HW_03_KNN

Poorvi Raut -20009560

The “breast cancer dataset” in CANVAS was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The features in the dataset, described below, have been categorized from 1 to 10. Use the knn methodology (k=3,5 and 10) to develop a classification model for the Diagnosis. Important: make sure your categories are represented by the “factor” data type in python and delete the rows with missing value. Use 30% test 70% training data.
Features Domain -- ----------

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)

Below is screenshot of code in Python with necessary steps generated using ChatGpt.Ai

To develop a classification model using k-nearest neighbors (k-NN) methodology, we can follow the following steps:

1. Load the breast cancer dataset into a pandas dataframe.
2. Convert the 'Diagnosis' column to a factor data type in Python.
3. Delete the rows with missing values.
4. Split the dataset into training and test sets using a 70/30 ratio.
5. Scale the features to ensure that all the features are on the same scale.
6. Fit the k-NN models using k=3, 5, and 10.
7. Use the fitted models to predict the test set.
8. Evaluate the performance of each model using accuracy, precision, recall, and F1-score.
9. Choose the best performing model.

Here's the code to implement these steps:

python 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_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]]
```