

MACHINE LEARNING (UE22CS352A)



Weighted K-Nearest Neighbour

Course Instructor: Dr Pooja Agarwal
Professor, Dept. of CSE

MACHINE LEARNING

Weighted K-Nearest Neighbour

The slides are generated from various internet resources and books, with valuable contributions from multiple professors and teaching assistants in the university.

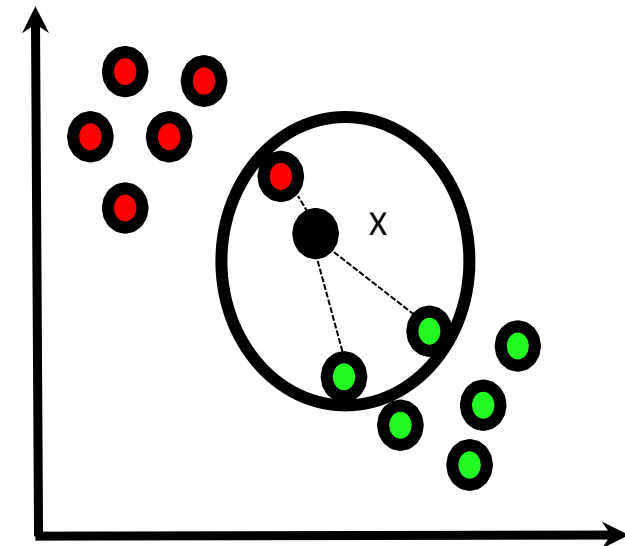


- Consider the given 2-D plot of a data set where the label is binary.
- Consider the query point x_q
- With vanilla K-NN where $K = 3$:
 - x will be classified as green, when it's more likely to be red.

How to handle this?

K-NN follows the basic rule of life!

The closer you are to me the more importance I give you.



- In *majority voting approach*, every **neighbor has the same impact** on the classification.
- This factor makes classification algo. sensitive to the choice of k.
- In order to reduce this impact of k, we assign weight to each of the distance for NN say x_i :

$$w_i = \frac{1}{d(x_q, x_i)^2}$$

- As a result of applying wt. to the distance, the training examples that are **located far away from x_q have a weaker impact on the classification.**
- Using the distance-weighted voting scheme, the class label can be determined:

Distance-wtd. Voting

$$y' = \arg \max_v \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$$

1. NN classification is a part of instance-based learning.
2. Lazy learners like NN classifiers do not need model building(They don't build model).
3. NN classifiers make their predictions based on local information whereas DT and rule-based classifiers attempt to find a global model that fits the entire input space (Not good for generalization)
4. Appropriate proximity measures(similarity measure/ distance measures) play a significant role in NN classifiers.

- Distance-weighted nearest neighbor algorithm
- Weight the contribution of each of the k neighbors according to their distance to the query x_q
- *Greater the weight, closer the neighbors*

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

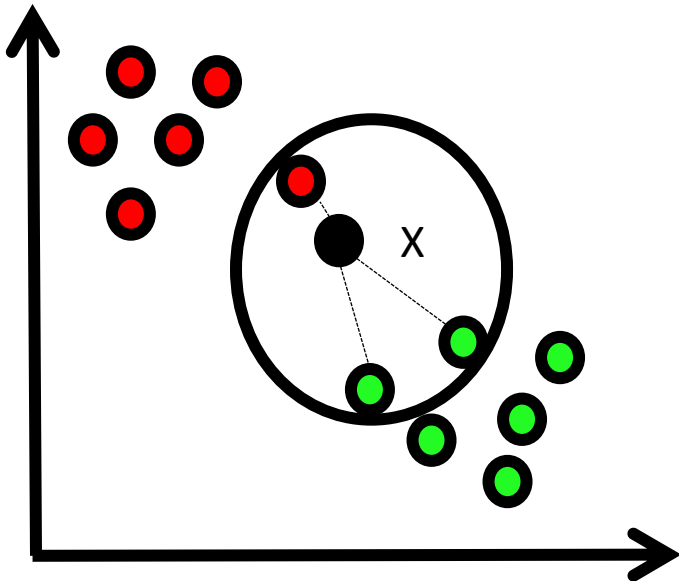
$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Consider the given 2-D plot of a two class data set

Consider the query point x

With normal KNN with $K=3$, x will be classified as green

With weighted KNN and $K=3$, x will be classified as red since its more close to the query x_q

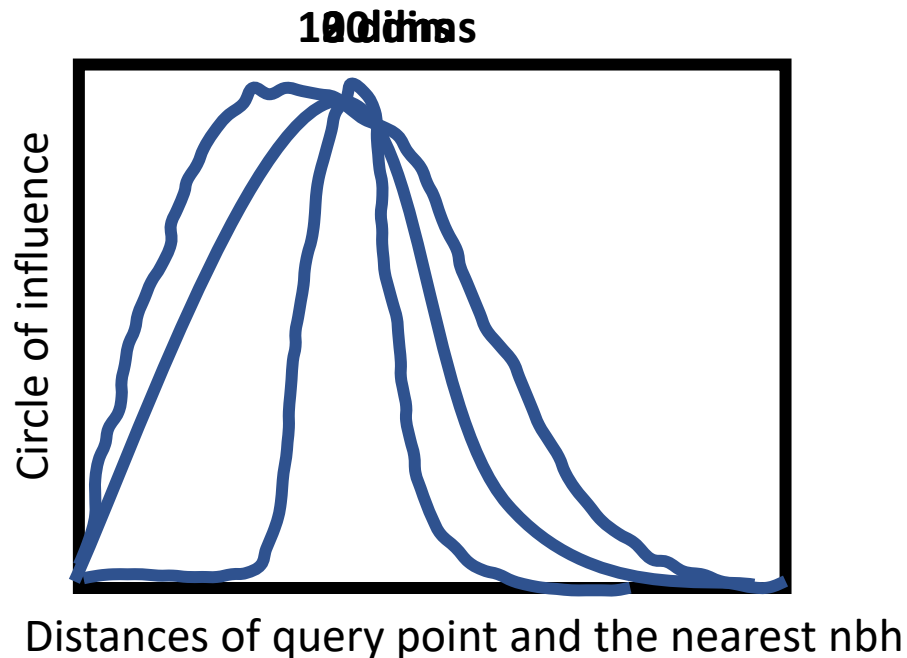


The classification of an instance x , will be *most similar to the classification of the K other instances that are in vicinity(neighborhood).*

In simple English **Birds of the same feather flock together**

- **K-NN slow algorithm:** K-NN might be very easy to implement **but as data set grows** efficiency or speed of algorithm declines very fast.
- **Curse of Dimensionality:** KNN works well with small number of input variables but as the **numbers of variables grow** K-NN algorithm struggles to predict the output of new data point.

- The *Radius or circle of influence* of each data point becomes smaller and smaller as we have more attributes.



The curse of Dimensionality can be overcome by:

- **Assigning weights** to the attributes when calculating distances.
- Using the “**Leave-one-out**” approach. Iteratively, we leave out one of the attributes and test the algorithm by the cross-validation method. The exercise can then lead us to the best set of attributes.

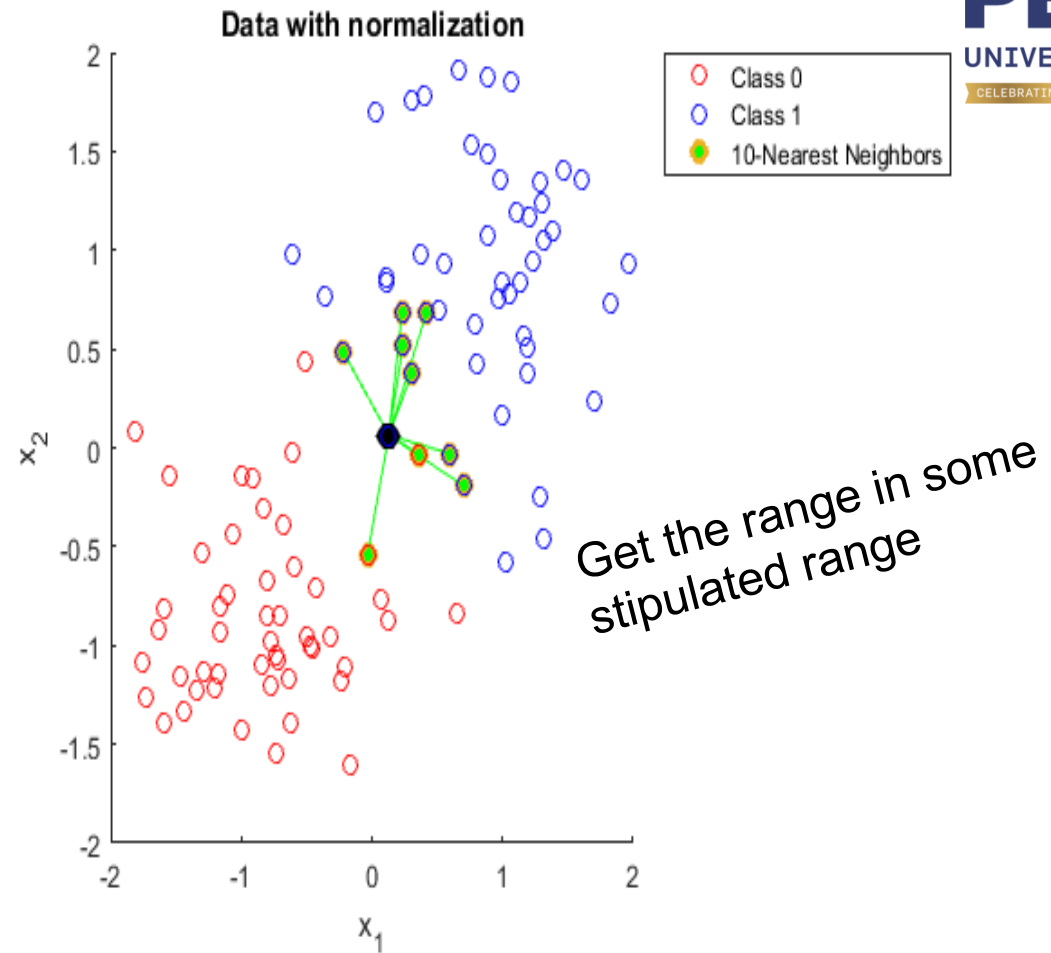
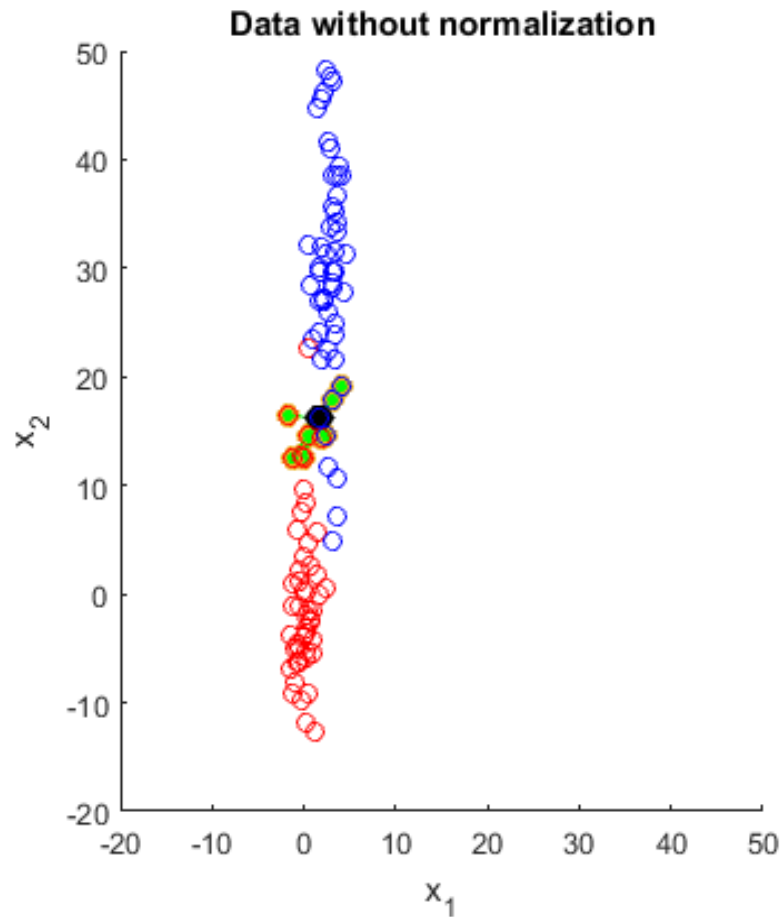
- **Optimal number of neighbors:** One of the biggest issues with K-NN is **to choose the optimal number of neighbors** to be considered while classifying the new data entry
- **Imbalanced data causes problems:** k-NN **doesn't perform well on imbalanced data**. If we consider two classes, A and B, and the majority of the training data is labelled as A, then the model will ultimately give a lot of preference to A. This might result in getting the less common class B wrongly classified.

- **Outlier sensitivity:** K-NN algorithm is **very sensitive to outliers** as it simply chose the neighbors based on distance criteria.
- **Missing Value treatment:** K-NN inherently has no capability of dealing with missing value problem.

- **K-NN needs homogeneous features:** If you decide to build k-NN using a common distance, like **Euclidean or Manhattan distances**, it is completely necessary that **features have the same scale**, since absolute differences in features weight the same, i.e., a given distance in feature 1 must mean the same for feature 2

MACHINE LEARNING

Issues with KNN



| ID | Age | Income(rupees) |
|----|-----|----------------|
| 1 | 25 | 80,000 |
| 2 | 30 | 100,000 |
| 3 | 40 | 90,000 |
| 4 | 30 | 50,000 |
| 5 | 40 | 110,000 |

Source: [Applied Machine Learning Course](#)

Lets calculate the distance between any two data points , say 1 and 2. Clearly the **Euclidean distance between these two points is quite large** bcoz of the **higher magnitude of Income attribute**.

So any distance based models are **highly sensitive to the scale of data** and thus influence the final outcome.

To avoid this biasedness, **scaling or normalization** of attributes is required.

Many techniques, **like computing mean and std deviation** of the attribute and

$$z = (x - \text{mean}) / \text{std. deviation}$$

Or One can use **Min-Max scaling** also

$$= (x - x_{\min}) / (x_{\max} - x_{\min})$$

MACHINE LEARNING

Effect of Normalizing

PRE NORMALIZING:

Euclidean distance between 1 and 2=

$$[(100000 - 80000)^2 + (30 - 25)^2]^{1/2} = \mathbf{20000}$$

| ID | Age | Income(rupees) |
|----|-----|----------------|
| 1 | 25 | 80,000 |
| 2 | 30 | 100,000 |
| 3 | 40 | 90,000 |
| 4 | 30 | 50,000 |
| 5 | 40 | 110,000 |

MACHINE LEARNING

Effect of Normalizing

High magnitude of income affected the distance between the two points.

This will impact the performance as higher weightage is given to variables with higher magnitude.

| ID | Age | Income(rupees) |
|----|-----|----------------|
| 1 | 25 | 80,000 |
| 2 | 30 | 100,000 |
| 3 | 40 | 90,000 |
| 4 | 30 | 50,000 |
| 5 | 40 | 110,000 |

MACHINE LEARNING

Effect of Normalizing

NORMALISING:

$$x_i = \frac{x_i - \mu}{\sigma}$$

μ = Mean

σ = Standard Deviation

here,

$$\mu_{age} = 33, \sigma_{age} = 6$$
$$\mu_{income} = 86000, \sigma_{income} = 20591.26$$

| ID | Age | Income(rupees) |
|----|-----|----------------|
| 1 | 25 | 80,000 |
| 2 | 30 | 100,000 |
| 3 | 40 | 90,000 |
| 4 | 30 | 50,000 |
| 5 | 40 | 110,000 |

POST NORMALISING:

Euclidean distance between 1 and 2 =
$$[(0.608 + 0.260)^2 + (-0.447 + 1.192)^2]^{1/2}$$

= **1.14**

Distance is not biased towards the income variable anymore.

Similar weightage given to both the variables.

| ID | Age | Income(rupees) |
|----|-----|----------------|
| 1 | 25 | 80,000 |
| 2 | 30 | 100,000 |
| 3 | 40 | 90,000 |
| 4 | 30 | 50,000 |
| 5 | 40 | 110,000 |

MACHINE LEARNING

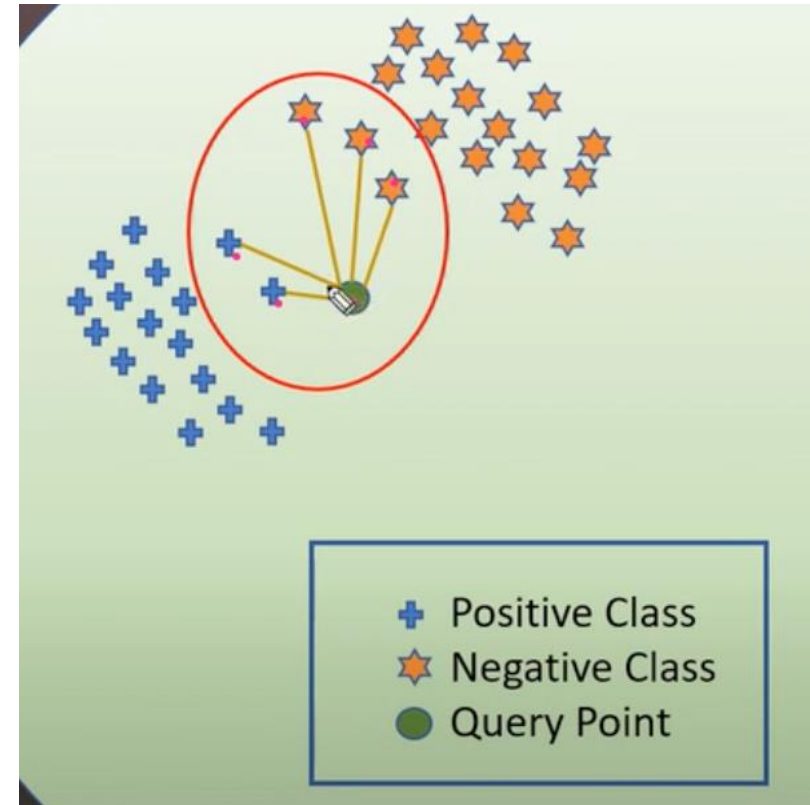
Weighted KNN

$$weight = F(distance) = \frac{1}{distance}$$

Give Weights based on distance

| Neighbour | True Label | Distance | Weight | Sum of Weights |
|-----------|------------|----------|--------|----------------|
| X_1 | Positive | 0.1 | 10 | 13.3 |
| X_2 | Positive | 0.3 | 3.3 | |
| X_3 | Negative | 1 | 1 | 1.8 |
| X_4 | Negative | 2 | 0.5 | |
| X_5 | Negative | 3 | 0.3 | |

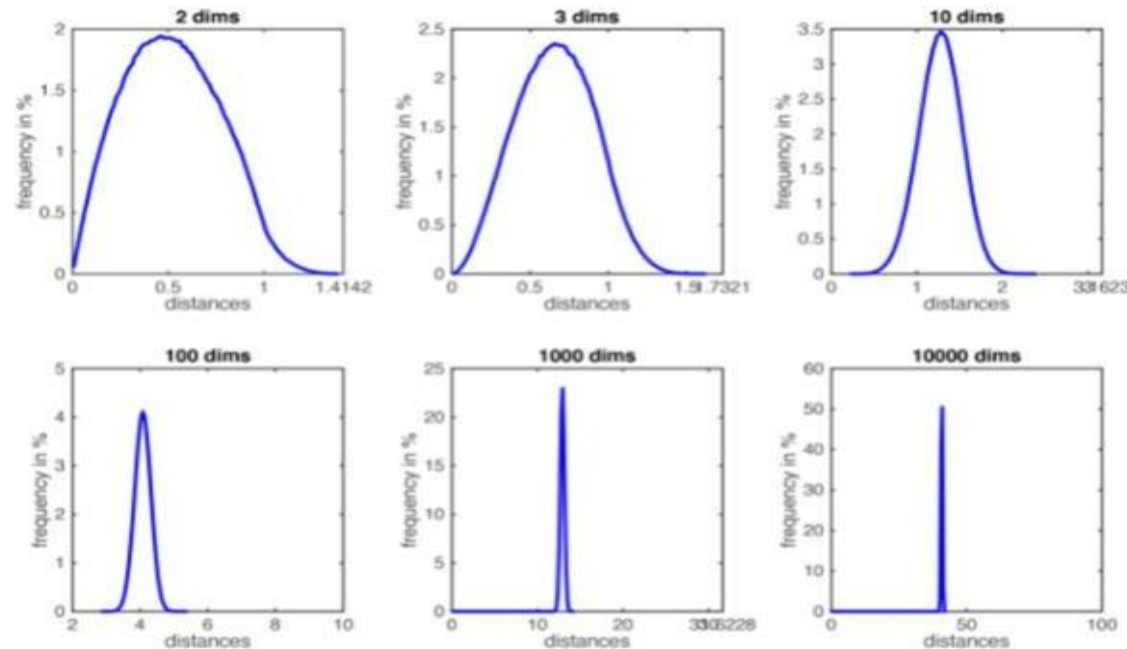
Predict Class labels based on the weighted-sum
not on majority vote



- Basic kNN algorithm stores all examples
- Suppose we have n examples each of dimension d
- $O(d)$ to compute distance to one example
- $O(nd)$ to find one nearest neighbor
- $O(knd)$ to find k closest examples
- Thus complexity is $O(knd)$
- This is prohibitively expensive for large number of samples
- But we need large number of samples for kNN to work well!

The histogram plots show the distributions of all pairwise distances between randomly distributed points within d -dimensional unit squares.

As the **number of dimensions d grows**, all distances concentrate within a very small **range**.



One might think that one rescue could be to increase the number of training samples, n , until the nearest neighbors are truly close to the test point.

How many data points would we need such that ℓ becomes truly small?




- Fix $\ell=1/10=0.1$
- $n=k/\ell^d=k*10^d$, which grows exponentially!
- For $d>100$ we would need far more data points than there are electrons in the universe...

The Curse Of Dimensionality – Dealing With It

- Assigning weights to the attributes when calculating distances. Let's say we're predicting the price of a house, we give higher weights to features like area and locality when compared to color.
- Iteratively, we leave out one of the attributes and test the algorithm. The exercise can then lead us to the best set of attributes.
- Dimensionality reduction using techniques like PCA.

Handling Types Of Attributes

- In case of Boolean values, then **convert to 0 and 1**.
- **Non-binary characterizations:**
 - Natural order among the data, then convert to numerical values based on order.
E.g. Educational attainment: HS, College, MS, PhD -> 1, 2, 3, 4.
 - No order and > 1 category, then convert to one hot encoding.
E.g. Animals: Cat, Dog, Zebra -> (1, 0, 0), (0, 1, 0), (0, 0, 1)

| | Cat | Dog | Zebra |
|---|-----|-----|-------|
|  | 1 | 0 | 0 |
|  | 0 | 1 | 0 |
|  | 0 | 0 | 1 |

Handling Types Of Attributes – Categorical Features

Let us say we have the following data:

- Education, Place and Gender are attributes
- Eligibility is the target
- Education has natural order that is High school < College < Ph.d
- Place needs to be one-hot encoded
- Gender is binary

| Education | Place | Gender | Eligibility |
|-------------|----------|--------|-------------|
| High school | Banglore | M | Yes |
| College | Udupi | F | No |
| Phd | Mandya | M | No |
| Phd | Mandya | M | Yes |
| College | Udupi | F | No |
| High School | Mandya | M | No |
| Phd | Udupi | M | Yes |
| College | Banglore | F | Yes |
| Phd | Banglore | F | Yes |
| High School | Mandya | M | No |

Converted data:

| Education | Place | Gender | Eligibility |
|-------------|----------|--------|-------------|
| High school | Banglore | M | Yes |
| College | Udupi | F | No |
| Phd | Mandya | M | No |
| Phd | Mandya | M | Yes |
| College | Udupi | F | No |
| High School | Mandya | M | No |
| Phd | Udupi | M | Yes |
| College | Banglore | F | Yes |
| Phd | Banglore | F | Yes |
| High School | Mandya | M | No |

| Education | P1 | P2 | Gender | Eligibility |
|-----------|----|----|--------|-------------|
| 1 | 0 | 0 | 1 | Yes |
| 2 | 0 | 1 | 0 | No |
| 3 | 1 | 0 | 1 | No |
| 3 | 1 | 0 | 1 | Yes |
| 2 | 0 | 1 | 0 | No |
| 1 | 1 | 0 | 1 | No |
| 3 | 0 | 1 | 1 | Yes |
| 2 | 0 | 0 | 0 | Yes |
| 3 | 0 | 0 | 0 | Yes |
| 1 | 1 | 0 | 1 | No |

Converted data:

- In Education 1,2,3 represents high school college and Phd respectively.
- Place which had three types of value is broken into two attribute p1 and p2 where $(p1,p2)=(0,0)$ represents bangalore, $(p1,p2)=(0,1)$ represents Udupi and $(p1,p2)=(1,0)$ represents Mandya.
- Since Gender is binary here , we can give 0 to Female and 1 to Male or the other way too.

| Education | P1 | P2 | Gender | Eligibility |
|-----------|----|----|--------|-------------|
| 1 | 0 | 0 | 1 | Yes |
| 2 | 0 | 1 | 0 | No |
| 3 | 1 | 0 | 1 | No |
| 3 | 1 | 0 | 1 | Yes |
| 2 | 0 | 1 | 0 | No |
| 1 | 1 | 0 | 1 | No |
| 3 | 0 | 1 | 1 | Yes |
| 2 | 0 | 0 | 0 | Yes |
| 3 | 0 | 0 | 0 | Yes |
| 1 | 1 | 0 | 1 | No |

- <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
- <https://www.geeksforgeeks.org/instance-based-learning/>
- <https://www.cs.ucdavis.edu/~vemuri/classes/ecs271/ch8.pdf>
- <https://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python/>
- <https://datagy.io/python-knn/>
- https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_k-NN_k-nearest-neighbors-algorithm.php
- <https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/>