UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Department of Information and Communication Technology

SUBJECT: Machine Learning and Data Mining 1

Lecturer: Dr. Le Huu Ton

MIDTERM EXAM REPORT

GROUP 21

Weather Data Analysis and Classification

Student Name:

Hoàng Việt - 22BI13465

Chu Xuân Thăng - 22BI13403

Nguyễn Tường Quang - 22BI13385

Hanoi, 20 May 2024

Table of Contents

Abstract

This project performs an exploratory data analysis (EDA) on a weather dataset, visualizes various weather parameters, and applies multiple classification algorithms to predict weather conditions. The study utilizes data visualization techniques to uncover patterns and relationships within the dataset. Subsequently, five different classification models—Logistic Regression, Support Vector Machine (SVM), Naive Bayes, Decision Tree, and K-Nearest Neighbors (KNN)—are trained and evaluated for their accuracy in predicting weather categories. This comprehensive approach not only enhances our understanding of weather patterns but also demonstrates the effectiveness of different machine learning models in weather prediction.

1. Introduction

Weather forecasting is a critical aspect of daily life and various industries, influencing decision-making processes in agriculture, transportation, emergency management, and more. Accurate weather prediction helps mitigate risks and improve planning. This project aims to analyze weather data to identify trends and patterns and employ machine learning algorithms to predict weather conditions based on historical data.

The dataset used in this analysis includes various weather parameters recorded over time. The initial phase of the project involves data cleaning and exploratory data analysis (EDA) to ensure the quality and integrity of the dataset. Following the EDA, several data visualization techniques are employed to gain insights into the distribution and trends of weather parameters.

The core of the project focuses on applying and comparing different classification algorithms to predict weather conditions. The selected algorithms—Logistic Regression, SVM, Naive Bayes, Decision Tree, and KNN—are evaluated based on their accuracy and performance metrics. The results are visualized using confusion matrices and actual vs. predicted value plots, providing a comprehensive overview of each model's effectiveness.

This study not only showcases the application of machine learning in weather prediction but also provides a detailed workflow for performing similar analyses on other datasets. The insights gained from this project can be utilized to improve weather forecasting models and decision-making processes in weather-dependent sectors.

2. Materials and Methods

Dataset

The dataset utilized for this project comprises weather conditions data for Seattle. It is a commonly used dataset in weather prediction studies. The dataset includes 1461 rows and 6 attributes. Below is a brief description of each attribute:

1. **date**: The date on which the weather observation was recorded.
2. **temp_max**: The maximum temperature recorded on the given date.
3. **temp_min**: The minimum temperature recorded on the given date.
4. **wind**: The wind speed measured on the given date.
5. **weather**: The observed weather condition (e.g., sunny, rainy, etc).

3. Methodology

The methodology section outlines the steps and techniques employed in the analysis and prediction of weather conditions using the provided dataset. The process involves several stages, from data preprocessing to model evaluation. Each step is designed to ensure a

thorough and accurate analysis, leveraging both data visualization and machine learning techniques.

3.1. Import Libraries

To begin with, essential libraries for data manipulation, visualization, and machine learning are imported. This includes libraries such as Pandas, NumPy, Seaborn, Matplotlib, Plotly, and various Scikit-learn modules.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

3.2. Load the Dataset

The weather dataset is loaded into a Pandas DataFrame for analysis.

```
data = pd.read_csv("weather.csv")
data.info()
```

3.3. Exploratory Data Analysis (EDA):

EDA is performed to understand the dataset's structure and identify any missing values or anomalies. This includes checking for null values, converting data types, and examining the unique values in each column

```
print(data.isnull().sum())
# Convert the data type into datetime
```

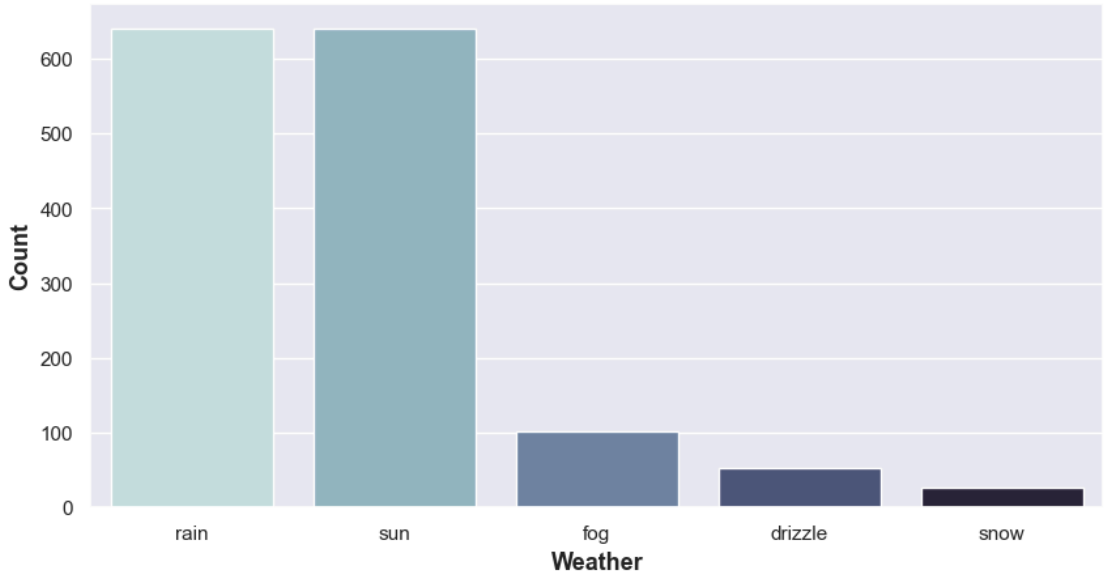
```
data['date'] = pd.to_datetime(data['date'])
print(data.columns)
print(data.nunique())
```

3.4. Data Visualization

Various data visualization techniques are employed to uncover patterns and relationships within the dataset. This includes:

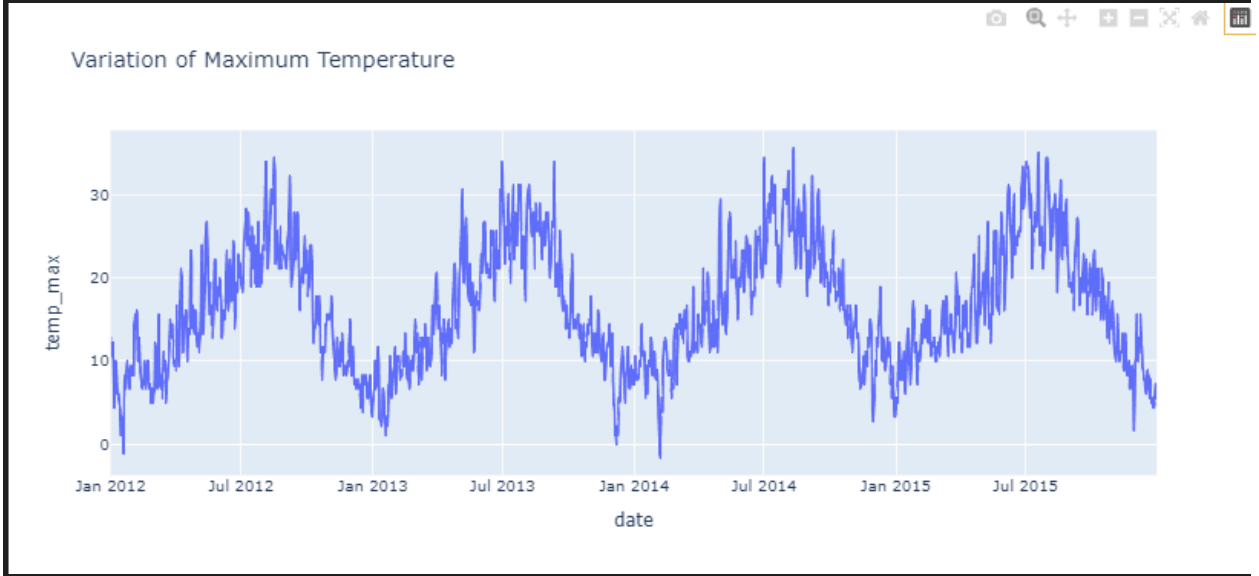
- Value counts for the weather category.

```
print(data['weather'].value_counts())
plt.figure(figsize=(10,5))
sns.set_theme()
sns.countplot(x='weather', data=data, palette="ch:start=.2,rot=-.3",
order=data['weather'].value_counts().index)
plt.xlabel("Weather", fontweight='bold', size=13)
plt.ylabel("Count", fontweight='bold', size=13)
plt.show()
```

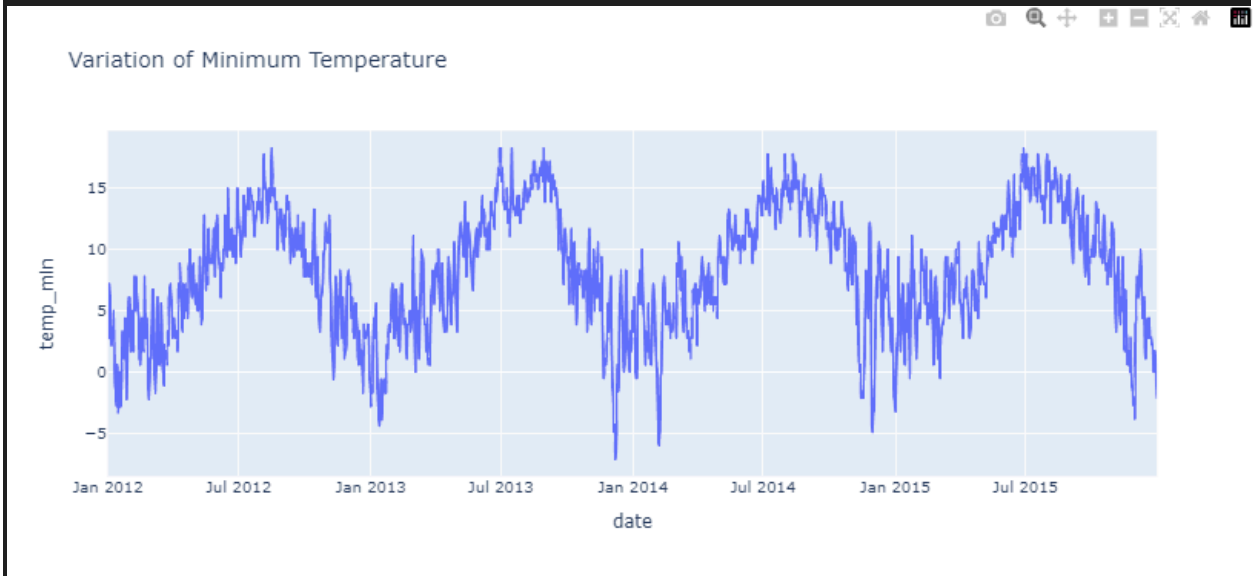


- Line plots for temperature and wind speed variations over time.

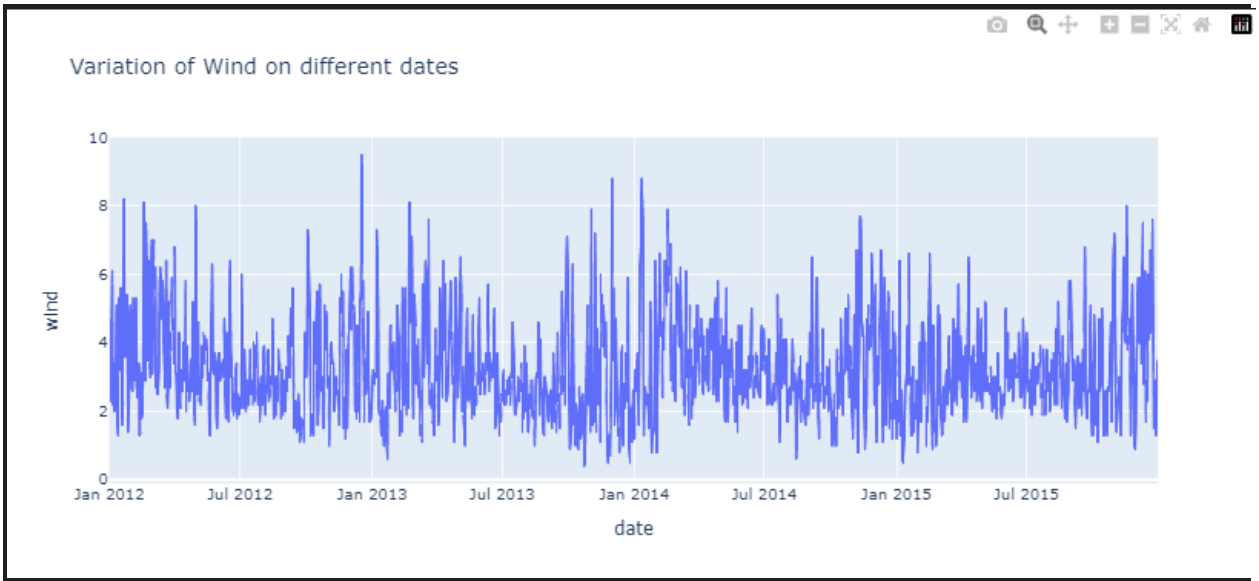
```
fig1 = px.line(data, x='date', y='temp_max', title='Variation of Maximum Temperature')
fig1.show()
```




```
fig2 = px.line(data, x='date', y='temp_min', title='Variation of Minimum Temperature')
fig2.show()
```

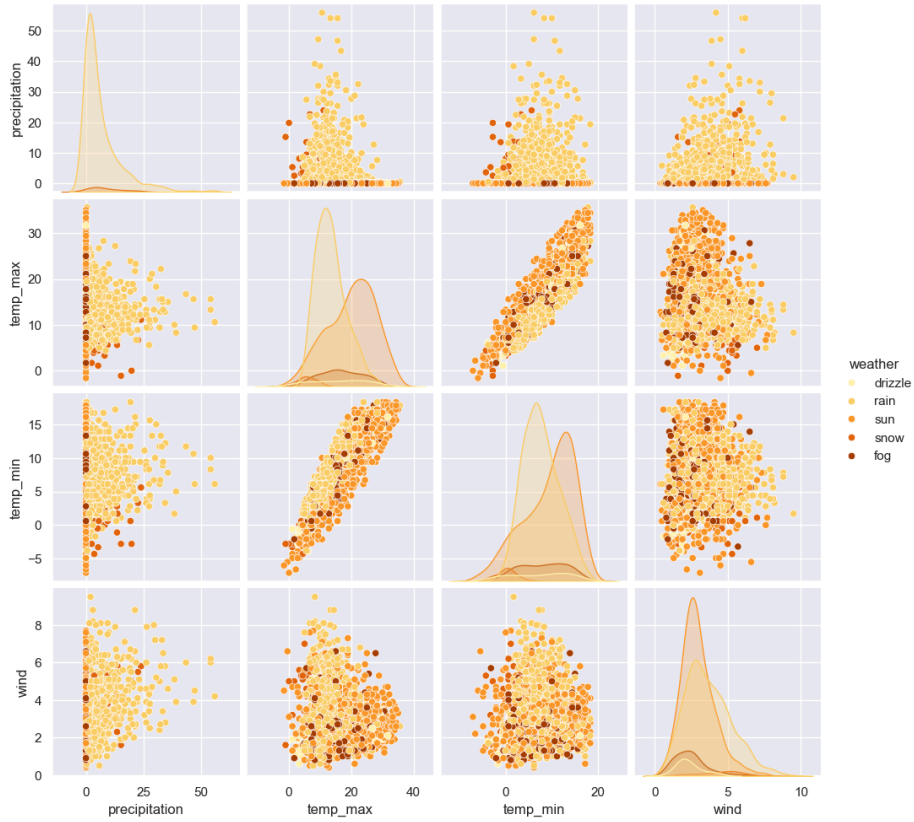



```
fig3 = px.line(data, x='date', y='wind', title='Variation of Wind on different dates')
fig3.show()
```



- Pair plots to examine relationships between different numerical features.

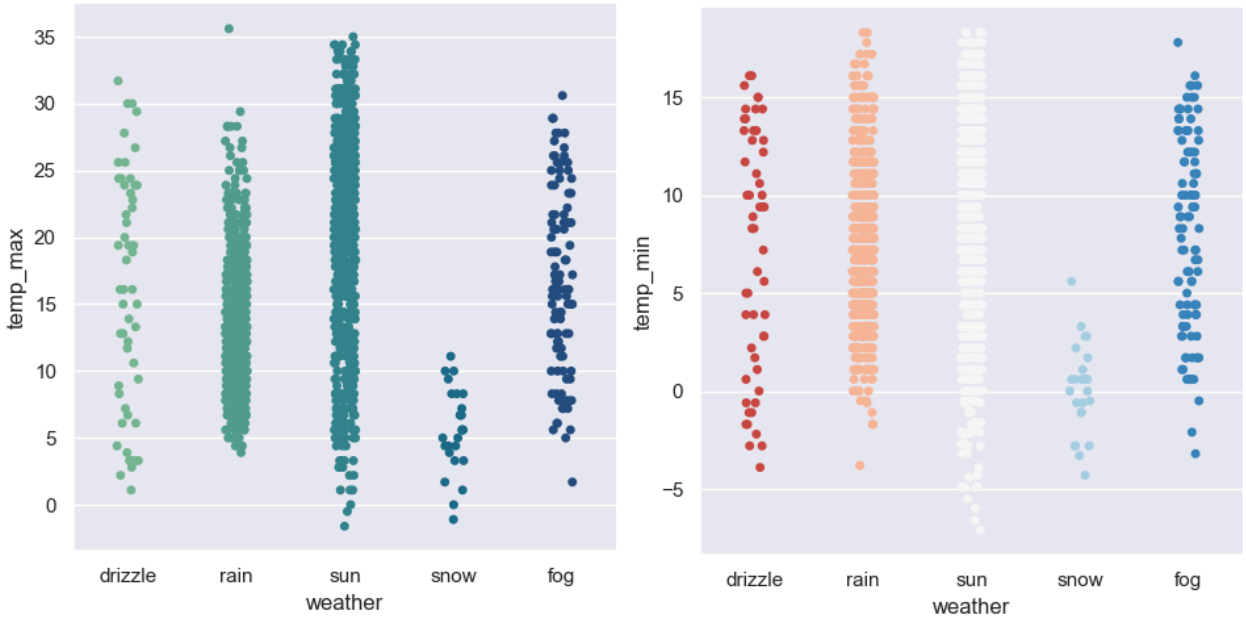
```
plt.figure(figsize=(14,8))
sns.pairplot(data.drop('date', axis=1), hue='weather', palette="YlOrBr")
plt.show()
```



- Violin plots to visualize the distribution of temperature values across different weather categories.

```
plt.figure(figsize=(10,5))
sns.catplot(x='weather', y='temp_max', hue='weather', data=data, palette="crest", legend=False)
plt.show()

plt.figure(figsize=(10,5))
sns.catplot(x='weather',y ='temp_min',hue='weather', data=data,palette = "RdBu", legend=False)
plt.show()
```



3.5. Data Preprocessing

Data preprocessing involves label encoding the categorical target variable and splitting the dataset into features (X) and target (y). Additionally, the date column is dropped as it is not needed for the classification task.

Following data visualization, the data can be pre-processed to convert the data into a form which can be better understood by Machine Learning models. We define a function called LABEL_ENCODING which can be used to convert categorical values into numerical values where in each categorical value is assigned with a unique integer value. We use this concept to convert the values in the weather column into integer values

```
def label_encoding(column):
    from sklearn.preprocessing import LabelEncoder
    label_encoder = LabelEncoder()
    data[column] = label_encoder.fit_transform(data[column])

label_encoding("weather")
data = data.drop('date', axis=1)
x = data.drop('weather', axis=1)
y = data['weather']
```

3.6. Train Test Split

The dataset is split into training and testing sets to evaluate the performance of the machine learning models. Standardization of the features is performed to ensure uniform scaling.

It is important to split the dataset into training data and testing data which can be later used to train the models and evaluate various models performance respectively.

75% of the data is considered as training data and 25% of the data is considered as testing data/test data.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

3.7. Model Training and Evaluation

Five different classification algorithms are trained and evaluated using a custom evaluation function. The models include:

- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Naive Bayes**
- **Decision Tree**
- **K-Nearest Neighbors (KNN)**

Each model is trained on the training set and evaluated on the test set using accuracy scores and confusion matrices.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
def evaluate_model(classifier, model_name):
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{model_name} Accuracy score: {acc}")
    plot_confusion_matrix(y_test, y_pred, model_name)
    return y_pred
```

3.8. Model Implementation

Logistic Regression

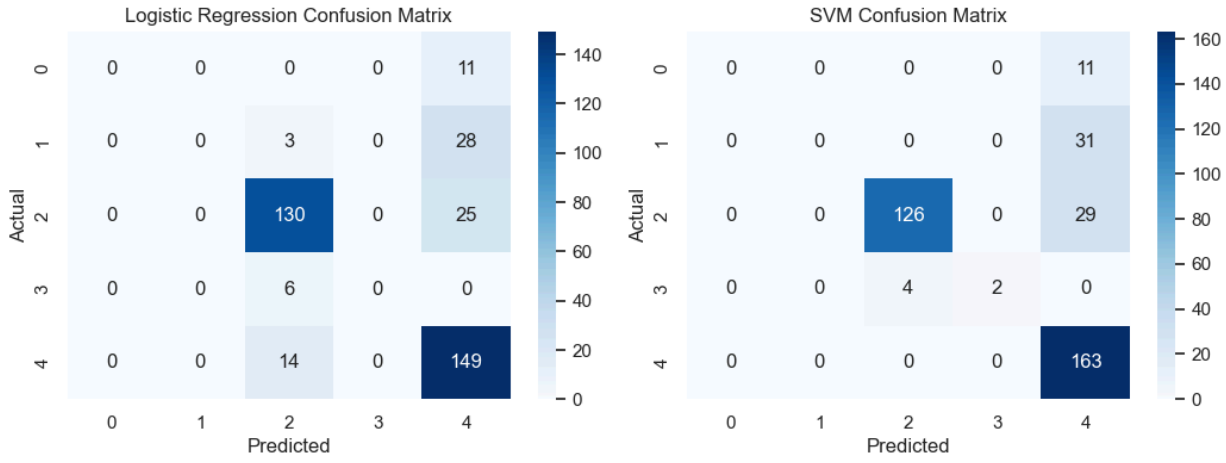
```
from sklearn.linear_model import LogisticRegression
logistic_classifier = LogisticRegression(random_state=0)
logistic_preds = evaluate_model(logistic_classifier, "Logistic Regression")
```

The values present in the diagonal of the confusion matrix (0,4,144,1,142) represent the number of data points which have been correctly classified by the model. From the accuracy score, it is evident that the model has an accuracy of **79.5%**.

SVM

```
from sklearn.svm import SVC
svm_classifier = SVC(kernel='linear', random_state=0)
svm_preds = evaluate_model(svm_classifier, "SVM")
```

The Support Vector Classifier provides an accuracy of **79.5** % on the dataset.



Naive Bayes

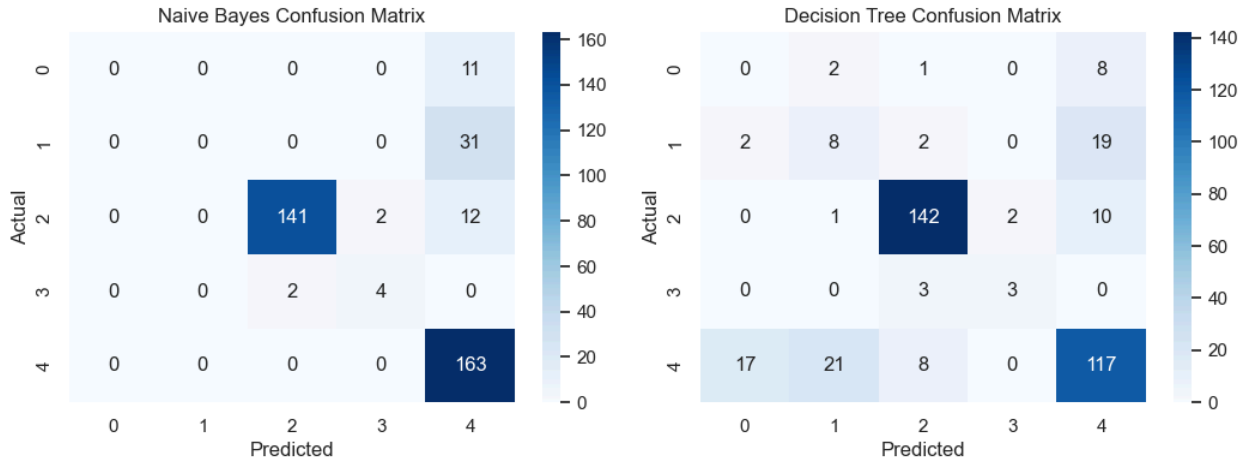
```
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_preds = evaluate_model(nb_classifier, "Naive Bayes")
```

The Naive Bayes Classifier has an accuracy of **84.15%**

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier()
dt_preds = evaluate_model(dt_classifier, "Decision Tree")
```

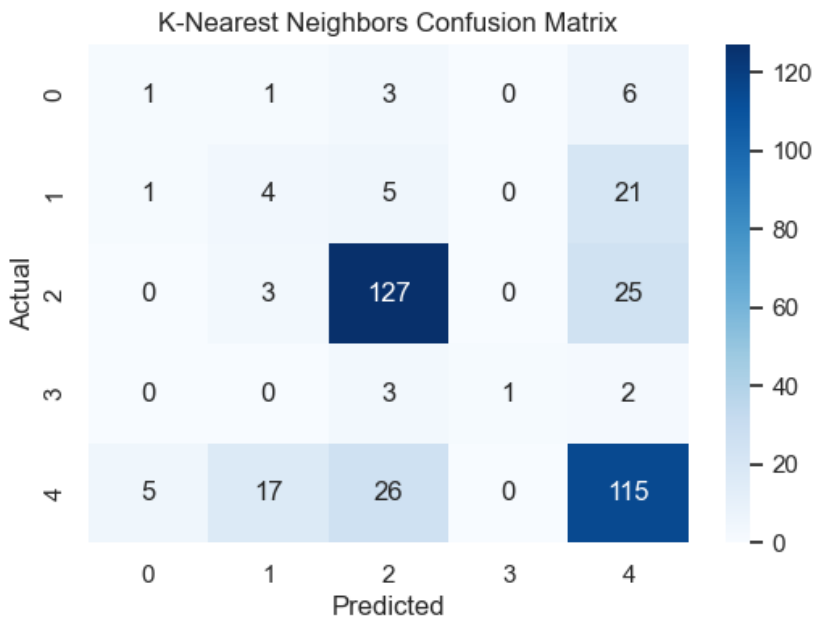
The Decision Tree has an accuracy of **73.77%**



K-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier()
knn_preds = evaluate_model(knn_classifier, "K-Nearest Neighbors")
```

The K-Nearest Neighbors has an accuracy of 67,6%



4. Conclusion

In summary, this project underscores the transformative role of machine learning in refining weather forecasting, thereby bolstering decision-making across diverse sectors like agriculture, transportation, and emergency response. While ongoing research and enhancements remain imperative, this endeavor illuminates the substantial promise of machine learning in elevating weather prediction accuracy and deepening insights into atmospheric dynamics.

5. Resources

Predict The Weather with Machine Learning: Beginner Project:

Predict The Weather with Machine Learning: Beginner Project

Weather-Forecast-And-Prediction-by-Machine-Learning:
https://github.com/yinglung174/Weather-Forecast-And-Prediction-by-Machine-Learning

Dataset: https://www.kaggle.com/datasets/ananthr1/weather-prediction