Weight-knn using Gradient descent
this method works in three steps
1 the chi-square score between each attribute and the class must be defined using the whole dataset
2 based on a weighting criterion, a vector containing the weights of each attribute is create
3 the weights of each attribute are used in the KNN classification task

Weight-knn using Gradient descent

Training data

#	class	x1	y1
1	**1**	1.5	2.4
2	**1**	2.4	2.1
3	**2**	3.2	1.8
4	**2**	4.2	2.8
5	**1**	1.8	0.8
6	**2**	3.1	2.5
7	**2**	2.5	3.2
8	**3**	3.5	4.2
9	**3**	4.1	3.6
10	**3**	3.8	4

Test data

#	class	x1	y1
1	**1**	0.5	2.1
2	**1**	2.3	1.8
3	**2**	3.3	2.1
4	**2**	4.2	2.8
5	**3**	3.6	4.5
6	**3**	4.5	3.8

learning rate (alpha)	0.2
No. of class	3
K	3

random	**weight_k1**	0.02
	weight_k2	0.50
	weight_k3	0.10

Scale data **Value = value / (1+value)**

Training data

#	class	x1	y1
1	**1**	0.60	0.71
2	**1**	0.71	0.68
3	**2**	0.76	0.64
4	**2**	0.81	0.74
5	**1**	0.64	0.44
6	**2**	0.76	0.71
7	**2**	0.71	0.76
8	**3**	0.78	0.81
9	**3**	0.80	0.78
10	**3**	0.79	0.80

Test data

#	class	x1	y1
1	**1**	0.33	0.68
2	**1**	0.70	0.64
3	**2**	0.77	0.68
4	**2**	0.81	0.74
5	**3**	0.78	0.82
6	**3**	0.82	0.79

Epoch #1

 # round 1 **test data**

#	class	x1	y1
1	**1**	0.33	0.68

#	Rank	distance	class
1	1	0.27	1
2	2	0.37	1
3	6	0.43	2
4	9	0.48	2
5	3	0.39	1
6	5	0.42	2
7	4	0.39	2

Rank	Class
1	1
2	1
3	1
4	2
5	2
6	2
7	3

8	7	0.46	3
9	10	0.48	3
10	8	0.47	3

8	3
9	2
10	3

random	**weight_k1**	0.02
	weight_k2	0.50
	weight_k3	0.10

weight_(k1:k3)	**0.62**

applying gradient descent	
w1	0.096
w2	0.576
w3	0.176

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	**actual class**	**error**
class of xq	0.848	1	0.152

learning rate = alpha

in the weight (update weig	
w1	0.1264
w2	0.6064
w3	0.2064

w' = w + (learning rate * error)

 # round 2 **test data**

#	**class**	**x1**	**y1**
2	**1**	**0.70**	**0.64**

#	**Rank**	**distance**	**class**
1	4	0.12	1
2	1	0.04	1
3	2	0.06	2
4	6	0.15	2
5	10	0.21	1
6	3	0.09	2
7	5	0.12	2
8	9	0.18	3
9	7	0.18	3
10	8	0.18	3

Rank	**Class**
1	1
2	2
3	2
4	1
5	2
6	2
7	3
8	3
9	3
10	1

use previous	**weight_k1**	0.126
	weight_k2	0.606
	weight_k3	0.206

weight_(k1:k3)	**1.752**

applying gradient descent	
w1	-0.024
w2	0.456
w3	0.056

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	**actual class**	**error**
class of xq	1.000	1	0.000

learning rate = alpha

in the weight (update weig	
w1	0.024
w2	0.456
w3	0.056

w' = w + (learning rate * error)

# round 3	test data			
	#	class	x1	y1
	3	2	0.77	0.68

#	Rank	distance	class
1	9	0.17	1
2	3	0.06	1
3	1	0.04	2
4	4	0.07	2
5	10	0.26	1
6	2	0.04	2
7	5	0.10	2
8	8	0.13	3
9	6	0.11	3
10	7	0.12	3

Rank	Class
1	2
2	2
3	1
4	2
5	2
6	3
7	3
8	3
9	1
10	1

use previous	weight_k1	0.024
	weight_k2	0.456
	weight_k3	0.056

weight_(k1:k3)	1.016

applying gradient descent	
w1	0.021
w2	0.453
w3	0.053

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	actual class	error
class of xq	1.000	2	1.000

learning rate = alpha

in the weight (update weigl	
w1	0.2208
w2	0.6528
w3	0.2528

w’ = w + (learning rate * error)

# round 4	test data			
	#	class	x1	y1
	4	2	0.81	0.74

#	Rank	distance	class
1	9	0.21	1
2	8	0.12	1
3	7	0.10	2
4	1	0.00	2
5	10	0.34	1
6	3	0.06	2
7	6	0.10	2
8	5	0.08	3
9	2	0.05	3
10	4	0.07	3

Rank	Class
1	2
2	3
3	2
4	3
5	3
6	2
7	2
8	1
9	1
10	1

use previous	weight_k1	0.221
	weight_k2	0.653
	weight_k3	0.253

weight_(k1:k3)	2.906

applying gradient descent	
w1	0.160
w2	0.272
w3	0.128

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	actual class	error
class of xq	1.392	2	0.608

learning rate = alpha

in the weight (update weig	
w1	0.282
w2	0.393
w3	0.250

w’ = w + (learning rate * error)

 # round 5 **test data**

#	class	x1	y1
5	**3**	**0.78**	**0.82**

#	Rank	distance	class
1	9	0.21	1
2	7	0.16	1
3	8	0.18	2
4	4	0.09	2
5	10	0.40	1
6	6	0.11	2
7	5	0.09	2
8	1	0.01	3
9	3	0.04	3
10	2	0.02	3

Rank	Class
1	3
2	3
3	3
4	2
5	2
6	2
7	1
8	2
9	1
10	1

use previous	**weight_k1**	0.282
	weight_k2	0.393
	weight_k3	0.250

weight_(k1:k3)	**2.775**

applying gradient descent	
w1	0.073
w2	0.038
w3	0.105

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	actual class	error
class of xq	0.649	3	2.351

learning rate = alpha

in the weight (update weig	
w1	0.5432
w2	0.5084
w3	0.5752

w’ = w + (learning rate * error)

 # round 6 **test data**

#	class	x1	y1
6	**3**	**0.82**	**0.79**

#	Rank	distance	class
1	7	0.81	1
2	3	0.74	1

Rank	Class
1	1
2	2

3	2	0.69	2
4	5	0.76	2
5	1	0.57	1
6	4	0.75	2
7	8	0.81	2
8	10	0.84	3
9	6	0.81	3
10	9	0.83	3

3	1
4	2
5	2
6	3
7	1
8	2
9	3
10	3

use previous	**weight_k1**	0.543
	weight_k2	0.508
	weight_k3	0.575

weight_(k1:k3)	**2.135**

applying gradient descent	
w1	0.316
w2	0.281
w3	0.348

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	**actual class**	**error**
class of xq	1.227	3	1.773

learning rate = alpha

in the weight (update weig	
w1	0.6708
w2	0.6359
w3	0.7028

w' = w + (learning rate * error)

Mean Square Error (MSE)

round	**predict class**	**actual class**	**error**	**quare Erro**
1	0.85	1.00	-0.15	0.023
2	1.00	1.00	0.00	0.000
3	1.00	2.00	-1.00	1.000
4	1.39	2.00	-0.61	0.369
5	0.65	3.00	-2.35	5.525
6	1.23	3.00	-1.77	3.143
			MSE	**1.677**

What does the Mean Squared Error Tell You?

The smaller the means squared error, the closer you are to finding the line of best fit. Depending on your data, it may be impossible to get a very small value for the mean squared error.

Mean Absolute Error (MAE)

round	**predict class**	**actual class**	**error**	**bsolute err**
1	0.85	1.00	-0.15	0.152
2	1.00	1.00	0.00	0.000
3	1.00	2.00	-1.00	1.000
4	1.39	2.00	-0.61	0.608
5	0.65	3.00	-2.35	2.351
6	1.23	3.00	-1.77	1.773
			sum	**5.883255**
			MAE	**1.176651**

n
1
0
1
1
1

What is Absolute Error?

Absolute Error is the amount of error in your measurements. It is the difference between the measured value and "true" value. For example, if a scale states 90 pounds but you know your true weight is 89 pounds, then the scale has an absolute error of 90 lbs – 89 lbs = 1 lbs.

The Mean Absolute Error(MAE) is the average of all absolute errors. The formula is:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

Where:

n = the number of errors,

Σ = summation symbol (which means "add them all up"),

|xi – x| = the absolute errors.

	1
sum	**5**

Epoch #2

 # round 1 **test data**

#	class	x1	y1
1	1	0.33	0.68

#	Rank	distance	class
1	7	0.81	1
2	3	0.74	1
3	2	0.69	2
4	5	0.76	2
5	1	0.57	1
6	4	0.75	2
7	8	0.81	2
8	10	0.84	3
9	6	0.81	3
10	9	0.83	3

Rank	Class
1	1
2	2
3	1
4	2
5	2
6	3
7	1
8	2
9	3
10	3

use previous	**weight_k1**	0.67
	weight_k2	0.64
	weight_k3	0.70

weight_(k1:k3)	**2.645**

applying gradient descent	
w1	0.342
w2	0.307
w3	0.374

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	**actual class**	**error**
class of xq	1.329	1	0.329

learning rate = alpha

in the weight (update weig	
w1	0.408
w2	0.373
w3	0.440

w’ = w + (learning rate * error)

 # round 2 **test data**

#	class	x1	y1
2	1	0.70	0.64

#	Rank	distance	class
1	4	0.12	1
2	1	0.04	1
3	2	0.06	2
4	6	0.15	2
5	10	0.21	1
6	3	0.09	2
7	5	0.12	2
8	9	0.18	3
9	7	0.18	3
10	8	0.18	3

Rank	Class
1	1
2	2
3	2
4	1
5	2
6	2
7	3
8	3
9	3
10	1

use previous	**weight_k1**	0.41

weight_(k1:k3)	**2.032**

weight_k2	0.37
weight_k3	0.44

applying gradient descent	
w1	0.201
w2	0.166
w3	0.233

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	actual class	error
class of xq	1.000	1	0.000

learning rate = alpha

w’ = w + (learning rate * error)

in the weight (update weig	
w1	0.201
w2	0.166
w3	0.233

 # round 3 **test data**

#	class	x1	y1
3	2	0.77	0.68

#	Rank	distance	class
1	7	0.81	1
2	3	0.74	1
3	2	0.69	2
4	5	0.76	2
5	1	0.57	1
6	4	0.75	2
7	8	0.81	2
8	10	0.84	3
9	6	0.81	3
10	9	0.83	3

Rank	Class
1	1
2	2
3	1
4	2
5	2
6	3
7	1
8	2
9	3
10	3

use previous	weight_k1	0.201
	weight_k2	0.166
	weight_k3	0.233

weight_(k1:k3)	0.767

applying gradient descent	
w1	0.248
w2	0.213
w3	0.280

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	actual class	error
class of xq	0.953	2	1.047

learning rate = alpha

w’ = w + (learning rate * error)

in the weight (update weig	
w1	0.457
w2	0.422
w3	0.489

 # round 4 **test data**

#	class	x1	y1
4	2	0.81	0.74

#	Rank	distance	class

Rank	Class

1	9	0.210	1
2	8	0.118	1
3	7	0.105	2
4	1	0.000	2
5	10	0.336	1
6	3	0.056	2
7	6	0.097	2
8	5	0.077	3
9	2	0.046	3
10	4	0.065	3

1	2
2	2
3	1
4	2
5	2
6	3
7	1
8	2
9	3
10	3

use previous	**weight_k1**	0.457
	weight_k2	0.422
	weight_k3	0.489

weight_(k1:k3)	**2.248**

applying gradient descent	
w1	0.208
w2	0.173
w3	0.240

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	**actual class**	**error**
class of xq	1.000	2	1.000

learning rate = alpha

ain the weight (update weigl	
w1	0.408
w2	0.373
w3	0.440

w' = w + (learning rate * error)

 # round 5 **test data**

#	**class**	**x1**	**y1**
5	3	0.78	0.82

#	**Rank**	**distance**	**class**
1	9	0.214	1
2	7	0.160	1
3	8	0.177	2
4	4	0.085	2
5	10	0.399	1
6	6	0.107	2
7	5	0.089	2
8	1	0.012	3
9	3	0.041	3
10	2	0.020	3

Rank	**Class**
1	3
2	3
3	3
4	2
5	2
6	2
7	1
8	2
9	1
10	1

use previous	**weight_k1**	0.408
	weight_k2	0.373
	weight_k3	0.440

weight_(k1:k3)	**3.659**

applying gradient descent	
w1	0.124
w2	0.159
w3	0.092

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	**actual class**	**error**

class of xq	1.127	3	1.873

learning rate = alpha

w' = w + (learning rate * error)

ain the weight (update weig	
w1	0.499
w2	0.534
w3	0.467

 # round 6 **test data**

#	class	x1	y1
6	3	0.82	0.79

#	Rank	distance	class
1	9	0.234	1
2	8	0.160	1
3	7	0.159	2
4	4	0.056	2
5	10	0.389	1
6	5	0.099	2
7	6	0.108	2
8	3	0.043	3
9	1	0.017	3
10	2	0.028	3

Rank	Class
1	3
2	3
3	3
4	2
5	2
6	2
7	2
8	1
9	1
10	1

use previous	weight_k1	0.499
	weight_k2	0.534
	weight_k3	0.467

weight_(k1:k3)	4.498

applying gradient descent	
w1	0.201
w2	0.166
w3	0.233

Gradient descent = w_(i) + (learning rate * (1 – weight_(k1:k3)))

	predict class	actual class	error
class of xq	1.798	3	1.202

learning rate = alpha

w' = w + (learning rate * error)

ain the weight (update weig	
w1	0.441
w2	0.406
w3	0.473

Mean Square Error (MSE)

round	predict class	actual class	error	quare Error
1	1.33	1.00	0.33	0.108
2	1.00	1.00	0.00	0.000
3	0.95	2.00	-1.05	1.095
4	1.00	2.00	-1.00	1.000
5	1.13	3.00	-1.87	3.506
6	1.80	3.00	-1.20	1.444
			MSE	1.192

Mean Absolute Error (MAE)

round	predict class	actual class	error	absolute error
1	1.33	1.00	0.33	0.329
2	1.00	1.00	0.00	0.000
3	0.95	2.00	-1.05	1.047
4	1.00	2.00	-1.00	1.000
5	1.13	3.00	-1.87	1.873
6	1.80	3.00	-1.20	1.202
			sum	**5.449829**
			MAE	**1.089966**

	n
	1
	0
	1
	1
	1
	1
sum	**5**

Epoch	MSE	MAE
1	1.677	1.177
2	1.192	1.090