LAB # 3

K-NEAREST NEIGHBOR (KNN) ALGORITHM

Lab Tasks:-

Weather	Temperature	Play
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	No
Sunny	Cool	Yes
Rainy	Mild	Yes
Sunny	Mild	Yes
Overcast	Mild	Yes
Overcast	Hot	Yes
Rainy	Mild	No

1. Implement K-Nearest Neighbor (KNN) Algorithm on the above dataset in Fig 1 to predict whether the players can play or not when the weather is overcast and the temperature is mild.Also apply confusion Matrix.

Code:-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

data = {
    'Weather': ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast', 'Sunny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy'],
    'Temperature': ['Hot','Hot','Hot','Mild','Cool','Cool','Cool', 'Mild','Cool','Mild','Mild','Mild','Hot','Mild'],
    'Play': ['No','No','Yes','Yes','Yes','No','Yes', 'No','Yes','Yes','Yes','Yes','Yes','No']
}
df = pd.DataFrame(data)

le_weather = LabelEncoder()
le_temp = LabelEncoder()
le_play = LabelEncoder()

df['Weather'] = le_weather.fit_transform(df['Weather'])
df['Temperature'] = le_temp.fit_transform(df['Temperature'])
df['Play'] = le_play.fit_transform(df['Play'])

X = df[['Weather', 'Temperature']]
y = df['Play']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)

print("Confusion Matrix:\n", cm)
print("Accuracy:", round(acc, 2))

weather_val = le_weather.transform(['Overcast'])[0]
temp_val = le_temp.transform(['Mild'])[0]

sample = pd.DataFrame([[weather_val, temp_val]], columns=['Weather', 'Temperature'])

prediction = knn.predict(sample)
```

Output:-

```
Confusion Matrix:
 [[1 1]
 [1 2]]
Accuracy: 0.6
Prediction for (Overcast, Mild): Yes
```

2. Here are 4 training samples. The two attributes are acid durability and strength. Now the factory produces a new tissue paper that passes laboratory test with X1=3 and X2=7. Predict the classification of this new tissue.

X1= Acid durability (sec)	X2=Strength (kg/m^2)	Y=Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Code:-

```python
import math
from collections import Counter

data = [
    [7, 7, 'Bad'],
    [7, 4, 'Bad'],
    [3, 4, 'Good'],
    [1, 4, 'Good']
]

query = [3, 7]
k = 3

distances = []
for sample in data:
    x1, x2, label = sample
    distance = math.sqrt((x1 - query[0])**2 + (x2 - query[1])**2)
    distances.append((distance, label))

print("Distances from query (3,7):")
for d, label in distances:
    print(f"{label:>4} -> {d:.3f}")

distances.sort(key=lambda x: x[0])

neighbors = distances[:k]
print("\nK Nearest Neighbors (k=3):")
for d, label in neighbors:
    print(f"{label:>4} -> {d:.3f}")

neighbor_labels = [label for _, label in neighbors]
prediction = Counter(neighbor_labels).most_common(1)[0][0]

print(f"\nPredicted class for query instance (3,7): {prediction}")
```

Output:-

```
Distances from query (3,7):
 Bad -> 4.000
 Bad -> 5.000
Good -> 3.000
Good -> 3.606

K Nearest Neighbors (k=3):
Good -> 3.000
Good -> 3.606
 Bad -> 4.000

Predicted class for query instance (3,7): Good
```