CPSC 552 – Assignment #4

KNN Algorithm

KNN algorithm is a very easy algorithm that can be used either for classification or for regression. It has a parameter k that indicates the number of neighbors to examine in order to make a decision. In this assignment, you will explore KNN on two different datasets.

Problem #1: Program and test the accuracy of the KNN algorithm on the Abalone dataset which has 8 features. Do a regression on the age of the abalone i.e., predict the age of the abalone (indicated by number of rings). The features in the dataset are:

```
abalone.columns = [
        "Sex",
        "Length",
        "Diameter",
        "Height",
        "Whole weight",
        "Shucked weight",
        "Viscera weight",
        "Shell weight",
        "Rings",
    ]
```

You will experiment with the number of neighbors (k) that determines the highest accuracy on the test set. We will divide the dataset into 80% training and 20% test.

Solution: Create a project called AbaloneKNN. Add a file called Utils.py with the following code in it. This file has simple functions to read the data, prepare the train and test sets, and plot the results.

```
#-----------Utils.py-------------
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from numpy import *

def get_dataset():
    url = (
        "https://archive.ics.uci.edu/ml/machine-learning-databases"
        "/abalone/abalone.data"
    )
    abalone = pd.read_csv(url, header=None)
    abalone.columns = [
        "Sex",
        "Length",
        "Diameter",
        "Height",
        "Whole weight",
        "Shucked weight",
        "Viscera weight",
        "Shell weight",
        "Rings",
    ]
    return abalone
```

```
def get_train_test_data(X, y):
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=12345)
    return X_train, X_test, y_train, y_test

def plot_predicted_vs_actual(ypred, y):
    mean_error = sum(abs(ypred-y))/len(y)
    step_size = 20
    a = [ypred[i] for i in range(0,len(ypred)) if i%step_size==0]
    b = [y[i] for i in range(0,len(ypred)) if i%step_size==0]
    t = linspace(0, len(a), len(a))

    plt.plot(t, a, 'red',linestyle='dashed', label='predicted')
    plt.plot(t, b, 'blue',label='actual')
    plt.scatter(t, a,marker='o', s=10, color="red", label="predicted")
    plt.scatter(t, b, s=10, color="blue", label="actual")
    plt.legend()
    plt.title('mean error ='+str(mean_error)) #title
    plt.show()
```

The code in the main AbaloneKNN.py appears as:

```
import sys
import Utils
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from math import sqrt

def main():
    df = Utils.get_dataset()
    print(df.head)

    df = df.drop("Sex", axis=1) # this column may not contribute to age prediction

    df["Rings"].hist(bins=15) # plot distribution of data for age i.e., rings
    plt.show()

    # see if the different features are correlated to age
    # positive or negative correlation between the features
    # and the target helps in better prediction
    correlation_matrix = df.corr()
    print(correlation_matrix["Rings"]) # correlation of each
    # column with the rings column

    X = df.drop("Rings", axis=1) # Rings is our target prediction
    X = X.values # convert to numpy
    print(X.shape)
    y = df["Rings"]
    y = y.values # convert to numpy array

    #------test KNN on an unknown data point----------
```

```
    new_data_point = np.array([
        0.417, # length
        0.396, # diameter
        0.134, # height
        0.816, # whole weight
        0.383, # shucked weight
        0.172, # viscera weight
        0.221, # shell weight
    ])

    distances = np.linalg.norm(X - new_data_point, axis=1)
    k = 3 # number of meighbors to examine
    nearest_neighbor_ids = distances.argsort()[:k] # top k neighbors' indices
    print(nearest_neighbor_ids)
    nearest_neighbor_rings = y[nearest_neighbor_ids]
    print(nearest_neighbor_rings)
    prediction = nearest_neighbor_rings.mean()
    print('pedicted number of rings =', prediction)

    # rather than taking the mean, an alternative is to
    # take the mode (most commonly occuring value)

    # mode based result on the abalone dataset
    k = 7
    nearest_neighbor_ids = distances.argsort()[:k]
    print(nearest_neighbor_ids)
    nearest_neighbor_rings = y[nearest_neighbor_ids]
    print(nearest_neighbor_rings)
    prediction = nearest_neighbor_rings.mean()
    print('pedicted number of rings (k=7) =', prediction)
    mode = scipy.stats.mode(nearest_neighbor_rings)
    print('Predicted rings, using mode with k = 7, # rings=',mode[0])

    #--------------using knn in sklearn-------
    # get train, test data using scikit's train_test_split
    X_train, X_test, y_train, y_test = Utils.get_train_test_data(X,y)
    knn_model = KNeighborsRegressor(n_neighbors=5)
    knn_model.fit(X_train, y_train)
    train_preds = knn_model.predict(X_train)
    mse = mean_squared_error(y_train, train_preds)
    rmse = sqrt(mse)
    print('training error = ',rmse)

    test_preds = knn_model.predict(X_test)
    mse = mean_squared_error(y_test, test_preds)
    rmse = sqrt(mse)
    print('testing error = ',rmse)
    Utils.plot_predicted_vs_actual(test_preds,y_test)
    # try different values of n_neighbors to see if error drops

if __name__ == "__main__":
    sys.exit(int(main() or 0))
```

Problem #2: Test the KNN algorithm for predicting the cancer type on the cancer dataset. Two files will be provided to you. One contains the gene expression data for 20531 genes for 801 patients. The other

file contains the class labels for the five types of cancer. Here is the code to read the data and convert the five categories of cancer into numeric values. The datafiles are available as a zip file on the course web site.

```
datafile = "D:/PythonAM2/Data/TCGA-PANCAN-HiSeq-801x20531/data.csv"
labels_file = "D:/PythonAM2/Data/TCGA-PANCAN-HiSeq-801x20531/labels.csv"

data = np.genfromtxt(
    datafile,
    delimiter=",",
    usecols=range(1, 20532),
    skip_header=1
)

true_label_names = np.genfromtxt(
    labels_file,
    delimiter=",",
    usecols=(1,),
    skip_header=1,
    dtype="str"
)
print(data.shape)

print(true_label_names[:5])
# The data variable contains all the gene expression values
#  from 20,531 genes. The true_label_names are the cancer
#  types for each of the 801 samples.
#  BRCA: Breast invasive carcinoma
#  COAD: Colon adenocarcinoma
#  KIRC: Kidney renal clear cell carcinoma
#  LUAD: Lung adenocarcinoma
#  PRAD: Prostate adenocarcinoma
```

Use 80% of the data for training and report the accuracy of detecting the correct cancer type using the KNN algorithm on the remaining 20% of the data.