Name - Gaurang A Raorane Div - D15A Roll no - 49 Batch - C

**Experiment - 6**

**Aim :-** To perform Classification modeling on a dataset.

a. Choose a classifier for classification problems.

b. Evaluate the performance of the classifier.

1. K-Nearest Neighbors (KNN)
2. Naive Bayes
3. Support Vector Machines (SVMs)
4. Decision Tree

**K-Nearest Neighbors (KNN):**

* + Theory: KNN is a simple and effective algorithm that works based on the distance between data points. It classifies a data point by a majority vote of its neighbors, with the data point being assigned to the class most common among its k nearest neighbors.
  + Selection Considerations: KNN is suitable for datasets with non-linear decision boundaries and can handle both binary and multi-class classification problems. However, it might not perform well on datasets with high dimensionality or imbalanced classes.

**Naive Bayes:**

* + Theory: Naive Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of independence between features. It calculates the probability of each class given a set of features and selects the class with the highest probability.
  + Selection Considerations: Naive Bayes is computationally efficient, simple, and works well with high-dimensional data. It assumes independence between features, which might not hold true in some cases.

**Support Vector Machines (SVMs):**

* + Theory: SVMs aim to find the hyperplane that best separates the classes in the feature space. It works by maximizing the margin between the classes, thus making it robust to outliers and effective in high-dimensional spaces.
  + Selection Considerations: SVMs are versatile and suitable for both linear and non-linear classification tasks. They perform well in scenarios with clear margin of separation and can handle datasets with high dimensionality. However, they might be sensitive to the choice of kernel and require careful parameter tuning.

**Decision Tree:**

* + Theory: Decision trees are hierarchical structures that recursively split the data based on the features, aiming to minimize impurity in the resulting subsets. They make decisions by following the paths from the root to the leaf nodes based on the feature values.
  + Selection Considerations: Decision trees are intuitive, easy to interpret, and can handle both numerical and categorical data. They are prone to overfitting, especially on complex datasets, and may not generalize well to unseen data without proper regularization techniques.

**Evaluation Metrics :-**

* **Accuracy:** Measures the proportion of correctly classified instances out of the total instances.
* **Precision:** Measures the proportion of true positive instances out of all instances predicted as positive.
* **Recall (Sensitivity):** Measures the proportion of true positive instances that were correctly classified.
* **F1 Score:** Harmonic mean of precision and recall, provides a balance between precision and recall.
* **Confusion Matrix:** Provides a detailed breakdown of the true positive, true negative, false positive, and false negative predictions.
* **Receiver Operating Characteristic (ROC) Curve:** Graphical representation of the true positive rate against the false positive rate at various threshold settings.
* **Area Under the ROC Curve (AUC-ROC):** Quantifies the overall performance of the classifier across all threshold settings.

**Input:-**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

# Load the dataset

# Data Preparation

# Define features and target variable

X = df[['Latitude', 'Longitude', 'Precip', 'Pressure', 'Humidity\_2m', 'Temp\_2m']] # Features

y = df['District'] # Target variable (assuming 'District' is the classification label)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Choose Classifiers

classifiers = {

"K-Nearest Neighbors (KNN)": KNeighborsClassifier(),

"Naive Bayes": GaussianNB(),

"Support Vector Machines (SVMs)": SVC(),

"Decision Tree": DecisionTreeClassifier()

}

# Evaluate Performance

for name, classifier in classifiers.items():

# Train the classifier

classifier.fit(X\_train, y\_train)

# Predictions

y\_pred = classifier.predict(X\_test)

# Evaluation Metrics

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted')

recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

# Print results

print(f"Classifier: {name}")

print(f"Accuracy: {accuracy}")

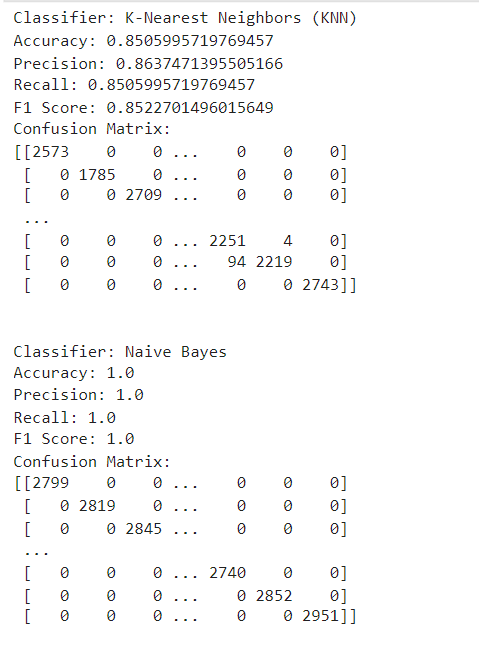
print(f"Precision: {precision}")

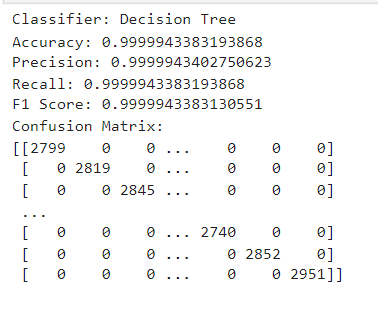
print(f"Recall: {recall}")

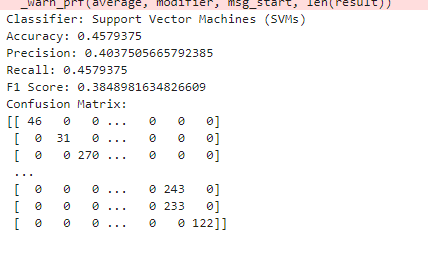
print(f"F1 Score: {f1}")

print(f"Confusion Matrix:\n{conf\_matrix}")

print("\n")

****

****



**Conclusion:-** Performed classical modeling like KNN, Naive Bayes, Random Forest, Support Vector Machines on the climate dataset.