HW_03_KNN

Poorvi Raut -20009560

Sample code number id number

F1. Clump Thickness 1 - 10 F2.

Uniformity of Cell Size 1 - 10 F3.

Uniformity of Cell Shape 1 - 10 F4.

Marginal Adhesion 1 - 10 F5.

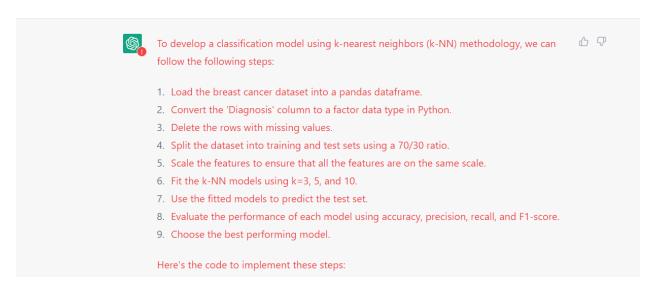
Single Epithelial Cell Size 1 - 10 F6.

Bare Nuclei 1 - 10 F7.

Bland Chromatin 1 - 10 F8.

Normal Nucleoli 1 - 10 F9.

Mitoses 1 - 10 Diagnosis Class: (2 for benign, 4 for malignant)



```
Copy code
python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Load the dataset into a pandas dataframe
df = pd.read_csv('breast_cancer_dataset.csv')
# Convert the 'Diagnosis' column to a factor data type
df['Diagnosis'] = pd.factorize(df['Diagnosis'])[0]
# Delete rows with missing values
df.dropna(inplace=True)
# Split the dataset into training and test sets
X = df.drop(['Sample code number', 'Diagnosis'], axis=1)
y = df['Diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

```
# scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Fit the k-NN models
k_values = [3, 5, 10]
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

# Use the fitted model to predict the test set
    y_pred
```

After running the code on jupyter lab:

```
import pandas as pd
from sklearn.model selection import train test split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy score, confusion matrix
# Assuming your dataset is a CSV file named "breast cancer data.csv"
data = pd.read csv("breast-cancer-wisconsin.csv")
# Convert all columns (except 'Sample' and 'Class') to numeric
columns to convert = [col for col in data.columns if col not in ['Sample',
 'Class']]
data[columns to convert] = data[columns to convert].apply(pd.to numeric, e
rrors='coerce')
# Remove rows with missing values
data = data.dropna()
# Split the dataset into training (70%) and testing (30%) sets
train data, test data = train test split(data, test size=0.3, random state
=123)
```

```
# Remove the Sample code number (id) from the dataset
train data = train data.drop(columns=['Sample'])
test data = test data.drop(columns=['Sample'])
train labels = train data.pop('Class')
test labels = test data.pop('Class')
k \text{ values} = [3, 5, 10]
for k in k values:
    knn = KNeighborsClassifier(n neighbors=k)
    knn.fit(train data, train labels)
    predicted class = knn.predict(test data)
    accuracy = accuracy score(test labels, predicted class)
    confusion = confusion matrix(test labels, predicted class)
We get output as:
Accuracy for k = 3: 0.9853658536585366
Confusion Matrix:
[[131 1]
[ 2 71]]
Accuracy for k = 5: 0.9853658536585366
Confusion Matrix:
[[130
      2]
[ 1 72]]
Accuracy for k = 10: 0.9804878048780488
Confusion Matrix:
[[131 1]
[ 3 70]]
```