220801140

**EX. NO:**

**DATE :**

10

11/05/2024

# K-NN classifier to predict signal quality based on distance from the transmitter, signal strength, and frequency

**AIM:**

To classify signal quality based on various parameters such as distance from the transmitter, signal strength, and frequency.

**PROGRAM:**

import numpy as np import pandas as pd import matplotlib.pyplot as plt from sklearn.model\_selection import train\_test\_split from sklearn.neighbors import KNeighborsClassifier from sklearn.preprocessing import StandardScaler from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix import seaborn as sns

# Example dataset: Distance (meters), Signal Strength (dBm), Frequency (MHz) vs. Signal Quality data = {

'Distance': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4, 5, 6],

'Signal\_Strength': [-30, -35, -40, -45, -50, -55, -60, -65, -70, -75, -33, -38, -43, -48, -53],

'Frequency': [850, 850, 850, 850, 850, 1900, 1900, 1900, 1900, 1900, 850, 850, 1900, 1900, 1900],

'Signal\_Quality': ['Good', 'Good', 'Good', 'Good', 'Bad', 'Bad', 'Bad', 'Bad', 'Bad', 'Bad', 'Good', 'Good', 'Bad', 'Bad', 'Bad'] }

# Convert the data into a DataFrame df = pd.DataFrame(data)

# Separate features and target variable

X = df[['Distance', 'Signal\_Strength', 'Frequency']].values # Features y = df['Signal\_Quality'].values # Target

# Encode the target variable

from sklearn.preprocessing import LabelEncoder le = LabelEncoder() y = le.fit\_transform(y) # 'Good' -> 1, 'Bad' -> 0

220801140

# 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)

# Standardize the features scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create and train the k-NN classifier k = 3 # Number of neighbors model = KNeighborsClassifier(n\_neighbors=k) model.fit(X\_train, y\_train)

# Make predictions y\_pred = model.predict(X\_test)

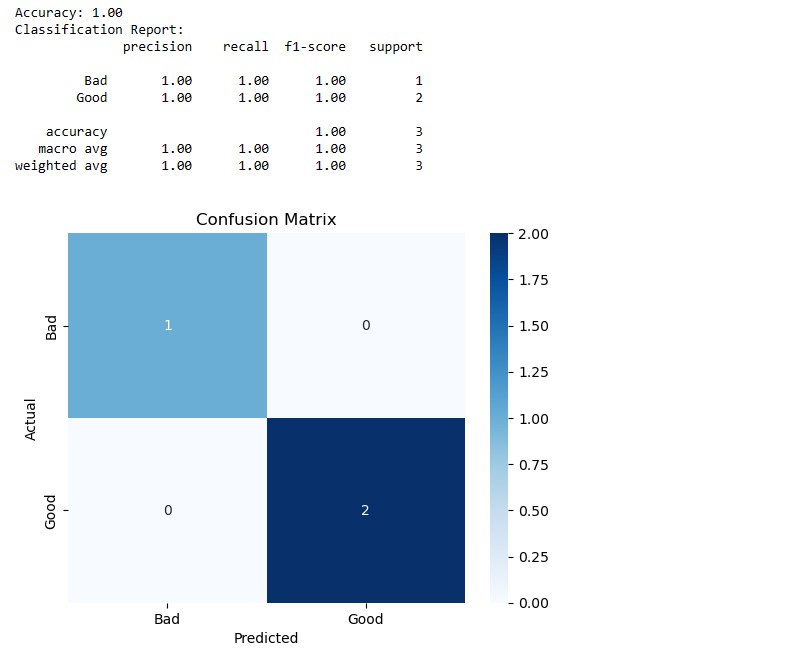
# Evaluate the model accuracy = accuracy\_score(y\_test, y\_pred) report = classification\_report(y\_test, y\_pred, target\_names=['Bad', 'Good'])

print(f'Accuracy: {accuracy:.2f}') print('Classification Report:') print(report)

# Confusion Matrix conf\_matrix = confusion\_matrix(y\_test, y\_pred) sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Bad', 'Good'], yticklabels=['Bad', 'Good']) plt.xlabel('Predicted') plt.ylabel('Actual') plt.title('Confusion Matrix') plt.show()

220801140

**OUTPUT :**



**RESULT:**

Hence the signal quality is classified based on various parameters using K-NN classifier.