**AI AND MACHINE LEARNING APPLICATION TO ELECTRICAL ENGINEERING**

A

Report

Submitted in partial fulfillment of the

**EE441: Electrical Project**

Course for the

Award of B. Tech (Electrical) Degree

EE441: Electrical Project

By

Project Group No: 1

**Bhavesh H. Samani (17EE055)**

**Jagdish D. Yadav (17EE057)**

Under the guidance of

**Dr. Rashesh P. Mehta**

**Prof. Jaydeepsinh C. Baria**

At

**Electrical Engineering Department**

**Birla Vishvakarma Mahavidyalaya (Engineering College)**

**(An Autonomous Institute)**

**Vallabh Vidyanagar: 388120**

**(May 2021)**

**Birla Vishvakarma Mahavidyalaya (Engineering College)**

An Autonomous Institution

**Vallabh Vidyanagar 388120**

***Certificate***

Certified that, work presented by **Mr. Bhavesh Samani** and **Jagdish Yadav** bearing **ID. No. 17EE055, 17EE057** & **Enrollment No: 170070109053, 170070109062** in the Project entitled **AI AND MACHINE LEARNING APPLICATION TO ELECTRICAL ENGINEERING** for EE441: Electrical Project of B. Tech (Electrical) Final year is satisfactory to warrant its acceptance for the purpose of evaluation. It is to be understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusions drawn but approve the study for the purpose for which it is submitted.

Project Faculty Guide:

**Dr. Rashesh P. Mehta**

**Prof. Jaydeepsinh C. Baria**

Head

Electrical Engineering Department

Date:

**Acknowledgement**

We would like to thank with a deep sense of gratitude to all those who helped us to complete our project report. We thank all of them to give their valuable time and guidance in the completion of this project.

We are grateful to our respected project guide and Head of Department Dr. Rashesh P. Mehta sir and Prof. Jaydeepsinh C. Baria sir for their invaluable guidance and support throughout the project as a project guide. We would like to express our sincere gratitude to our instructor who helped us with our project by sharing their suggestion and instruction and experiences.

Last but not the least, heartily thanks to our friends & other faculties who helped us in making this machine learning project.

**Abstract**

Artificial intelligence and machine learning include a broad range of algorithms and statistical models that make it possible for systems to **predict future data, find patterns and learn to perform tasks without specific instructions.**

In this project we have made five ML models. With the help of first (Linear regression) and second (polynomial regression) model **we can predict the future power generation in India with specific accuracy**. With the help of third (K-means clustering) and fourth (Hierarchical clustering) model **we can identify the optimal location for the installation of DG units in the Distribution Systems**. Last model is k-nearest neighbors (KNN) classification. **It enables us to differentiate among three phases which phase of a three phase power system is experiencing a fault**.

**TABLE OF CONTENTS**



Certificate …………………………………………………………………………………………………………… i

Acknowledgement ………………………………………………………………………………………………. ii

Abstract ……………………………………………………………………………………………………………… iii

List of Tables ………………………………………………………………………..………………………….…. iv

List of Figures ………………………………………………………………..….………………………………… v

List of Abbreviations and Nomenclature used …………………..…….…………………………… vi

1. Project Orientation ………………………………………………………………………………………… 08
2. Introduction………………...…………………………………………………………………………………. 09
3. Project Definition, Project Identification, Objective(s)………………………………………..11
4. Project Plan ……………………….…………………………………………………………………………….12
5. Literature Review …………………………………………………………………………………………….13
6. Model 1(Linear regression) ……….………………………………………………………………………14
7. Model 2(Polynomial regression)………………………………………………………………………...25
8. Model 3(K-means clustering) …………………………………………………………………………….31
9. Model 4(Hierarchical clustering) ……………………………………………………………………….40
10. Model 5(KNN classification)………………………………………………………………………………49

References ……………………………………………..………………………………………………………….….61

Plagiarism Report…………………………………………………………………………………………………..62

**List of Figures:-**

**Chapter 2**

Figure 2.1 (Types of ML models). ………………………………………………………………Page No: 10

**Chapter 6**

Fig 6.1 (Linear regression model)…………………….………………………………………. Page No: 14

Fig 6.2 (Output of regression model)…………………………………………………………. Page No: 16

**Chapter 7**

Fig 7.1 (polynomial model) ……………………………………………………………………….Page No: 25

Fig 7.2 (Block diagram) …………………………………………………………………………….Page No: 26

**Chapter 8**

Fig 8.1 (k means cluster model)…………………………………………………………………Page No: 33

Fig 8.2 (Elbow method)……………………………………………………………………………. page No: 35

Fig 8.3(input of model 3)…………………………………………………………………………...page No: 36

Fig 8.4 (Output of model 3)………………………………………………………………………. page No: 37

**Chapter 9**

Fig 9.1 (Output of model 4)………………………………………………………………………..page No: 42

**Chapter 10**

Fig 10.1, 10.2 (Working of KNN model)………………………………………………….page No: 51, 52

Fig 10.3, 10.4 (Output of model 5)……………..…………………………………………page No: 53,54

**List of Tables:-**

**Chapter 2**

Table 2.1 Different ML libraries for this project.

Page no: 10

**Chapter 4**

Table 4.1 Project plan.

Page No: 12

**Chapter 5**

Table 5.1 Literature review.

Page No: 13

**Chapter 7**

Table 7.1 Conclusion of the model 2.

Page no: 28

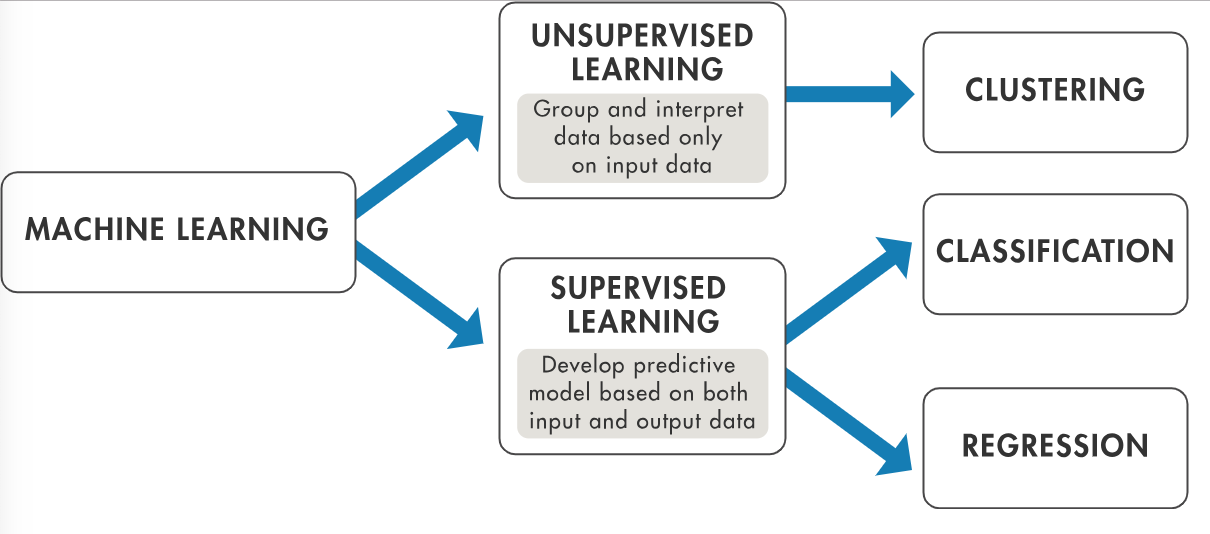
1. **PROJECT ORIENTATION:-**

**1.1 WHAT IS THE ARTIFICIAL INTELLIGENCE (AI) AND MACHINE LEARNING (ML)?**

* Artificial intelligence and machine learning are the part of computer science that are correlated with each other. These two technologies are the most trending technologies which are used for creating intelligent systems.
* Although these are two related technologies and sometimes people use them as a synonym for each other, but still both are the two different terms in various cases.
* *Artificial intelligence is a technology using which we can create intelligent systems that can simulate human intelligence.*
* *Machine learning is a subfield of artificial intelligence, which enables machines to learn from past data or experiences without being explicitly programmed.*

1. **INTRODUCTION :**

* **Machine Learning** is an application of artificial intelligence (AI) that provides systems the ability to automatically learn, improve from experience and then future data can be predicted.
* **2.1 Applications:**
* Load forecasting.
* Fault detection and classification in electrical power transmission system.
* Load classification in power system.
* Reactive power planning and its control.
* Identifying optimal location for the installation of DG units and capacitor bank in the Distribution Systems.
* Speed control of the machine.
* **2.2 ALGORITHMS :-**



**Figure 2.1: ML Algorithms**

**2.3 DIFFERENT ML LIBRARIES:-**

|  |  |  |
| --- | --- | --- |
| **NO.** | **Library name:** | **Feature of library:** |
| 1. | Numpy | N-darray,mathematics operation |
| 2. | pandas | Data-frame |
| 3. | matplotlib | Plotting library |
| 4. | scikit-learn. | Machine learning tools. |

**Table 2.1. ML Libraries**

1. **PROJECT DEFINITION & OBJECTIVE OF THE PROJECT:**

Model 1: To predict the power generation in India using Linear Regression Model.

Model 2: To predict the power generation with higher accuracy (above 99%) using Polynomial Regression Model.

Model 3: Clustering Analysis and its Application in Electrical Distribution System using K-means Clustering Model.

Model 4: Clustering Analysis and its Application in Electrical Distribution System using Hierarchical Clustering Model.

Model 5: Load classification and Fault classification using K-Nearest Neighbours (KNN) Model.

1. **PROJECT PLAN:**

|  |  |  |  |
| --- | --- | --- | --- |
| Task No. | Title of the work. | Time(moth) | Work description. |
| 1 | Libraries of ML and model 1. | Jan-Feb | Numpy, pandas, matplotlib, sklearn.  Linear regression model. |
| 2 | ML model2. | Feb-Mar | Polynomial regression model.  (Supervised machine learning.) |
| 3 | ML model3. | Mar-Apr | K-means clustering.  Unsupervised machine learning. |
| 4 | Model 4 and Model 5 | Apr-may | Hierarchical clustering.  (Unsupervised machine learning.)  KNN classification (supervised ML) |

**Table 4.1 Project Plan**

1. **LITERATURE REVIEW:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model Name:** | **Linear and polynomial regression:** | **K-means and Hierarchical clustering:** | **KNN classification:** |
| **Title of research paper:** | Internet | Clustering Analysis and its Application in Electrical Distribution. | Fault classification in electrical power transmission. |
| **Authors:** | - | K.M.Rosalina, N.P. Kumar | Majid Jamil, Rajveer Singh |
| **Link:** | <http://irdindia.in/journal_ijraet/pdf/vol8_iss6/6.pdf>  <https://springerplus.springeropen.com/track/pdf/10.1186/s40064-015-1080-x.pdf> | | |
| **Method:** | Supervised ML. | Unsupervised ML. | Supervised ML. |
| **Input :**  **Output:** | Feature: years(1950-2019)  Label: Electricity generation of the given year(GWH)  We will get the best fit curve with accuracy. | Voltage and power loss factor of different load.  We will get the cluster of D.G(Distributed Generation) units for different load. | Feature: current values:(I1,I2,I3)  Label : Fault (L1-G,L2-G,L3-G)  We will get trained model  With specific accuracy. |
| **Accuracy:** | 80% to 100% | Above 90-100% | 90-100% |
| **Advantages:** | exceptionally well for linearly data and  Polynomial basically fits a wide range of curvature. | If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering | Very simple implementation |
| **Limitation :** | It fails to fit complex datasets properly.  The presence of outliers in the data can seriously affect the results of the model. | The user has to specify k (the number of clusters) in the beginning. | Require high memory to store all of the training data. |

**Table: 5.1. Literature Review**

1. **MODEL 1: LINEAR REGRESSION**

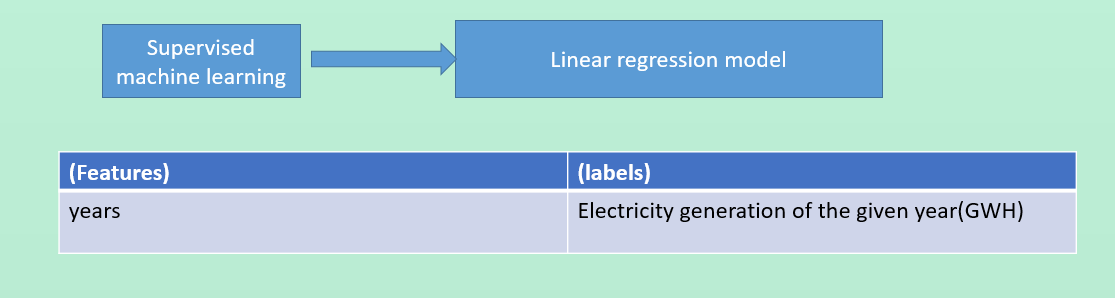
**6.1 WHAT IS THE LINEAR REGRESSION?**

Linear regression is the next step up after correlation. It is used when we want to predict the value of a variable based on the value of another variable. The variable we want to predict is called the dependent variable

**6.2 TRAINING AND TESTING DATA:**

Now, entire dataset is divided into training and testing set so that prediction does not over fit or under fit and correct values are obtained. train\_test\_split() is inbuilt function from scikit learn for splitting x and y variables data. “test\_size” parameter is used for testing data and remaining as training data.

In this Model-1, we are going to use:



**Fig. 6.1: Input of model-1**

**6.3 LOGIC BEHIND SIMPLE LINEAR REGRESSION MODEL:**

As the name suggests, linear regression follows the linear mathematical model for determining the value of one dependent variable from value of one given independent variable.

**6.4 WORKING OF LINEAR REGRESSION:**

First we will give the training data to the model.

Model will draw the best fit line using line formula and SSR.

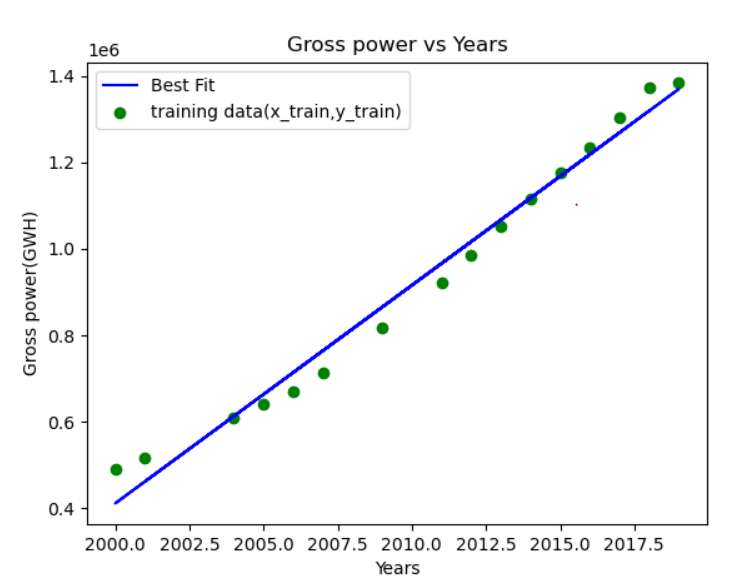
Formula🡪y=mx+c

where y is the dependent variable, m is slope, x is the independent variable and c is the intercept for a given line.

SSR= sum of square of error(y-yi).

Model will try to reduce the error by taking the different values of slop (m) and intercept(c).

Finally the best fit line will be drawn using this method.



**Fig. 6.2: [output of model when dataset is linear]**

**6.5 ACCURACY OF THE MODEL:**

The formula is (1-SSE/SSM)\*100

SSE=Sum of square of error(y-yi)

SSM=sum of square of M.

Where, M🡪 (y-(arithmetic mean of the y))

**6.6 CODING OF THE MODEL 1**:

import pandas as pd

import numpy as np

import os

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from matplotlib import pyplot as plt

from sklearn.metrics import r2\_score as ModelAccuracy

from prettytable import PrettyTable as PT

def csvfiles():

list\_csv=[]

current\_folder=os.getcwd()

files=os.listdir(current\_folder)

for f in files:

if f.split(".")[-1]=="csv":

list\_csv.append(f)

return(list\_csv)

def Table(x\_test,y\_test,y\_predict):

table=PT()

table.field\_names=["Years:","predicted value of power:","Actual value of power:"]

##convert array to list

x\_test\_list=list(x\_test)

y\_test\_list=list(y\_test)

y\_predict\_list=list(y\_predict)

for i in range(len(x\_test)):

table.add\_row([x\_test\_list[i],y\_predict\_list[i],y\_test\_list[i]])

print(table)

def trained\_graph(x\_train,y\_train,myobj\_regression):

plt.scatter(x\_train,y\_train,color='green',label='training data(x\_train,y\_train)')

plt.plot(x\_train,myobj\_regression.predict(x\_train),color='blue',label='Best Fit')

plt.title("Gross power vs Years")

plt.xlabel('Years')

plt.ylabel('Gross power(GWH)')

plt.legend()

plt.show()

def finalgraph(x\_train,y\_train,x\_test,y\_test,y\_predict,myobj\_regression):

plt.scatter(x\_train,y\_train,color='green',label='training data(x\_train,y\_train)')

plt.plot(x\_train,myobj\_regression.predict(x\_train),color='blue',label='Best Fit')

plt.scatter(x\_test,y\_test,color='red',label='known test data(x\_test,y\_test)')

plt.scatter(x\_test,y\_predict,color='black',label='Predicted test data(x\_test,y\_pred)')

plt.title("Gross power vs Years")

plt.xlabel('Years')

plt.ylabel('Gross power(GWH)')

plt.legend()

plt.show()

def main(user\_name):

print(f"Hello {user\_name}\n\nwelcome to Power prediction system:-")

print("Press ENTER key to proceed")

input()

csv\_files=csvfiles() ##This is the list of csv files.

print("choose a csv file for preparing model:-\n")

for i,name in enumerate(csv\_files):

print(f"{i}---->{name}")

index=int(input("Enter the index of the file:"))

model\_csv = csv\_files[index]

df=pd.read\_csv(model\_csv)

x=df.iloc[:,0].values #<class 'numpy.ndarray'> x=year and y=Total power generated

y=df.iloc[:,1].values

x\_list=[[i] for i in x] #[[1],[2],...]

y\_list=[[i] for i in y]

x=np.array(x\_list) #[[1] [2] [3] ...] in 2D array

y=np.array(y\_list)

p=float(input("Enter percentage value for training data:\n") ) #percentage of total data

test\_data=1-(p/100)

x\_train, x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=test\_data) #in which args must be nD array

### Here x\_train,x\_test,y\_train,y\_test are 2D array.###

print("\nModel is being created..\n")

myobj\_regression=LinearRegression()

myobj\_regression.fit(x\_train,y\_train)

print("Model has been created.") # It means (m=slop) and (c=intercept) has been fixed. (y=mx+c)

ans=int(input("For graph--> 1,else 0:-"))

if ans==1:

trained\_graph(x\_train,y\_train,myobj\_regression)

print("would you like to test this model using testing data?")

ans=int(input("if yes-->Enter 1:"))

if ans==1:

y\_predict=myobj\_regression.predict(x\_test)

print("press 1 to see predicted values of testing data ,else 0:-")

if int(input())==1:

Table(x\_test,y\_test,y\_predict) #To show the data in table format.

graph=int(input("if you want to see graph of the model then Enter:1 ,else Enter 0 :-\n"))

if graph==1:

finalgraph(x\_train,y\_train,x\_test,y\_test,y\_predict,myobj\_regression)

pf= ModelAccuracy(y\_test,y\_predict) ##pf=performance\_fact

if pf<0.10:

print("Error ,Run again.")

else:

print("\n")

print("-------------------------------------------------------")

print(f"This model is {round(pf\*100,2) } % accurate.\n")

print("-------------------------------------------------------")

print("Now you can predict Gross power of any year using this model")

while(True):

print("\nEnter the Year value yyyy format:")

x\_newdata=eval(input())

y=myobj\_regression.predict(np.array([[x\_newdata]]))

y=float(y)

print(f"TotalPowewr(gwh): { round(y,2) }")

x=int(input("\nEnter 0 for exit and 1 for continue:"))

if x==0:

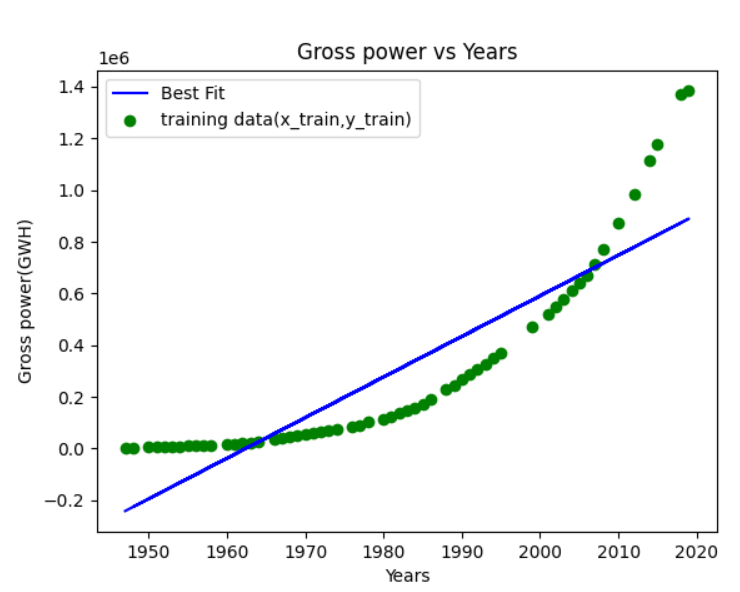
break

if \_\_name\_\_=="\_\_main\_\_":

name=input("Enter your name : ")

main(name)

input()



**Fig 6.3: Output of this model when dataset is not linear**

**6.7 FLOWCHART OF THE LINEAR REGRESSION:**

Start

Select your dataset

Split the training and testing data

Model is ready to predict the power generation

Accuracy > 80%?

NO YES

Training data will be given to the model

Stop

Best fit Line for the model

Testing data will be given to the model

Final Graph of the model

Check the accuracy

**6.8 CONCLUSION:**

If we want to predict the future power generation using dataset 1(from 2000 to 2020) then only we can use this model. We got 96% accuracy.

If we want use dataset 2(from 1950 to 2020) then accuracy will be reduced as shown in above figure. We got only 52% accuracy.

1. **MODEL 2: POLYNOMIAL REGRESSION**

**7.1 DEFINITION:**

Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial.

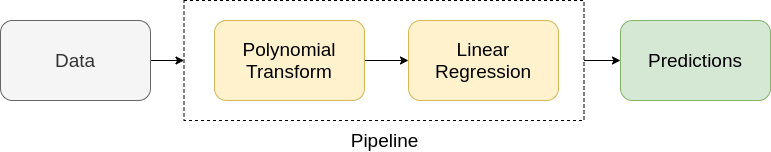
The Polynomial Regression equation is given below:

y= b0+b1x1+ b2x12+ b2x13+...... bnx1n

* It is also called the special case of Linear Regression in ML. Because we add some polynomial terms to the Multiple Linear regression equation to convert it into Polynomial Regression.
* It is a linear model with some modification in order to increase the accuracy.
* The dataset used in Polynomial regression for training is of non-linear nature.

It makes use of a linear regression model to fit the complicated and non-linear functions and datasets.

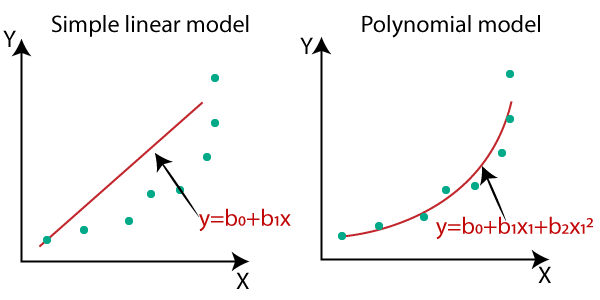
* *In* Polynomial *regression, the original features are converted into Polynomial features of required degree (2,3,..,n) .*



***7.2 NEED FOR POLYNOMIAL REGRESSION:***

The need of Polynomial Regression in ML can be understood in the below points:

* If we apply a linear model on a **linear dataset**, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a **non-linear dataset**, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.
* So for such cases, **where data points are arranged in a non-linear fashion, we need the Polynomial Regression model**. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.



**Fig. 7.1: [difference of linear and polynomial regression]**

* In the above image, we have taken a dataset which is arranged non-linearly. So if we try to cover it with a linear model, then we can clearly see that it hardly covers any data point.
* On the other hand, a curve is suitable to cover most of the data points, which is of the Polynomial model.

**7.3 FLOWCHART OF THE MODEL 2:**

Start

Select your dataset

Split the data and **set the degree of polynomial**

Model is ready to predict the power generation

Accuracy > **95%?**

NO YES

Training data will be given to the model

Stop

**Best fit curve** for the model

Testing data will be given to the model

Final Graph of the model

Check the accuracy

**7.4 CODING OF THE MODEL 2:**

The coding of model 2 is almost similar to model 1. To get polynomial regression, we have to convert linear data to polynomial data and then set the degree of polynomial regression and it can be done using following code:

from sklearn.preprocessing import PolynomialFeatures

poly=PolynomialFeatures(degree=4) #formula :y=w1x+w2x²+..+b

x\_train\_p = poly.fit\_transform(x\_train) #here x\_train will be converted to the x\_train\_p

x\_test\_p = poly.fit\_transform(x\_test)

**7.5 CONCLUSION:**

*If the datasets are arranged in a non-linear fashion, then we should use the Polynomial Regression model instead of Simple Linear Regression to get higher accuracy (above 97%).*

|  |  |  |
| --- | --- | --- |
| No. | **Type of training dataset:** | **Name of the Model:** |
| 1. | Last 10-15 years data | Linear regression. |
| 2. | More than 15 years data | Polynomial regression. |

**Table 7.1 Conclusion**

1. **MODEL 3 K-MEANS CLUSTERING:-**

**8.1 APPLICATION:**

Identifying optimal location for the installation of DG units in Radial Distribution Systems using “K-means cluster” technique.

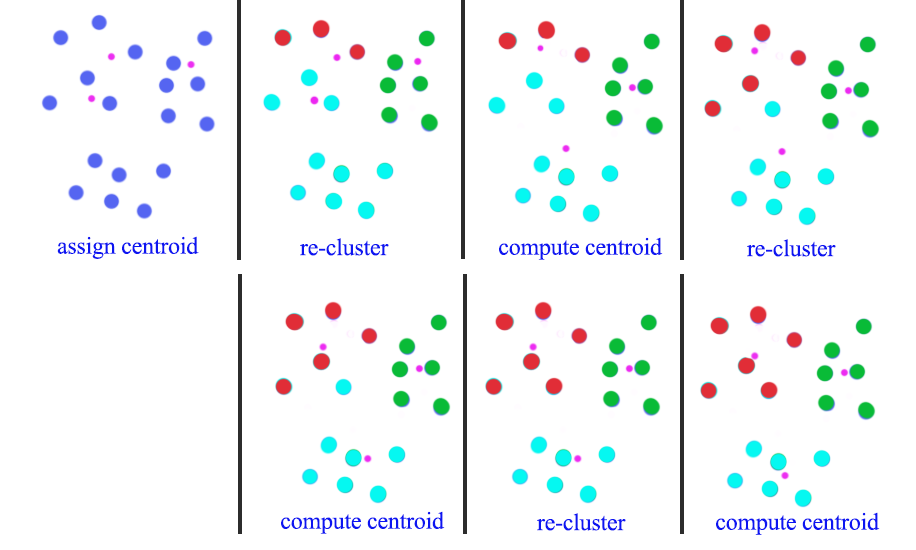
**8.2 INTRODUCTION OF THE DISTRIBUTED GENERATION:**

* + Distributed Generation (DG) in **the form of renewable power generation systems** is currently preferred for clean power generation.
  + **In distribution system, due to load uncertainties the load exceeds the generating capacity which leads to power loss and unreliable operation of the system**.

* To overcome this problem DG units are incorporated into the distribution system to meet the excess demand which results in **power loss minimization, improvement of voltage profile, power quality improvement, reliable operation**, etc.,

**8.3 WHAT IS K-MEANS ALGORITHM?**

* K-Means Clustering is an [Unsupervised Learning algorithm](https://www.javatpoint.com/unsupervised-machine-learning), which groups the unlabeled dataset into different clusters.
* It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.
* Here K defines the number of pre-defined clusters that need to be created in the process,
* As if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
* It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.
* The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.



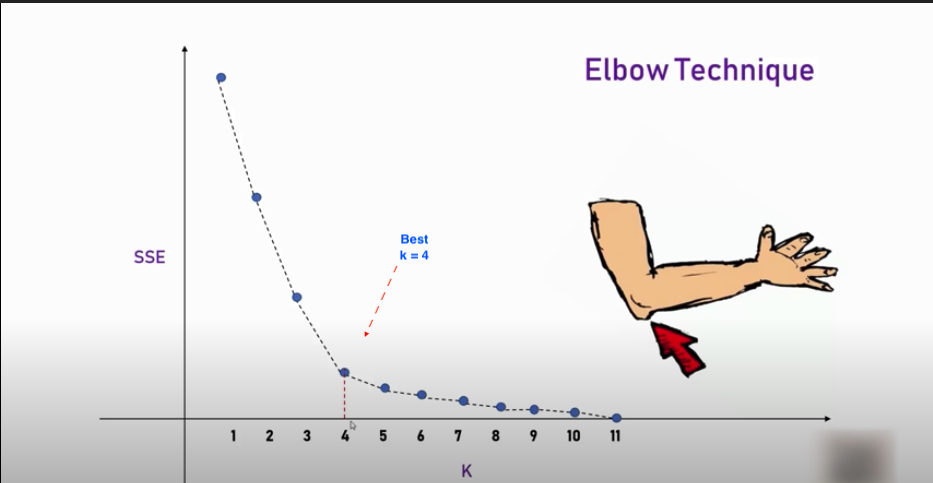
**Fig. 8.1: Working of k-means cluster model**

**8.3.1 THE K-MEANS CLUSTERING ALGORITHM MAINLY PERFORMS TWO TASKS**:

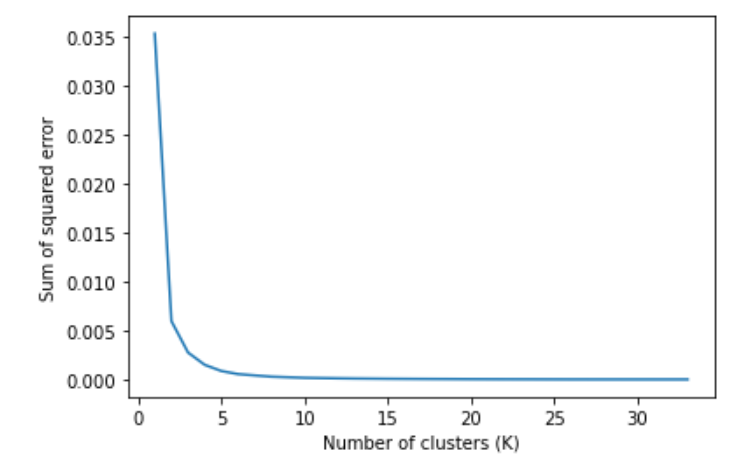
1. Determines the best value for K center points or centroids by an Elbow Method.
2. Assigns each data point to its closest k-center.

**8.4 WHAT IS THE ELBOW METHOD?**

* The Elbow method is one of the most popular ways to ***find the optimal number of clusters***.
* This method uses the concept of WCSS value.
* **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 2 clusters) is given below:
* WCSS= ∑Pi in Cluster1 distance(Pi C1)2 +∑Pi in Cluster2distance(Pi C2)2
* ∑Pi in Cluster1 distance (Pi C1)2:🡪 It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the second term.
* To measure the distance, we can use **Euclidean distance**.



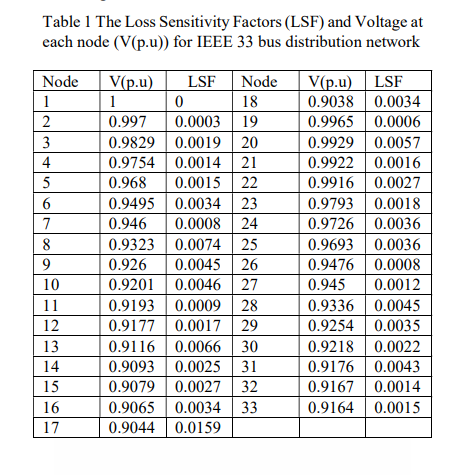
**Fig. 8.2: [Elbow method to find the clusters in K-means model.]**



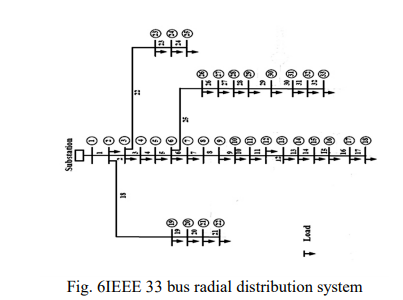
According to our dataset the best value for cluster (K) is 3.

So, there will be three clusters (D.G.) for this model.

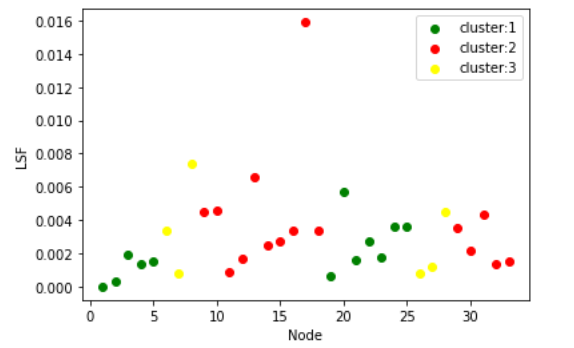
**8.5 INPUT OF THE MODEL:**

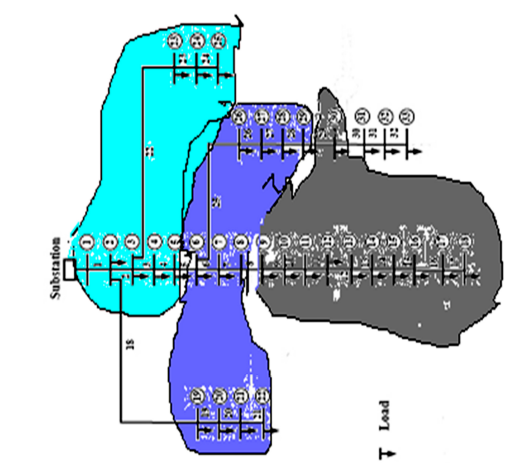


**Fig. 8.3 Input of the model**



**8.6 OUTPUT OF THE MODEL:**





**Fig. 8.4: [output of K-means cluster model]**

**8.7 FLOWCHART OF THE MODEL**:

Start

Select your dataset

**Select the K using Elbow method**

Data will be given to the model

Model is ready with k clusters

Stop

**8.8 CONCLUSION:**

In k-means clustering model, when we change the k value, we will get different clusters. To select this we can use ELBOW method. We got k=3. It means we should use 3 D.G. units for the given load data.

With the help of this model (“K-means cluster”) we can identify the optimal cluster for the installation of DG units in Radial Distribution Systems.

1. **MODEL 4: HIERARCHICAL CLUSTERING:-**

* Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA.
* In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.
* There is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

**9.1 WHY HIERARCHICAL CLUSTERING?**

We have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters.

To solve challenge, we can use the hierarchical clustering algorithm because, in this algorithm, **we don't need to have knowledge about the predefined number of clusters.**

**9.2 THE HIERARCHICAL CLUSTERING TECHNIQUE HAS TWO APPROACHES:**

**Agglomerative**: Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

**Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

**9.2.1 AGGLOMERATIVE HIERARCHICAL CLUSTERING:**

This algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

The working of the AHC algorithm can be explained using the below steps:

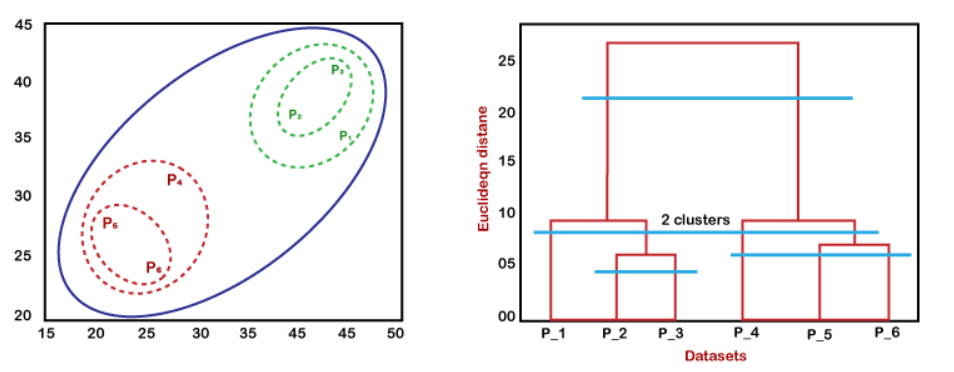
**Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.

**Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

**Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.

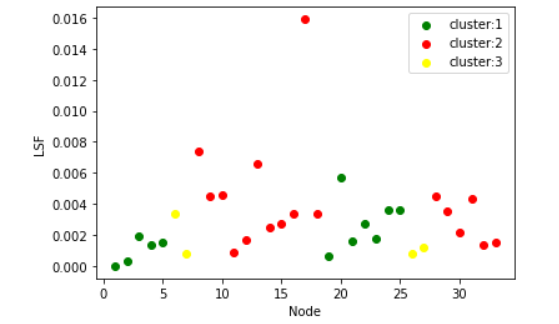
**Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:

**Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters.

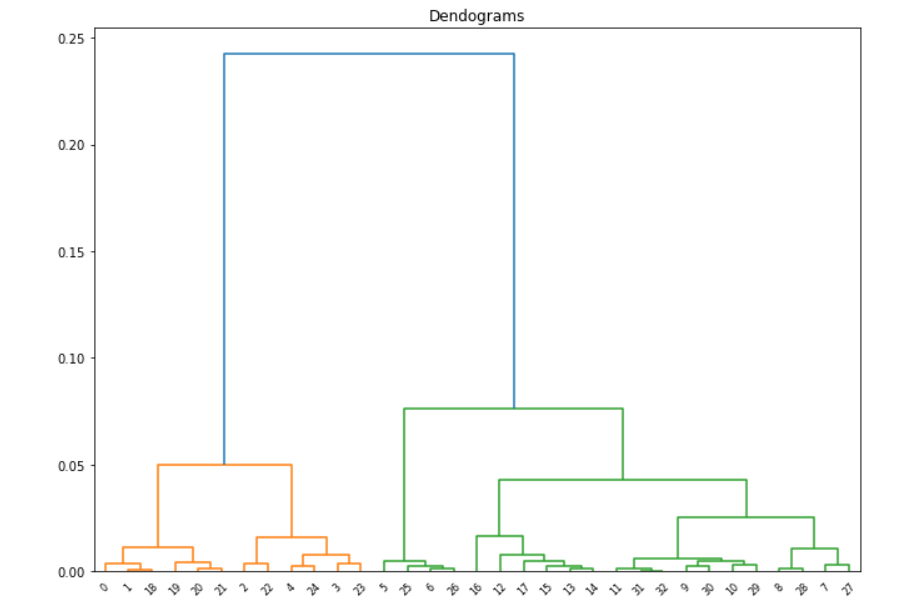


**Fig. 9.1 Agglomerative Hierarchical clustering**

**9.3 OUTPUT OF THE MODEL:**



**Fig. 9.2 output of the model**

****

**Fig. 9.3: output of the model**

**9.4 CODING OF THE MODEL 3 AND 4:**

from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

df = pd.read\_csv("data(V,LSF)kmeans.csv")

plt.scatter(df.Node,df.iloc[:,2].values)

plt.xlabel('Node')

plt.ylabel('LSF')

plt.scatter(df.Node,df.iloc[:,1].values)

plt.xlabel('Node')

plt.ylabel('voltage')

data = df.iloc[:,1:3].values

km = KMeans(n\_clusters=3)

y\_predicted = km.fit\_predict(data)

y\_predicted

df['cluster']=y\_predicted

e=km.cluster\_centers\_

cdf=pd.DataFrame(e,columns=['valtage','LSF'])

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

plt.scatter(df1.Node,df1['LSF'],color='red',label='cluster:1')

plt.scatter(df2.Node,df2['LSF'],color='green',label='cluster:2')

plt.scatter(df3.Node,df3['LSF'],color='yellow',label='cluster:3')

plt.xlabel('Load')

plt.ylabel('LSF')

plt.legend()

plt.show()

sse = []

k\_rng = range(1,34)

for k in k\_rng:

km = KMeans(n\_clusters=k)

km.fit(data)

sse.append(km.inertia\_)

**### The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid. Since this is a measure of error, the objective of k-means is to try to minimize this value. The initialization of the centroids is an important step :**

plt.xlabel('Number of clusters (K)')

plt.ylabel('Sum of squared error')

plt.plot(k\_rng,sse)

**MODEL 4:**

import scipy.cluster.hierarchy as shc

from sklearn.cluster import AgglomerativeClustering

plt.figure(figsize=(11, 8))

plt.title("Dendograms")

dend = shc.dendrogram(shc.linkage(data, method='ward'))

model = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

model.fit(data)

labels = model.labels\_

df = pd.read\_csv("data(V,LSF)kmeans.csv")

df["cluster"]=labels

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

plt.scatter(df1.Node,df1.LSF,color='green',label='cluster:1')

plt.scatter(df2.Node,df2.LSF,color='red',label='cluster:2')

plt.scatter(df3.Node,df3.LSF,color='yellow',label='cluster:3')

plt.xlabel('Node')

plt.ylabel('LSF')

plt.legend()

plt.show()

**9.5 FLOWCHART OF THE MODEL:**

Start

Select your dataset

Data will be given to the model

**Dendograms**

**Select the value cluster(N)**

Model is ready with N clusters

Stop

**9.6 CONCLUSION:**

* The output of the Hierarchical clustering is almost similar to the K-means clustering, so we can say that our model is reliable.
* So, with the help of this model we can predict the location of DG units in distribution system.

**10. MODEL 5 KNN CLASSIFICATION**

Load classification and Fault classification using k-nearest neighbors (KNN) Model.

**10.1 LOAD CLASSIFICATION:**

In the output of previous model (clustering model), we got three clusters (D.G.) for different load.

**In that distribution system, if we want to connect a new load to the D.G. then there are 3 ways (we can connect with DG1 or DG2 or DG3) but which one would be the best choice?**

To get answer of this, we are going to use KNN classification model.

**10.2 FAULT CLASSIFICATION:**

In this, we are going to classify the most common fault (Line-Ground) that occur in the power system.

It enables us to differentiate among three phases which phase of a three phase power system is experiencing a fault.

To classify the single line to ground fault, we are going to use KNN classification model.

**10.3 K-NEAREST NEIGHBOR (KNN) ALGORITHM**

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on **Supervised Learning** technique.

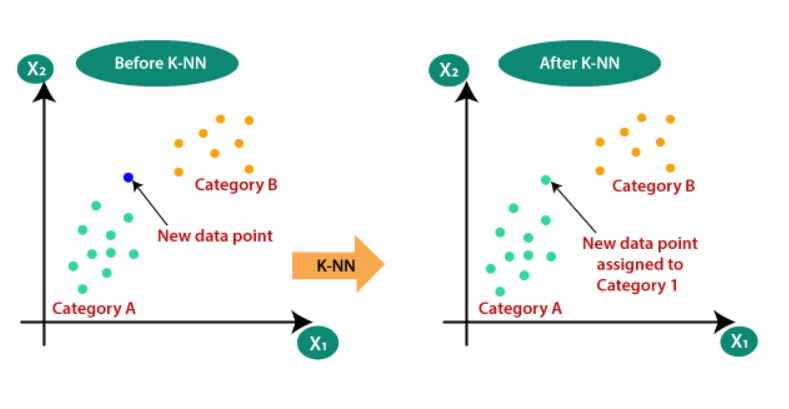
K-NN algorithm assumes the similarity between the new case/data and available cases and **put the new case into the category that is most similar to the available categories.**

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

**10.4 WHAT IS THE USE OF KNN ALGORITHM?**

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.

To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



**Fig. 10.1: [KNN classification]**

**10.5 HOW DOES KNN WORK?**

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

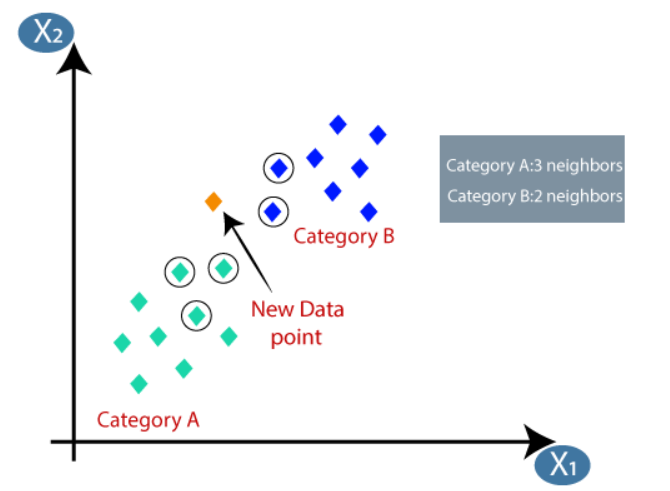
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6**:** Our model is ready.

**10.6 HOW TO SELECT THE VALUE OF KNN ALGORITHM?**

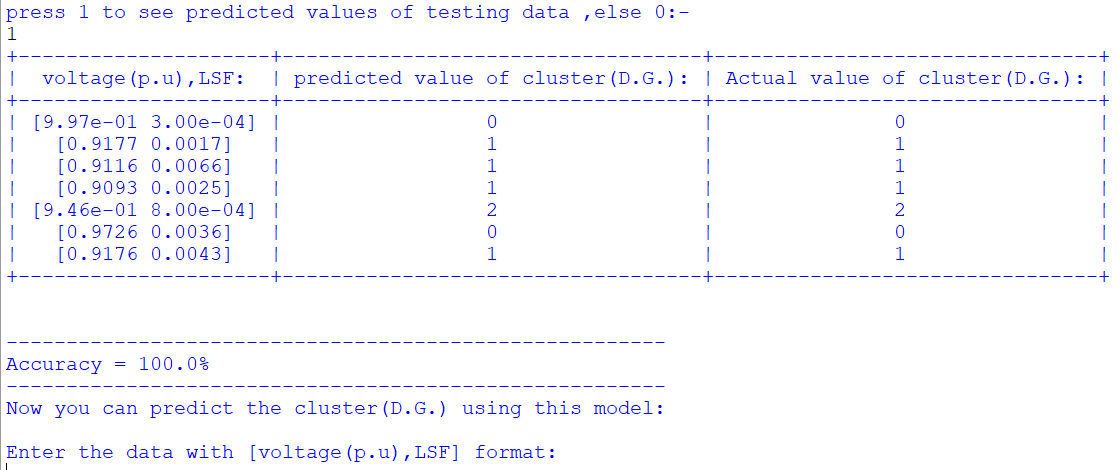
Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. **The most preferred value for K is 5.**
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.
* So, we will use the k=5 in our model.

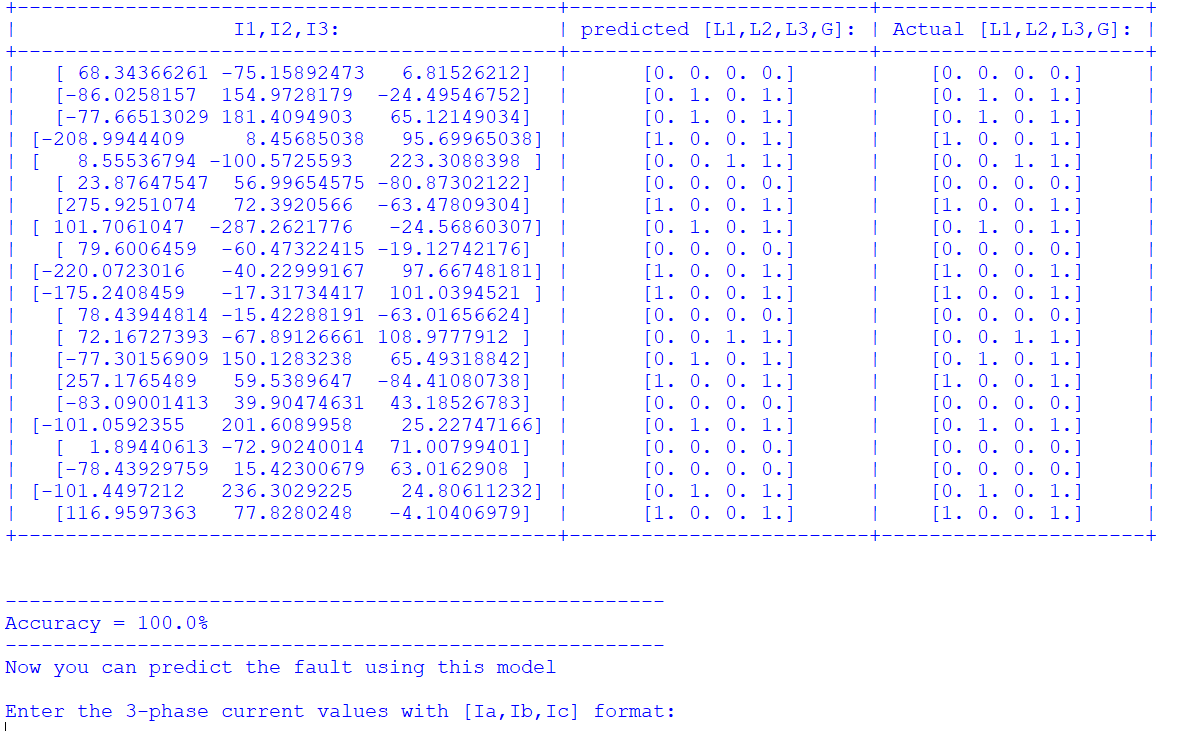


**Fig 10.2: Working of K-Nearest Neighbour Algorithm**

* 1. **OUTPUT OF THE MODEL:**



**Fig. 10.3: Output for** *L****oad Classification Model***



**Fig. 10.4 Output for Fault Classification Model**

**10.8 CODING OF THE MODEL 5:**

import pandas as pd

import numpy as np

import os

from sklearn.model\_selection import train\_test\_split

from matplotlib import pyplot as plt

from prettytable import PrettyTable as PT

def csvfiles():

list\_csv=[]

current\_folder=os.getcwd()

files=os.listdir(current\_folder)

for file in files:

if file.split(".")[-1]=="csv":

list\_csv.append(file)

return(list\_csv)

def Table(x\_test,y\_test,y\_predict):

table=PT()

table.field\_names=["I1,I2,I3:","predicted [L1,L2,L3,G]:","Actual [L1,L2,L3,G]:"]

##**convert array to list**

x\_test\_list=list(x\_test)

y\_test\_list=list(y\_test)

y\_predict\_list=list(y\_predict)

for i in range(len(x\_test)):

table.add\_row([x\_test\_list[i],y\_predict\_list[i],y\_test\_list[i]])

print(table)

def main(user\_name):

print(f"Hello {user\_name}\n\nwelcome to Power prediction system:-")

print("Press ENTER key to proceed")

input()

csv\_files=csvfiles() ##**this is the list of csv files.**

print("choose a csv file for preparing model:-\n")

for position,name in enumerate(csv\_files):

print(f"{position}---->{name}")

model\_csv=csv\_files[int(input("Enter the index of the file:"))]

df=pd.read\_csv(model\_csv)

#Data preprocessing to improve the model accuracy.

arrays=df.iloc[:,:].values #numpy.ndarray

newl=[]

for row in arrays:

if(all(row[-4:]==0) or any(row[1:4]>85) ):

newl.append(row)

arr=np.array(newl)

df= pd.DataFrame(arr)

model\_input=df.iloc[:,[1,2,3]].values #<class 'numpy.ndarray'> #change the index of row and column for new csv file.

fault\_output=df.iloc[:,[4,5,6,7]].values

p=float(input("Enter percentage value for training data:\n") ) #percentage of total data

test\_data=1-(p/100)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(model\_input,fault\_output,test\_size=test\_data) #in which args must be nD array

**### Here x\_train,x\_test,y\_train,y\_test are 2D array.###**

print("\nModel creation in progression\n")

from sklearn.neighbors import KNeighborsClassifier

classifier\_obj = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2) #where p=2 is equivalent to the Euclidean distance

classifier\_obj.fit(x\_train,y\_train) #args are arrays in sklearn..

print("KNN Model has been created.")**# It means clusters have been fixed.** (c1,c2,c3)

print("would you like to test this model using testing data?")

ans=int(input("if yes-->Enter 1:"))

if ans==1:

y\_predict=classifier\_obj.predict(x\_test)

print("press 1 to see predicted values of testing data ,else 0:-")

if int(input())==1:

Table(x\_test,y\_test,y\_predict)

print("\n")

print("-------------------------------------------------------")

true=0

for i in range(len(y\_predict)):

x=(list(y\_predict)[i])==(list(y\_test)[i])

if (all(x)):

true+=1

accuracy=(true/len(y\_predict)\*100)

print(f"Accuracy = {round(accuracy,2)}%")

print("-------------------------------------------------------")

print("Now you can predict the fault using this model")

while(True):

try:

print("\nEnter the 3-phase current values with [Ia,Ib,Ic] format:")

x\_newdata=eval(input())

print(f"cluster number: {classifier\_obj.predict(np.array([x\_newdata]))}")

x=int(input("\nEnter 0 for exit and 1 for continue:"))

if x==0:

break

except:

print("Error.")

print("Enter the data in given format.")

if \_\_name\_\_=="\_\_main\_\_":

name=input("Enter your name : ")

main(name)

input()

**10.9 FLOWCHART OF THE MODEL 5:**

Start

Select your dataset

Split the data and **select the K value**

Model is ready to predict the fault

Accuracy > 90%?

NO YES

Training data will be given to the model

Stop

Trained model

Testing data will be given to the model

Predicted values and actual values of the testing data

Check the accuracy

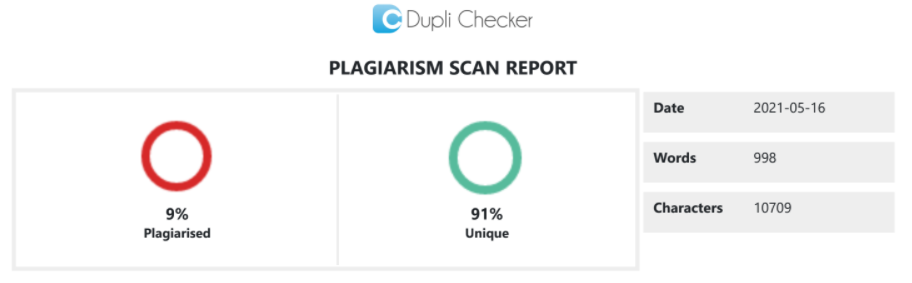
**10.10 CONCLUSION:**

With the help of this model (“K-Nearest Neighbor (KNN) Algorithm”) we can identify the optimal cluster (D.G.) for the new load in power system and we can also differentiate among three phases which phase of a three phase power system is experiencing a fault with specific accuracy. **We got 100% accuracy** as shown in figure 10.4(page: 50).

**REFERENCES:**

* [https://www.javatpoint.com/machine-learning](http://corsera.org/)
* [https://www.coursera.org/learn/machine-learning](http://corsera.org/)
* <https://www.youtube.com/user/krishnaik06> (K-means & Hierarchical clustering ,Regression)
* <https://www.youtube.com/c/sentdex> (KNN classification.)
* <https://www.youtube.com/watch?v=HVXime0nQeI&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=38> (KNN classification)
* <http://irdindia.in/journal_ijraet/pdf/vol8_iss6/6.pdf> (K-means clustering and Hierarchical clustering.)
* <https://springerplus.springeropen.com/track/pdf/10.1186/s40064-015-1080-x.pdf> (KNN classification)

**PLAGIARISM REPORT:**



I have uploaded this report to the google drive:-

<https://drive.google.com/file/d/14I9C3NNwXQzm4bgplGmp85K6F5fao88s/view?usp=drivesdk>