KNN from scratch

Soham Shah(J059)

KNN is a non-prarametric algo which dosent assume anything about the underlying data.

It stores all the avaialble data and classifies some new data based on similarity.

It is also called the lazy-learner algo as it dosen't immediatley learn from the training set rather stores the datasert and performs an action on the dataset at the time of classification

In [90]:

```python
import math
import pandas as pd
import numpy as np
from sklearn import datasets
```

Loading and Pre-processing data

In [91]:

```python
iris= pd.read_csv('iris.csv')
iris= iris.drop(['Id'], axis=1)
iris.head()
```

Out[91]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [92]:

```python
'''
Setosa- 0
Versicolor -1
Virigica- 2
'''
```

Out[92]:

```
'\nSetosa- 0\nVersicolor -1\nVirigica- 2\n'
```

In [93]:

```python
from sklearn import preprocessing
encoder= preprocessing.LabelEncoder()
iris['Species']= encoder.fit_transform(iris['Species'])
```

In [94]:

```python
from sklearn.model_selection import train_test_split
train ,test= train_test_split(iris)
```

In [95]:

```python
train.head()
```

Out[95]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
30	4.8	3.1	1.6	0.2	0
107	7.3	2.9	6.3	1.8	2
56	6.3	3.3	4.7	1.6	1
57	4.9	2.4	3.3	1.0	1
50	7.0	3.2	4.7	1.4	1

In [96]:

```python
train= train.to_numpy()
test= test.to_numpy()
```

In [133…

```python
#calculating the Eucledian distance
def eucledian_dist(r1, r2):
    distance = 0.0
    for i in range(len(r1)- 1):  #last col is output value
        distance += (r1[i] -r2[i])**2
    return math.sqrt(distance)

#Getting the nearest neighbours
def get_neighbour(train, test_row, num_neighbours):
    distances= []
    for train_row in train:
        dist = eucledian_dist(test_row,train_row)
        distances.append((train_row, dist))
    distances.sort(key=dist_sort) #sorting using distances
    neighbours= []
    for i in range(num_neighbours):
        neighbours.append(distances[i][0])
    return neighbours

def dist_sort(tup):
    return tup[1]

def prediction(train, test_row, num_neighbours):
    neighbours= get_neighbour(train, test_row, num_neighbours)
    output= []
    for class_pre in neighbours:
        output.append(class_pre)
    #counting the max output value which will be the result
    pred_class = [i[-1] for i in output]
    return max(pred_class, key= pred_class.count)
```

In [141…

```python
outcome= 0
for i in range(len(test)):
    if test[i][-1] == prediction(test, train[i], 3):
        outcome += 1
print(f'Final accuracy is {outcome/len(test)}')
```

```
Final accuracy is 0.42105263157894735
```

In [ ]: