nhau, một số trường hợp như: xử lý trước dữ liệu, nhận dạng mẫu…

Ta tiến hành đào tạo mô hình sử dụng KNN bằng cách gọi hàm KNeighborsClassifier() từ thư viện sklearn.neighbors.

Input[23]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
knn_score = knn.score(x_test,y_test)
print("KNN Accuracy:", knn_score)
```

Output[23]:

```
KNN Accuracy: 0.75
```

Input[24]:

```
y_pred_knn = knn.predict(x_test)
conf_matrix = confusion_matrix(y_test, y_pred_knn)
print("Confusion Matrix")
print(conf_matrix)
```

Output[24]:

```
Confusion Matrix
[[ 0  0  0  0  1]
 [ 0  2  4  0  5]
 [ 1  0 26  0  6]
 [ 0  0  1  1  1]
 [ 1  5  6  0 64]]
```

Input[25]:

```
print('KNN\n',classification_report(y_test,y_pred_knn,
zero_division=0))
```

Output[25]:

```
KNN
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.29      0.18      0.22        11
           2       0.70      0.79      0.74        33
           3       1.00      0.33      0.50         3
           4       0.83      0.84      0.84        76

    accuracy                           0.75       124
   macro avg       0.56      0.43      0.46       124
weighted avg       0.75      0.75      0.74       124
```

2.4.2. Decision Tree.

Là một công cụ hỗ trợ quyết định sử dụng mô hình quyết định dạng cây và các hệ quả có thể xảy ra của chúng, bao gồm cả kết quả sự kiện may rủi, chi phí tài nguyên và tiện ích. Đó là một cách để hiển thị một thuật toán chỉ chứa các câu lệnh điều khiển có điều kiện.

Cây quyết định thường được sử dụng trong nghiên cứu hoạt động, đặc biệt là trong phân tích quyết định, để giúp xác định chiến lược có nhiều khả năng đạt được mục tiêu nhất, nhưng cũng là một công cụ phổ biến trong học máy.

Ta tiến hành đào tạo mô hình sử dụng Decision Tree bằng cách gọi hàm DecisionTreeClassifier từ thư viện sklearn.tree.

Input[26]:

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dec = DecisionTreeClassifier(max_leaf_nodes=15,random
_state=0)
dec.fit(x_train,y_train)
dec_score = dec.score(x_test,y_test)
print("Decison Tree Accuracy : ", dec score)
```

Output[26]:

```
Decison Tree Accuracy :  0.8306451612903226
```

Input[27]:

```
y_pred_dt = dec.predict(x_test)
conf_matrix = confusion_matrix(y_test, y_pred_dt)
print("Confusion Matrix")
print(conf_matrix)
```

Output[27]:

```
Confusion Matrix
[[ 0  0  0  0  1]
 [ 0  0  0  0 11]
 [ 0  0 25  0  8]
 [ 0  0  0  3  0]
 [ 0  0  1  0 75]]
```

Input[28]:

```
print('Decision Tree\n',classification_report(y_test,y
_pred_dt, zero_division=0))
```

Output[28]:

```
Decision Tree
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        11
           2       0.96      0.76      0.85        33
           3       1.00      1.00      1.00         3
           4       0.79      0.99      0.88        76

    accuracy                           0.83       124
   macro avg       0.55      0.55      0.54       124
weighted avg       0.76      0.83      0.79       124
```

2.4.3. Logistic Regression.

Phương pháp hồi quy logistic là một mô hình hồi quy học có giám sát nhằm dự đoán giá trị đầu ra rời rạc (discrete target variable) y ứng với một vector đầu vào x.

Mục đích của hồi quy logistic là ước tính xác suất của các sự kiện, bao gồm xác định mối quan hệ giữa các tính năng từ đó đự đoán xác suất của các kết quả, nên đối với hồi quy logistic ta sẽ có:

Input: dữ liệu input (ta sẽ coi có hai nhãn là 0 và 1).

Output: Xác suất dữ liệu input rơi vào nhãn 0 hoặc nhãn 1.

Ta tiến hành đào tạo mô hình sử dụng Logistic Regression bằng cách gọi hàm LogisticRegression từ thư viện sklearn.linear_model.

Input[29]:

```
from sklearn.linear_model import LogisticRegression
lg = LogisticRegression()
lg.fit(x_train,y_train)
lg_score = lg.score(x_test,y_test)
print("Logictic Accuracy : ", lg_score)
```

Ouput[29]:

```
Logistic Accuracy :  0.8064516129032258
```

Input[30]:

```
y_pred_lg = lg.predict(x_test)
conf_matrix = confusion_matrix(y_test, y_pred_lg)
print("Confusion Matrix")
print(conf_matrix)
```

Ouput[30]:

```
Confusion Matrix
[[ 0  0  0  0  1]
 [ 0  0  3  0  8]
 [ 0  0 26  0  7]
 [ 0  0  3  0  0]
 [ 0  0  2  0 74]]
```

Input[31]:

```
print('Logistic Regression\n',classification_report(y_
test,y_pred_lg, zero_division=0))
```

Output[31]:

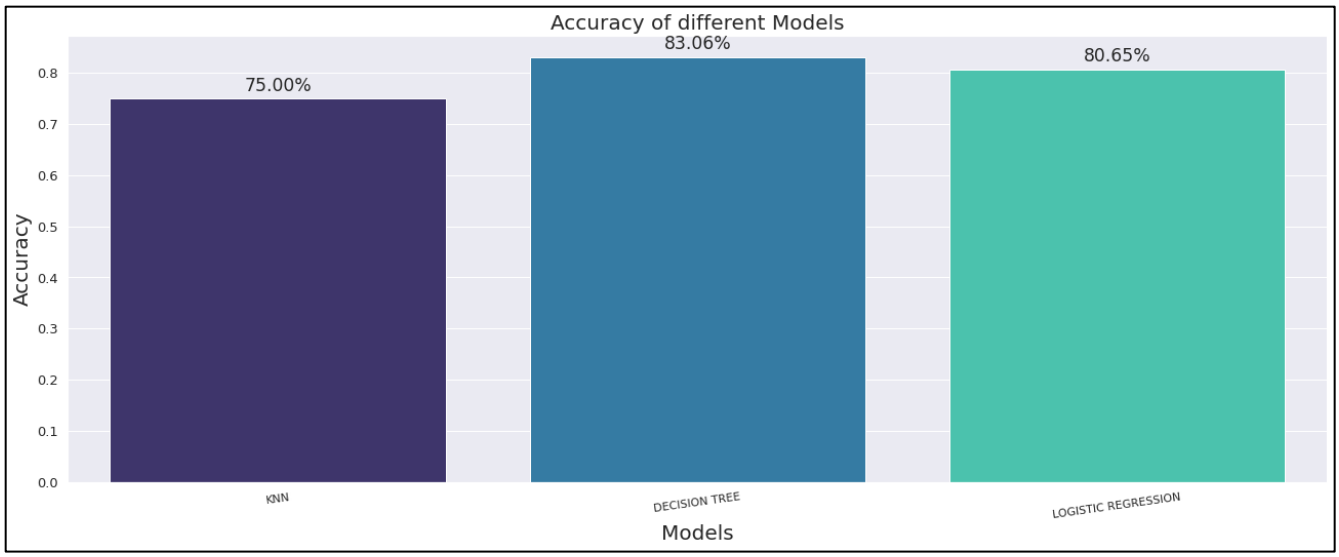
```
Logistic Regression
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        11
           2       0.76      0.79      0.78        33
           3       0.00      0.00      0.00         3
           4       0.82      0.97      0.89        76

    accuracy                           0.81       124
   macro avg       0.32      0.35      0.33       124
weighted avg       0.71      0.81      0.75       124
```

2.4.4. Biểu đồ so sánh các mô hình.

Input[32]:

```
mylist=[]
mylist2=[]
mylist.append(knn_score)
mylist2.append("KNN")
mylist.append(dec_score)
mylist2.append("DECISION TREE")
mylist.append(lg_score)
mylist2.append("LOGISTIC REGRESSION")
plt.rcParams['figure.figsize']=8,6
sns.set_style("darkgrid")
plt.figure(figsize=(22,8))
ax = sns.barplot(x=mylist2, y=mylist, palette = "
mako", saturation =1.5)
plt.xlabel("Models", fontsize = 20 )
plt.ylabel("Accuracy", fontsize = 20)
plt.title("Accuracy of different Models", fontsiz
e = 20)
plt.xticks(fontsize = 11, horizontalalignment = '
center', rotation = 8)
plt.yticks(fontsize = 13)
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.2%}', (x + width/2, y
+ height*1.02), ha='center', fontsize = 'x-
large')
plt.show()
```

Output[32]:



3. KIỂM THỬ KẾT QUẢ DỰ ĐOÁN CỦA MÔ HÌNH.

Chúng ta tiến hành lựa chọn kết quả đào tạo của mô hình sử dụng Decision Tree làm đại diện tiêu biểu để kiểm tra kết quả dự đoán trong đề tài này. Kết quả đào tạo của Decision Tree trong bài toán này đạt được 0.83 độ tin cậy, là mô hình có điểm cao nhất trong các mô hình được sử dụng trong bài báo cáo này.

Input[31]:

```
for i in (range(len(y_test))):
    print("-----------------------------------")
    ot=dec.predict([x_test[i]])
    if(ot==0):
        print("The weather predict is: Drizzle")
    elif(ot==1):
        print("The weather predict is: Fog")
    elif(ot==2):
        print("The weather predict is: Rain")
    elif(ot==3):
        print("The weather predict is: snow")
    else:
        print("The weather predict is: Sun")
```

```
    ac = y_test[i]
    if(ac==0):
        print("The weather actual is: Drizzle")
    elif(ac==1):
        print("The weather actual is: Fog")
    elif(ac==2):
        print("The weather actual is: Rain")
    elif(ac==3):
        print("The weather actual is: snow")
    else:
        print("The weather actual is: Sun")
```

Output[31]: Một vài dòng kết quả được xuất ra.

```
---------------------------------
The weather predict is: Sun
The weather actual is: Sun
---------------------------------
The weather predict is: snow
The weather actual is: snow
---------------------------------
The weather predict is: Sun
The weather actual is: Sun
---------------------------------
The weather predict is: Sun
The weather actual is: Sun
---------------------------------
The weather predict is: snow
The weather actual is: snow
---------------------------------
The weather predict is: Rain
The weather actual is: Rain
---------------------------------
The weather predict is: Sun
The weather actual is: Sun
---------------------------------
The weather predict is: Sun
The weather actual is: Sun
---------------------------------
The weather predict is: Rain
The weather actual is: Rain
---------------------------------
```

Kiểm thử kết quả với dữ liệu được nhập thủ công:

Input[32]:

```
input=[[0.3,15.6,0.0,-5.3]]
ot=dec.predict(input)
print("The weather is:")
if(ot==0):
    print("Drizzle")
elif(ot==1):
    print("Fog")
elif(ot==2):
    print("Rain")
elif(ot==3):
    print("snow")
else:
    print("Sun")
```

Output[32]:

```
The weather is:
Sun
```

4. THÔNG TIN BÀI NỘP (BẢN TRÌNH CHIẾU).

- Đường dẫn đến bài báo cáo trên nền tảng Google Colab: https://colab.research.google.com/drive/1xF1djO43fKVzlKZNV0FfUqmXBlMRsIhV?usp=sharing
- Đường dẫn đến bài báo cáo trên nền tảng Github: https://github.com/trunghq0205/IntroductionToDataScience_MTH10171

KẾT LUẬN

Qua việc xây dựng các mô hình trên, ta thấy rằng khi dự đoán về các tình trạng thời tiết như snow, fog và dizzle mô hình dự đoán không được hiệu quả. Nguyên do là trong tập dữ liệu có quá ít dữ liệu đề cập về 3 tình trạng thời tiết này, làm ảnh hưởng đến độ chính xác của mô hình. Trong khi đó, mô hình dự đoán khá tốt khi làm việc với tình trạng thời tiết rain và sun.

Ta cũng nhận thấy được rằng dữ liệu được đào tạo với mô hình Decision Tree cho độ chính xác cao nhất trong số 3 mô hình được lựa chọn với độ chính xác là 83.06%. Vì vậy, ta có thể kết luận rằng Decision Tree là mô hình phù hợp nhất đối với tập dữ liệu được sử dụng trong số 3 mô hình trên.