KNN From Scratch

Group Members:

1. Ruben Chevez
2. Kratika Naskulwar

Code inspired in the following blog post

https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

The complete source code for this report can be found in the following link:

https://colab.research.google.com/drive/193b21WymzI12LnkDd178UDqjcKVb97WO

In this document, we describe our steps taken to implement a KNN algorithm from scratch. The steps are the following:

1. Declare the hyperparameters
    a. K - Number of closest neighbors to be taken for distance calculations.
    b. Training Filename
    c. Test Filename
2. Using the sys library, we obtain the arguments from the command line.
3. Using the panda's library, we read from the TSV file and parse it using \t as the delimiter.

```
import pandas as pd
train_data = pd.read_csv(traning_filename, sep='\t')
test_data  = pd.read_csv(test_filename, sep='\t')
```

4. We declare the function euclideanDistance which takes as input, two instances and the number of columns that represent features. For example: If we have 10 columns but only 9 represent attributes and the last one represent the class label, the input for the function is 9. Euclidean Distance calculation is calculated as the square root of the sum of the squared differences between the two instances.

```
import math
def euclideanDistance(instance1, instance2, length):
  distance = 0
  for x in range(length):
    distance += pow((instance1[x] - instance2[x]), 2)
  return math.sqrt(distance)
```

5. We declare the getNeighbours function that receives as input all the training set, one test set instance and the K constant parameter. The function will go through each training instance calculating the distances, select and return the k number of closest points.

```
import operator
def getNeighbours(trainingInstance, testInstance, k):
  distances = []
  length = len(testInstance)-1
  for x in range(len(trainingInstance)):
    dist = euclideanDistance(testInstance, trainingInstance[x], length)
    distances.append((trainingInstance[x], dist))
  distances.sort(key=operator.itemgetter(1))
  neighbours = []
  for x in range(k):
    neighbours.append(distances[x][0])
  return neighbours
```

6. We declare getPredictions function to find the class which has the majority votes among the nearest neighbors and the conditional probability. It takes neighbors as an only input returned by getNeighbours function.

```
def getPredictions(neighbours):
  class_votes = {}
  for x in range(len(neighbours)):
    prediction = neighbours[x][-1]
    if prediction in class_votes:
      class_votes[prediction] += 1
    else:
      class_votes[prediction] = 1
  majority_votes = sorted(
    class_votes.items(),
    key=operator.itemgetter(1),
    reverse=True
  )
#   print("Majority vote table: \n\n {} \n".format(pd.DataFrame(majority_votes
  return majority_votes[0][0]  , majority_votes[0][1] / len(neighbours) * 100
```

```
       RI     Na    Mg    Al     Si     K    Ca   Ba    Fe  Class
0  1.51789  13.19  3.90  1.30  72.33  0.55  8.44  0.0  0.28    2.0
1  1.51844  13.25  3.76  1.32  72.40  0.58  8.42  0.0  0.00    2.0
2  1.51829  13.24  3.90  1.41  72.33  0.55  8.31  0.0  0.10    2.0
3  1.51846  13.41  3.89  1.33  72.38  0.51  8.28  0.0  0.00    2.0
4  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.00    1.0

Belongs to Class: 2
With a confidence of 80.0
```

7. Predict for each test instance using the getPrediction function inside a for-loop.

```
trainingSet  = train_data.as_matrix(columns=None)
testSet      = test_data.as_matrix(columns=None)

predictions=[]
for x in range(len(testSet)):
  neighbors = getNeighbours(trainingSet, testSet[x], k_distance)
  result = getPredictions(neighbors)
  predictions.append(result)
  print('{}\t{}'.format(
      str(int(result[0])),
      result[1]
  ))

if debug:
  print("\nShape test set:       ",   testSet.shape )
  print("\nShape predictions set: ", len(predictions)  )
```

```
2	80.0
1	40.0
2	80.0
1	60.0
1	80.0
1	60.0
1	60.0
2	60.0
7	100.0
1	80.0
```

8. We tested each function separately to make sure calculations are correct. For more information, check our Jupyter Python Notebook. https://colab.research.google.com/drive/193b21Wymzl12LnkDd178UDqjcKVb97WO