Assignment 1- K-Nearest Neighbor Algorithm implementation

By Neeraj Sharma

Design:

Following functions are created for this assignment

- **Functions for KNN Algorithm**

```
#function is to load the test data
def load_dataset(filename, split, train_dataset=[],
test_dataset=[]):
```

```
#calculate Euclidean distance between each row of test data and training data
def calculate_euclidean_distance(instance1, instance2, length):
```

```
# finding the neighbors of the test_instance after sorting them by distance
def fetch_neighbors(train_dataset, test_instance, k):
```

```
# Calculate the voting for a perticular result and fetch the predicted and actual value
def fetch_response(neighbors_list):
```

```
# Calculating accuracy of the KNN program
def fetch_accuracy(test_dataset, knn_predictions):
```

```
# Main program defined here, it will take input k from GUI and
def knnmain(k):
```

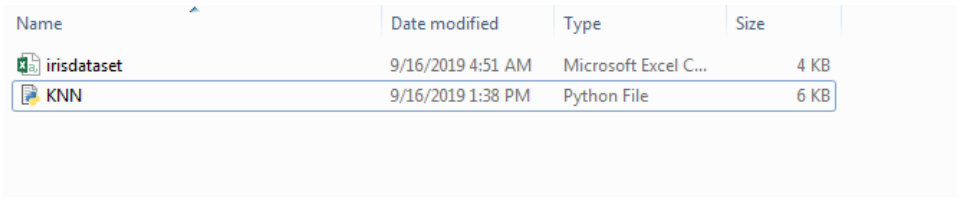
- **Functions for GUI part**

```
#This function is used to fetch value entered in the input box and it is passed to the knnmain function
def fetch(entries):
```

```
#This function creates the form
def makeform(root, fields):
```

Instruction to run the program:

1. Copy the file KNN.py in your local folder



2. Copy the data file irisdataset.csv in the same folder and provide complete file path in the knnmain function

```
#call the function load dataset and provide above information including file name.
load_dataset(r'C:\Neeraj Sharma\Kennesaw State\Fall 2019\Machine Learning\KNN Project\iri
```

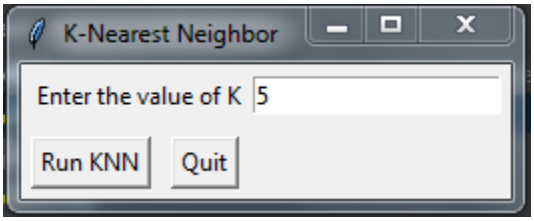
3. Open Power shell and navigate to the folder where we have the program

```
PS C:\Neeraj Sharma\Kennesaw State\Fall 2019\Machine Learning> cd '.\KNN Assignment'
PS C:\Neeraj Sharma\Kennesaw State\Fall 2019\Machine Learning\KNN Assignment> ls


    Directory: C:\Neeraj Sharma\Kennesaw State\Fall 2019\Machine Learning\KNN Assignment


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         9/16/2019   4:51 AM       3861 irisdataset.csv
-a---         9/16/2019   1:38 PM       5488 KNN.py
```

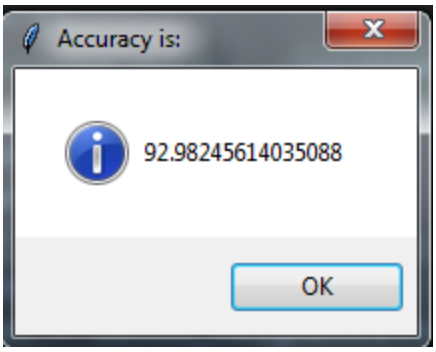
4. Run the comman python

```
PS C:\Neeraj Sharma\Kennesaw State\Fall 2019\Machine Learning\KNN Assignment> python .\KNN.py
```

5. It will open the input window, Enter the value and click Run KNN.



6. It will open the Accuracy Shows in another window:



`

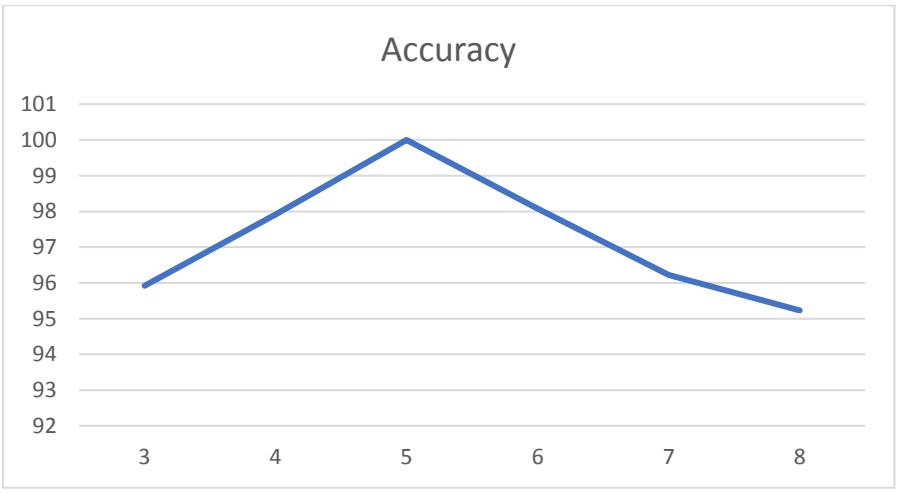
We can close the Accuracy window and again use the KNN window to enter different K value and run the program again.

Quit Button – It exits from the program.

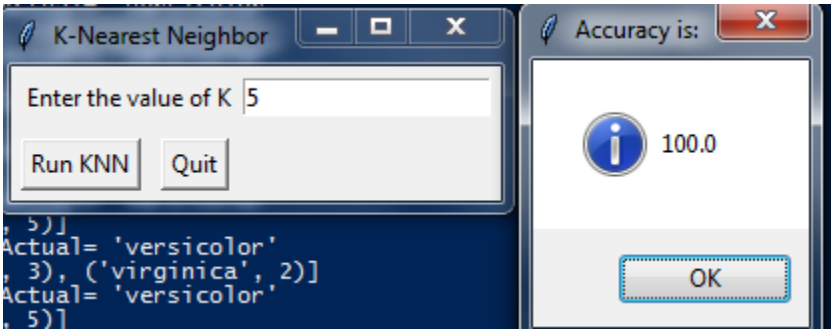
If user will not input anything and click on Run-KNN then also program will exit.

Comparison chart of different K Value and Accuracy:

Comparison Chart	
K	**Accuracy**
3	95.918
4	97.916
5	100
6	98.076
7	96.226
8	95.23



Screen shot of highest accuracy result



Program Code KNN.py

```
import random
import csv
import math
import operator
import tkinter.messagebox
from tkinter import *

#load data set and split it to training set and data set
def load_dataset(filename, split, train_dataset=[],
test_dataset=[]):
    with open(filename, 'r') as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        print(f"Total Data: {len(dataset)-1}")
        for x in range(len(dataset)-1):
            for y in range(4):
                dataset[x+1][y] = dataset[x+1][y]
            if random.random() < split:    #Return the next
random floating point number in the range [0.0, 1.0).
                train_dataset.append(dataset[x+1])
            else:
                test_dataset.append(dataset[x+1])


# find Euclidean distance between 2 data points
def calculate_euclidean_distance(instance1, instance2, length):
    dist_btw_points = 0
    for x in range(length):
        dist_btw_points += pow((float(instance1[x]) -
float(instance2[x])), 2)
    return math.sqrt(dist_btw_points)
```

```
# finding the neighbors of the test_instance after sorting them
by distance
def fetch_neighbors(train_dataset, test_instance, k):

    dist = []
    length = len(test_instance) - 1
    for x in range(len(train_dataset)):
        dist_btw_points =
calculate_euclidean_distance(test_instance, train_dataset[x],
length)
        dist.append((train_dataset[x], dist_btw_points))
    dist.sort(key=operator.itemgetter(1))
    neighbors_list = []
    for x in range(k):
        neighbors_list.append(dist[x][0])
    return neighbors_list


# Calculate the voting for a perticular result and fetch the
predicted and actual value
def fetch_response(neighbors_list):
    class_votes = {}
    for x in range(len(neighbors_list)):
        response = neighbors_list[x][-1]
        if response in class_votes:
            class_votes[response] += 1
        else:
            class_votes[response] = 1
    sorted_votes = sorted(class_votes.items(),
key=operator.itemgetter(1), reverse=True)
    print(f"Sorted votes {sorted_votes}")
    return sorted_votes[0][0]


# Calculating accuracy of the KNN program
def fetch_accuracy(test_dataset, knn_predictions):
    correct_prediction = 0
    for x in range(len(test_dataset)):
        if test_dataset[x][-1] == knn_predictions[x]:
            correct_prediction += 1
    return (correct_prediction / float(len(test_dataset))) *
100.0


# Main program defined here
def knnmain(k):
    # Initializing different datasets
```

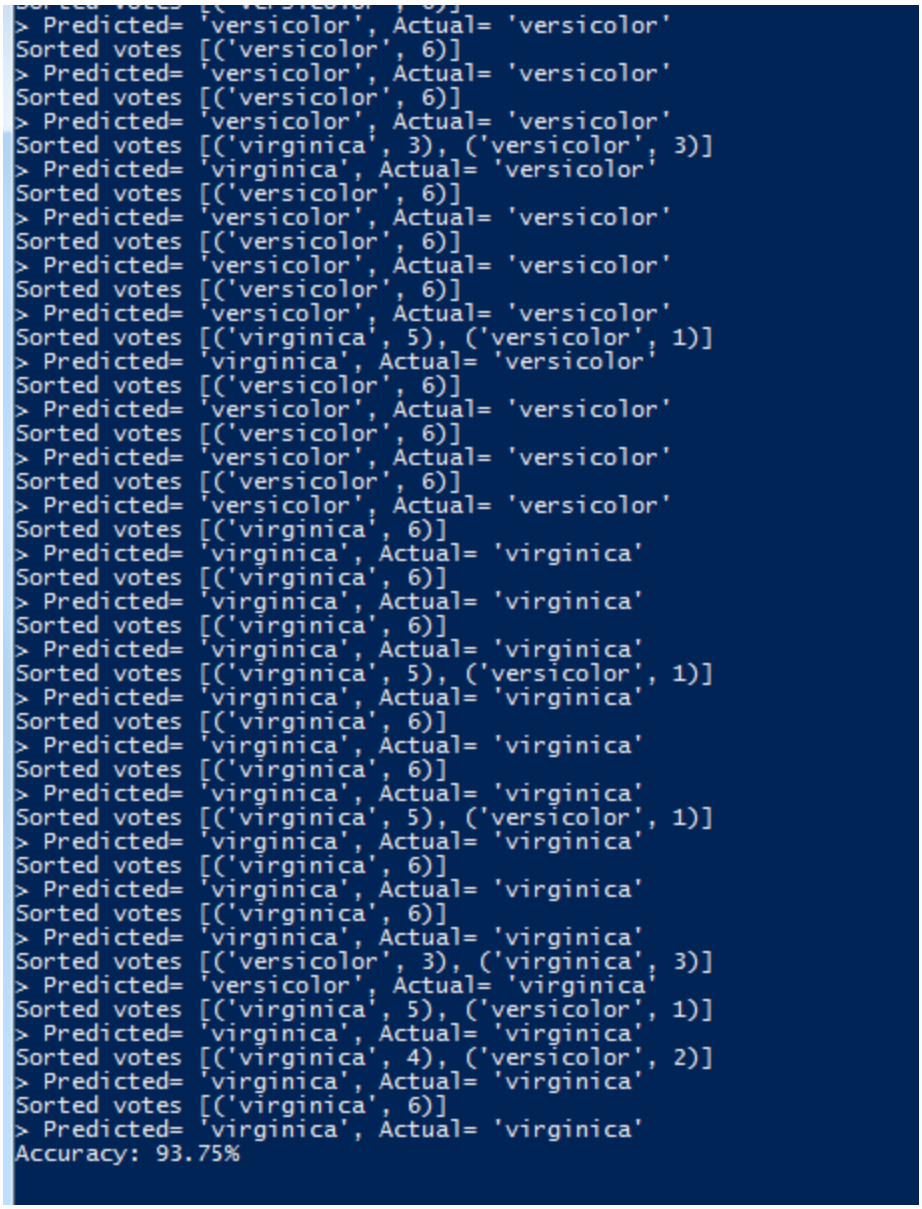
```
    train_dataset = []
    test_dataset = []
    split = 0.67
    #call the function load_dataset and provide above
information including file name.
    load_dataset(r'C:\Neeraj Sharma\Kennesaw State\Fall
2019\Machine Learning\KNN Project\irisdataset.csv', split,
train_dataset, test_dataset)
# generating knn_predictions
    knn_predictions = []
    #display split dataset and total count
    print(f"Total training data : {len(train_dataset)}")
    print(f"Total number of test data: {len(test_dataset)}")
    print(train_dataset)
    print(test_dataset)
   # k = int(input("Please enter the value of k: "))
    for x in range(len(test_dataset)):
        neighbors_list = fetch_neighbors(train_dataset,
test_dataset[x], k)
        result = fetch_response(neighbors_list)
        knn_predictions.append(result)
        print('> Predicted= '+ repr(result)+', Actual=
'+repr(test_dataset[x][-1]))
    knn_accuracy = fetch_accuracy(test_dataset, knn_predictions)
    print('Accuracy: ' + repr(knn_accuracy) + '%')
    tkinter.messagebox.showinfo("Accuracy is: ",knn_accuracy,)
#displaying accuracy in the message box.

#=====================================GUI
Portion=============================================================
====
fields = 'Enter the value of K',

#This function is used to fetch value entered in the input box
and it is passed to the knnmain function
def fetch(entries):
    for entry in entries:
        field = entry[0]
        text  = entry[1].get()
       # print('%s: "%s"' % (field, text))
        my_input_str = text
    #
    # #my_input_str = input('please enter the input string: ')
    if my_input_str == "":
        return root.quit
    else:
        knnmain(int(my_input_str))
```

```
    return

#This function creates the form
def makeform(root, fields):
    entries = []
    for field in fields:
        row = Frame(root)
        lab = Label(row, width=15, text=field, anchor='w')
        ent = Entry(row)
        row.pack(side=TOP, fill=X, padx=5, pady=5)
        lab.pack(side=LEFT)
        ent.pack(side=RIGHT, expand=YES, fill=X)
        entries.append((field, ent))

    return entries

if __name__ == '__main__':
    root = Tk()    #create an object of the TK function to create
the window.
    root.title("K-Nearest Neighbor") #window title
    ents = makeform(root, fields)     #call makeform function to
create the form.
    root.bind('<Return>', (lambda event, e=ents: fetch(e)))
    #Creating button Run KNN in the window
    b1 = Button(root, text='Run KNN',
                command=lambda e=ents: fetch(e))
   #Creatign button Quit in the input window
    b1.pack(side=LEFT, padx=5, pady=5)
    b2 = Button(root, text='Quit', command=root.destroy)
    b2.pack(side=LEFT, padx=5, pady=5)
    #loop the window until user exits from the program
    root.mainloop()
```

Output Console:

It logs total data value, Total training data , Total number of test data and also it show how much vote has been received by predicted value and list of predicted and actual value followed by accuracy value at the end.

```
Accuracy: 96.01010949152543%
PS C:\Neeraj Sharma\Kennesaw State\Fall 2019\Machine Learning\KNN Assignment> python .\KNN.py
Total Data: 150
Total training data : 102
Total number of test data: 48
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('setosa', 6)]
> Predicted= 'setosa', Actual= 'setosa'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 5), ('virginica', 1)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('virginica', 3), ('versicolor', 3)]
> Predicted= 'virginica', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
```


> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('virginica', 3), ('versicolor', 3)]
> Predicted= 'virginica', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('virginica', 5), ('versicolor', 1)]
> Predicted= 'virginica', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('versicolor', 6)]
> Predicted= 'versicolor', Actual= 'versicolor'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 5), ('versicolor', 1)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 5), ('versicolor', 1)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('versicolor', 3), ('virginica', 3)]
> Predicted= 'versicolor', Actual= 'virginica'
Sorted votes [('virginica', 5), ('versicolor', 1)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 4), ('versicolor', 2)]
> Predicted= 'virginica', Actual= 'virginica'
Sorted votes [('virginica', 6)]
> Predicted= 'virginica', Actual= 'virginica'
Accuracy: 93.75%