TITLE OF THE PROJECT

A SYSTEMATIC KNOWLEDGE MINING APPROACH TOWARDS MOBILE APPS ANALYSIS AND RATE PREDICTION USING MODIFIED KNN

PROJECT REPORT

Submitted in fulfilment for the J Component of ITA5007 – Data Mining and Business Intelligence

CAL COURSE

in

Master of Computer Applications

by

ARKA SEAL (20MCA0126)

DEV SHARMA (20MCA0129)

RUCHI MANTRI (20MCA0132)

Under the guidance of

Dr. N. DEEPA

SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Winter Semester 2020-21

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
1.	**Introduction**	4
2.	**Literature Survey**	5
3.	**Problem Statement**	9
4.	**Proposed System**	10
5.	**Architecture Diagram**	11
6.	**System design**	
	6.1 Software and Hardware requirements	13
7.	**Module Description**	14
8.	**Implementation**	
	8.1 Source code	23
9.	**RESULTS**	
	9.1. Screenshots	39
10.	**Conclusion**	57
11.	**References**	58

Abstract

K NN is a popular approach of supervised data mining to extract the knowledge of desired output value from given input values. KNN works with the nearest neighbour caused difference values, but sometimes the K^{th} nearest difference value might not be an optimum value as it can be way larger considering the other difference values. This kind of situation may not provide optimum solution(s) especially while working with a huge set of data where there can be a lot of uncertainty in the data ranges to be processed. So we in this project have devised an algorithm updating the nearest neighbour concept of KNN as Modified KNN.

Instead of finding the given K number of nearest neighbours for each input variable Modified KNN will search for the nearest differences up to a certain limit depending on the least difference value. This will reduce the uncertainty and produce a more optimised result each time.

Introduction

Traditional KNN has been a very useful algorithm over the times, I can be used for predicting Categorical data and with a added mean calculation further used for Numerical data prediction that is, whenever we require to find the numerical value of an output variable we just need to find the mean of the probable nearest values of the same variable given values. A graphical representation and error or accuracy calculation may be given to see the correctness of the data.

In our paper we have just simply updated KNN for checking till a limit. This added limit causes the prediction to be more optimized as it reduces the range and more focuses on a centralized and more correct output. A sample dataset would be taken to test our Modified KNN and graphical representation would be given that resembles the working of the said mechanism.

Literature Survey

[1] In this very rapid development, many methods can be used to determine the poverty level. One of them is with the use of the rapid development of E-commerce in Indonesia, which can determine the level of poverty in Indonesia. In this study, we proposed a method to predict the poverty level based on an e-commerce dataset using K-Nearest Neighbour and Information Theoretical Based Feature Selection. Our method is expected to be able to complement the BPS Census and Susenas in predicting poverty levels in an area. Our test results show that our method data can predict the poverty level although there are rooms for improvement in terms of accuracy.

[2] For any classification or prediction problem, machine learning algorithm plays an important role in analysis of data. It enhances the performance capability of classifiers and ease the difficulties. In this research analysis, a prediction model is developed for diabetes disease using K-nearest neighbour's classification algorithm. The model is implemented in Python using PIMA Indian diabetes dataset. The objective of research work is to enhance the prediction capability of KNN classifier with the help of feature selection and normalization of data. A study is also performed to find the optimal number of neighbours on which KNN returns its best result. The F1 score is considered as the main performance metrics for this work.

[3] This model is utilized to distinguish the severity of COVID-19. In HHO-FKNN, the purpose of introducing HHO is to optimize the FKNN's optimal parameters and feature subsets simultaneously. Also, based on actual COVID-19 data, we conducted a comparative experiment between HHO-FKNN and several well-known machine learning algorithms, which result shows that not only the proposed HHO-FKNN can obtain better classification performance and higher stability on the four indexes but also screen out the key features that distinguish severe COVID-19 from mild COVID-19. Therefore, we can conclude that the proposed HHO-FKNN model is expected to become a useful tool for COVID-19 prediction

[4] Multiple features extracted from radar returns in different domains have ability but not enough to solely distinguish radar returns with target from sea clutter. Joint

exploitation of multiple features becomes the key to improve detection performance. In this article, the K-nearest neighbour (KNN) algorithm and anomaly detection idea are cooperated to develop a novel sea-surface target detection method in the feature space spanned by the eight existing salient features. The detection is realized by the anomaly detection followed by a specially designed KNN-based classifier with a controllable false alarm rate. In the anomaly detection, a decision region is determined by the hyper-spherical coverage of the training set of sea clutter that is sufficient and ergodic in the feature space. The KNN-based classifier is designed based on the training sample set of sea clutter and the training sample set of simulated target returns plus sea clutter that is sufficient but non-ergodic, by joint usage of feature weighting, neighbour weighting, and distance weighting.

[5] Mujair fish (Oreochromis Mozambicans) is a popular consumption fish in Indonesia. The fish is a freshwater fish found in rivers, ponds, and lakes, with a salinity of less than 0.05% for the breed. Mujair fish is widely consumed by the public as a cheap and tasty fish that is often found in traditional markets or modern markets. The fish is often sold in a fresh condition (fresh) as well as through the process of freezing (frozen). However, consumers sometimes do not fully know the information about the fish condition. Too long storage process causes the physical changes of the fish into blurry eyes, colours tend to fade, soft fish meat texture, and unpleasant smell. In this research, the process to visualize fresh fish images (suitable for consumption) and not fresh (not suitable for consumption) can be detected using K-Nearest Neighbour (K-NN) using a Smartphone. The purpose of this study is to determine the results of the calculation accuracy using the K-Nearest Neighbour (K-NN) Algorithm with the method of digital image processing of Mujair fish, so consumers can choose whether the fish is suitable for consumption or not, using a smartphone as a visualization that also can be implemented as an additional facility in an online shop.

[6] In seeing and summarizing public opinion towards Nokia's products, this research will develop a website-based application namely YouTube Sentiment Analysis. Application is built using Python and JavaScript with the Flask framework. In building the sentiment analysis model, this research use K-Nearest Neighbours algorithm with value of k=5 and getting an accuracy value of 88.6%.

Application that has been built can carry out processes ranging from collecting automatically using Vader Sentiment, pre-processing comments using NLTK, weighting words using TFIDF, classifying using K-Nearest Neighbors algorithm evaluating algorithms using confusion matrix, to visualizing classification and evaluation results in the form of pie charts, bar charts and confusion matrix.

[7] K-neighbour algorithm is an effective way to classify things according to their characteristics. Novel Coronavirus epidemic development and response situation of various countries around the world can be classified to simplify the difficulty of epidemic prevention and control and provide useful information for them. Based on the data of WHO and the existing research results at home and abroad, this paper analyses the evaluation indexes of national epidemic situation, selects representative countries as training samples, and points out the epidemic situation types of corresponding countries. It aims to provide reference for relevant researchers in epidemic prevention and control and trend prediction through global epidemic classification, and hopes to improve relevant technologies in further research, so as to make the classification more scientific and accurate.

[8] In this paper, we propose a clustering and reclassification method for movie recommendation. We use the improved K-means algorithm to cluster according to the scores of similar users, Firstly, the elbow function is used to estimate the number of clusters, and the elbow method is used to determine the K value. Then, the K-means algorithm of the maximum and minimum distance method is used to select the initial cluster centre, and finally the cluster and cluster centre are obtained. According to the similarity between the test data of the user's rating and user's personal information and the clustering centre, they are divided into the cluster to which they belong, and the sample set in the cluster is used as the training set for K-nearest neighbour classification.

[9] The k-nearest neighbour algorithm (KNN) is one of the most widely used and effective nonparametric classification algorithms. The classification mechanism of this algorithm involves computing the distance between new instance and the other instances. When the dataset contains non-numerical (ordinal and nominal) attributes, the performance of the algorithm can be significantly affected by how this distance is measured. This paper presents a distance measurement method for

improving the performance of KNN. The idea of the proposed method is based on the notion of dynamic distance, which refers to the distance defined between the two values of a non-numerical attribute and depends on the nature of the problem. The determination mechanism of this dynamic distance is formulated in the form of an optimization problem, which is embedded within the structure of KNN and solved using the invasive weed optimization algorithm. The performance of the proposed algorithm is tested on the datasets of the UCI machine learning repository. The results show a minimum of 8% and a maximum of 48.1% improvement in classification accuracy, compared to classic KNN.

[10] Identification of patients at high risk for hospital readmission is of crucial importance for quality health care and cost reduction. Predicting hospital readmissions among diabetic patients has been of great interest to many researchers and health decision makers. We build a prediction model to predict hospital readmission for diabetic patients within 30 days of discharge. The core of the prediction model is a modified k Nearest Neighbour called Hybrid Fuzzy Weighted k Nearest Neighbour algorithm. The prediction is performed on a patient dataset which consists of more than 70,000 patients with 50 attributes. We applied data pre-processing using different techniques in order to handle data imbalance and to fuzzify the data to suit the prediction algorithm. The model so far achieved classification accuracy of 80% compared to other models that only use k Nearest Neighbour.

Problem Statement

Before going directly into the problem lets discuss little about KNN. KNN or K Nearest Neighbours is a supervised data mining algorithm where distance is calculated using a specified formula between train dataset's variables and its respective given variable. And the top K least differences are taken to find the desired variable.

Now, the limitation found is for K least distances it maybe found that the later values maybe way larger than the initial ones, that is a wide range of values maybe found in some difference arrays. To rectify than we are modifying the KNN algorithm to choose the optimum difference values.

This method of Modified KNN is used in predicting the App Ratings of a Google apps dataset. Some App Ratings are given in the dataset which will at as the train dataset and with the help of these we will train our algorithm to predict the values of the required App Ratings of the test dataset.

Proposed System

Modified KNN

Consider some given input variable as X and Y, and a data set having columns x, y and z such that each column elements are x_i, y_i and z_i where i=1, 2, 3… n.

Step 1: Using Euclidian formula to find the distance for each and every row as $\sqrt{((X-x_i)^2 + (Y-y_i)^2)}$ for each row elements.

Step 2: Create a distance array using the distances formed by the above formula as d_i, and chose the least among them (min_dist).

Step 3: Now use the predefined K vale to set a range of values of distances from min_dist that is min_dist + K as a check that if any di value is greater than this then it won't be considered.

Step 4: Now find the mean of these refined d_i values mapped output variable (zi for example).

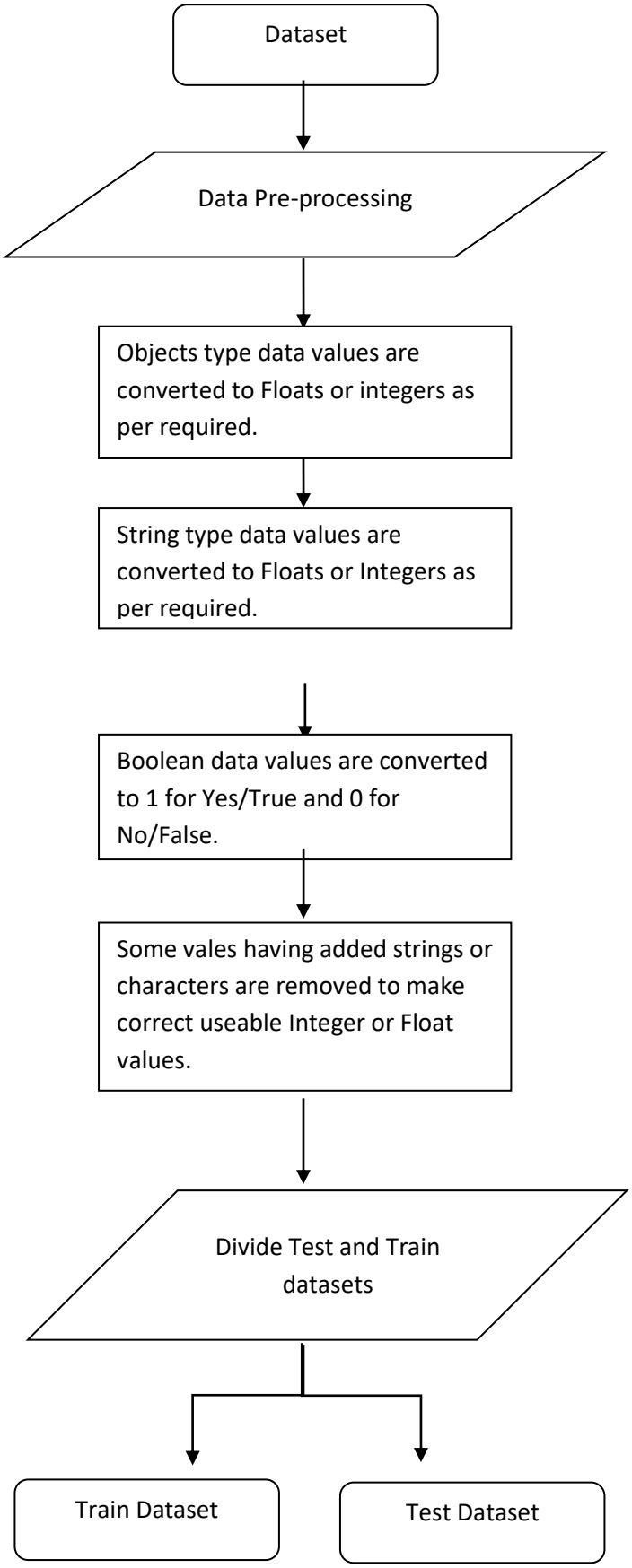
Step 5: This mean value will be the desired output variable's value.

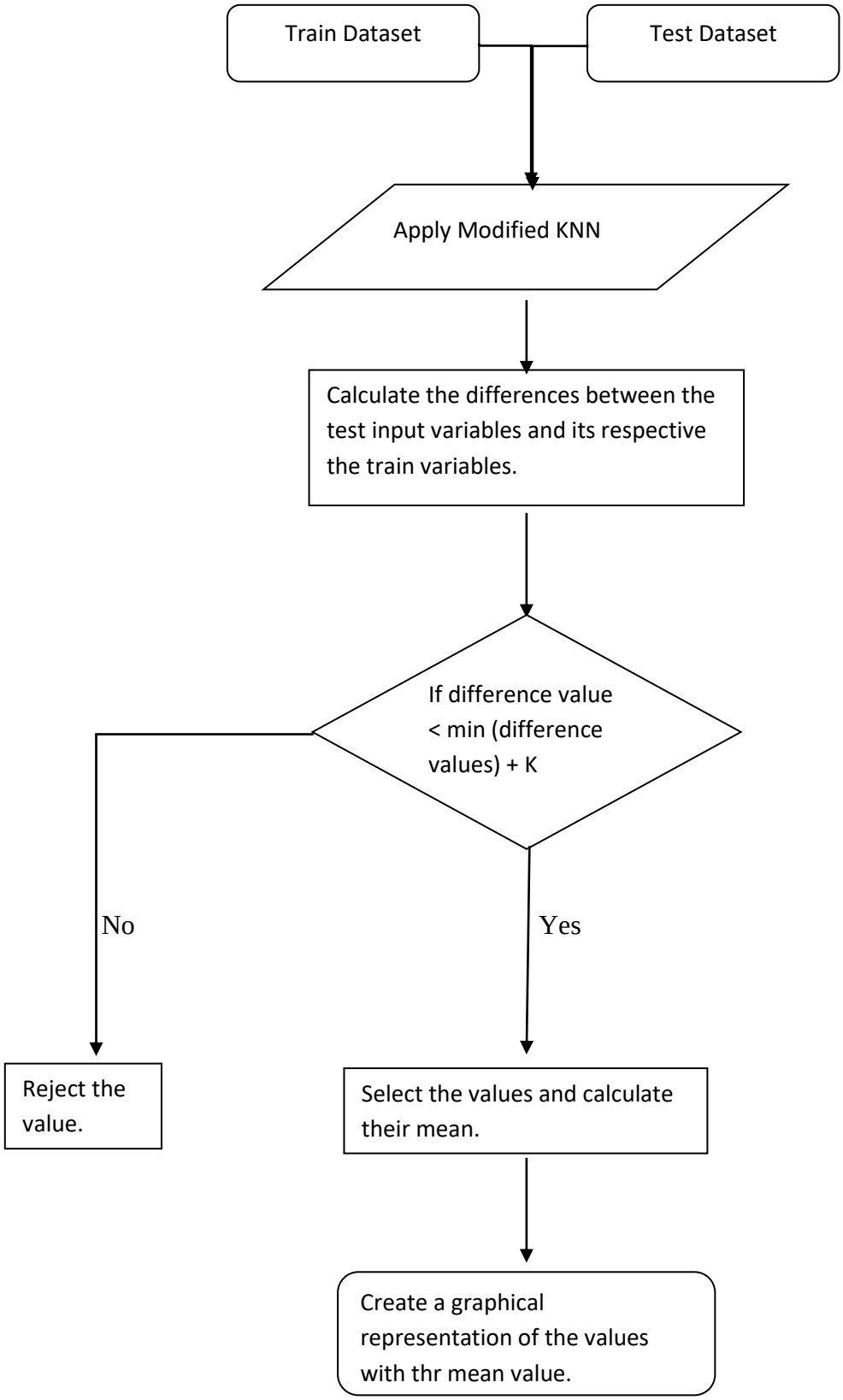
In this project in the above said dataset of Google Apps we have used this Modified KNN technique to find the App Rating of the apps considering other attributes.

We are finding the mean and putting a range to it as this will limit the difference values up to a certain optimum limit of less wider range.

Wrong definition of the K's value may be erroneous and may dissolve the credibility of this modification on traditional KNN.

Architectural Design




Train Dataset
Test Dataset
Apply Modified KNN
Calculate the differences between the test input variables and its respective the train variables.
If difference value < min (difference values) + K
No
Yes
Reject the value.
Select the values and calculate their mean.
Create a graphical representation of the values with thr mean value.

System Design

Hardware requirements:

Processor: 2GHz or more for better data processing

Ethernet Connection or a Wireless Adapter

Hard Drive: 1 TB (enough to store data and intermediate processed outputs plus extra apps Space).

RAM: 8GB (greater the better)

Software Requirements:

Any Operating System with suitable runnable applications and better processing.

Jupyter Notebook (or any such application for data processing)

Module Description

Module 1: Gathering Data

Dataset is being gathered from Kaggle website with 15 columns and 5000 rows. We took the dataset of Google App Store from there with 1048575 rows and 23 columns and work with only 6000 rows and 15 columns for easement in testing and data processing. This data set is further read through our code for processing.

The working-model selected columns are:

- Category
- Rating Content
- Installs
- Minimum Installs
- Maximum Installs
- Free
- Price
- Size
- Released
- Last Updated
- Content Rating
- Ad Supported
- In App Purchases
- Editors Choice
- Rating

All the columns expect ‘Rating’ column are the input variables and the ‘Rating’ column is the required target variable, which is the value which is to be predicted.

Below is the diagram of these columns along with their respective datatypes before pre-processing is done. In the next Module these are converted to more useable and optimised versions of themselves.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5479 entries, 5344 to 4739
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Category          5479 non-null   int64
 1   Rating Count      5469 non-null   float64
 2   Installs          5479 non-null   float64
 3   Minimum Installs  5477 non-null   float64
 4   Maximum Installs  5479 non-null   int64
 5   Free              5479 non-null   int64
 6   Price             5479 non-null   float64
 7   Size              5479 non-null   float64
 8   Released          5479 non-null   int64
 9   Last Updated      5479 non-null   int64
 10  Content Rating    5479 non-null   int64
 11  Ad Supported      5479 non-null   int64
 12  In App Purchases  5479 non-null   int64
 13  Editors Choice    5479 non-null   int64
 14  Rating            5469 non-null   float64
dtypes: float64(6), int64(9)
memory usage: 684.9 KB
```

Figure 1: Google App Dataset from Kaggle information

Module 2: Data Pre-processing

Data Pre-processing is a very important step to optimize the data set as per the decided algorithm to be working with. It requires conversion of unusable or less usable or erroneous data elements to more useable form of data which can be used while implementing the algorithm. Pre-processing is an algorithm dependent step where we need to modify the data elements as per the following mechanism's demand. For our Modified KNN approach we require calculations in 'integer or float' format that's why all the string and other datatypes are converted into 'integers' or 'floats' in this module.

After reading the data the data is pre-processed. The following are the pre-processing stages:

- Conversion of dates such as ‘Released’ and ‘Last Updated’ from ‘object’ type to ‘datetime’ datatype.
- Then sorting the rows with respect to the sorting order of datetime columns.
- Assigning integer values to each such datatime elements for calculation purposes.
- Converting each ‘Category’ object type to its respective Integer value.
- Removal of string with undesired values from the dataset.
- Optimizing ‘Size; column values by removing ‘M’ as the last character and commas from the size lengths. Further converting the String object type to ‘Integer’ datatype.
- Removal of ‘commas’ from Installs and then converting the datatype from ‘object’ to ‘integer’.
- ‘Boolean’ data values converted to ‘Integers’ by converting ‘true’ to ‘1’ and ‘false’ to ‘0’.
- Type converting ‘Content Rating’ i.e. ‘String’ to respective ‘integer’ values conversion.
- Identifying the ‘Rating’ column with empty values. Hence, dividing the dataset into ‘Train’ and ‘Test’ datasets with ‘Test’ dataset having empty ‘Rating’ column.

After this module all the unwanted datatype seen in Module 1 is ben optimized fully for Modified KNN and the resultant dataset can be viewed as follow.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5479 entries, 5344 to 4739
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Category           5479 non-null   int64
 1   Rating Count       5469 non-null   float64
 2   Installs           5479 non-null   float64
 3   Minimum Installs   5477 non-null   float64
 4   Maximum Installs   5479 non-null   int64
 5   Free               5479 non-null   int64
 6   Price              5479 non-null   float64
 7   Size               5479 non-null   float64
 8   Released           5479 non-null   int64
 9   Last Updated       5479 non-null   int64
 10  Content Rating     5479 non-null   int64
 11  Ad Supported       5479 non-null   int64
 12  In App Purchases   5479 non-null   int64
 13  Editors Choice     5479 non-null   int64
 14  Rating             5469 non-null   float64
dtypes: float64(6), int64(9)
memory usage: 684.9 KB
```

Figure 2: Pre-processed dataset information

Further the Test and Train datasets are presented with the row numbers and informations as per the division wit respect to the target variable 'Rating'.

```
train.info()
train
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 5344 to 4705
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Category           5000 non-null   int64
 1   Rating Count       4990 non-null   float64
 2   Installs           5000 non-null   float64
 3   Minimum Installs   4998 non-null   float64
 4   Maximum Installs   5000 non-null   int64
 5   Free               5000 non-null   int64
 6   Price              5000 non-null   float64
 7   Size               5000 non-null   float64
 8   Released           5000 non-null   int64
 9   Last Updated       5000 non-null   int64
 10  Content Rating     5000 non-null   int64
 11  Ad Supported       5000 non-null   int64
 12  In App Purchases   5000 non-null   int64
 13  Editors Choice     5000 non-null   int64
 14  Rating             4990 non-null   float64
dtypes: float64(6), int64(9)
memory usage: 625.0 KB
```

Figure 3: Train dataset information

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 478 entries, 5167 to 4739
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Category          478 non-null    int64
 1   Rating Count      478 non-null    float64
 2   Installs          478 non-null    float64
 3   Minimum Installs  478 non-null    float64
 4   Maximum Installs  478 non-null    int64
 5   Free              478 non-null    int64
 6   Price             478 non-null    float64
 7   Size              478 non-null    float64
 8   Released          478 non-null    int64
 9   Last Updated      478 non-null    int64
 10  Content Rating    478 non-null    int64
 11  Ad Supported      478 non-null    int64
 12  In App Purchases  478 non-null    int64
 13  Editors Choice    478 non-null    int64
 14  Rating            478 non-null    int64
dtypes: float64(5), int64(10)
memory usage: 59.8 KB
```

Figure 4: Test dataset information

Module 3: Developing Algorithm and Implementation

As per defined in the 'Proposed System' section the Modified KNN Algorithm is implemented.

Modified KNN is devised based on the basic rules that,

- there can be some discrepancies while working with a predefined number of difference array values to work with as some later values maybe be way larger than initial ones
- to rectify this least value is selected from the differences and only the difference values who are less than or equal to min(differences) + K are

considered. (here K is the predefined value to set a range depending on the difference values)

Here, difference of each row elements is computed using Euclidian formula.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

That is done by implementation of lists in python code. And then the mean of the Rating values mapped to the rows of the selected difference values caused by row elements is computed.

This mean value is presented as the desired 'Rating' output value. Here, we have used the Train dataset to find the difference values for each Test dataset row individually.

```
test
```

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Released	Last Updated	Content Rating	Ad Supported	In App Purchases	Editors Choice	Rating
5167	4	1579.0	5000000.0	50000.0	84014	1	0.00	71.0	5002	5002	3	1	1	0	4.2
4333	44	109.0	100000.0	1000.0	4383	0	2.99	5.7	5003	5003	1	0	0	0	4.3
3793	10	424.0	1000000.0	10000.0	35167	1	0.00	55.0	5004	5004	1	1	0	0	4.3
5767	26	702.0	1000000.0	10000.0	13282	0	1.49	48.0	5005	5005	1	0	0	0	4.8
5831	47	613.0	10000000.0	100000.0	111467	1	0.00	8.5	5006	5006	1	1	1	0	4.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4722	26	555.0	1000000.0	10000.0	12828	0	1.99	288.0	5475	5475	1	0	0	0	4.7
2281	21	0.0	10000.0	100.0	318	1	0.00	523.0	5476	5476	1	1	0	0	1.5
4740	30	576.0	10000000.0	100000.0	121116	1	0.00	273.0	5477	5477	1	0	0	0	3.4
4741	30	9758.0	50000000.0	500000.0	802217	1	0.00	76.0	5478	5478	1	0	0	0	4.7
4739	5	35.0	100000.0	1000.0	4491	1	0.00	337.0	5479	5479	1	0	0	0	3.9

478 rows × 15 columns

Figure 5: Test Dataset after Modified KNN implementation

Module 4: Graphical Representation

In this Module a graphical visualization is presented for each outcome 'Rating' value in Test Dataset edged to the respective 'Rating' values of the Train dataset from where it is averaged to.

This is done to make the user aware of the Modified KNN process working in a more visual manner and to ease in the understanding of the algorithm. We have calculated mean using traditional average formula and notified it in the graph with 'Rating' + alphabets in 'red nodes' to signify the Train Dataset elements used to find mean and 'blue nodes' to signify resultant Test 'Rating' data element as outcome.

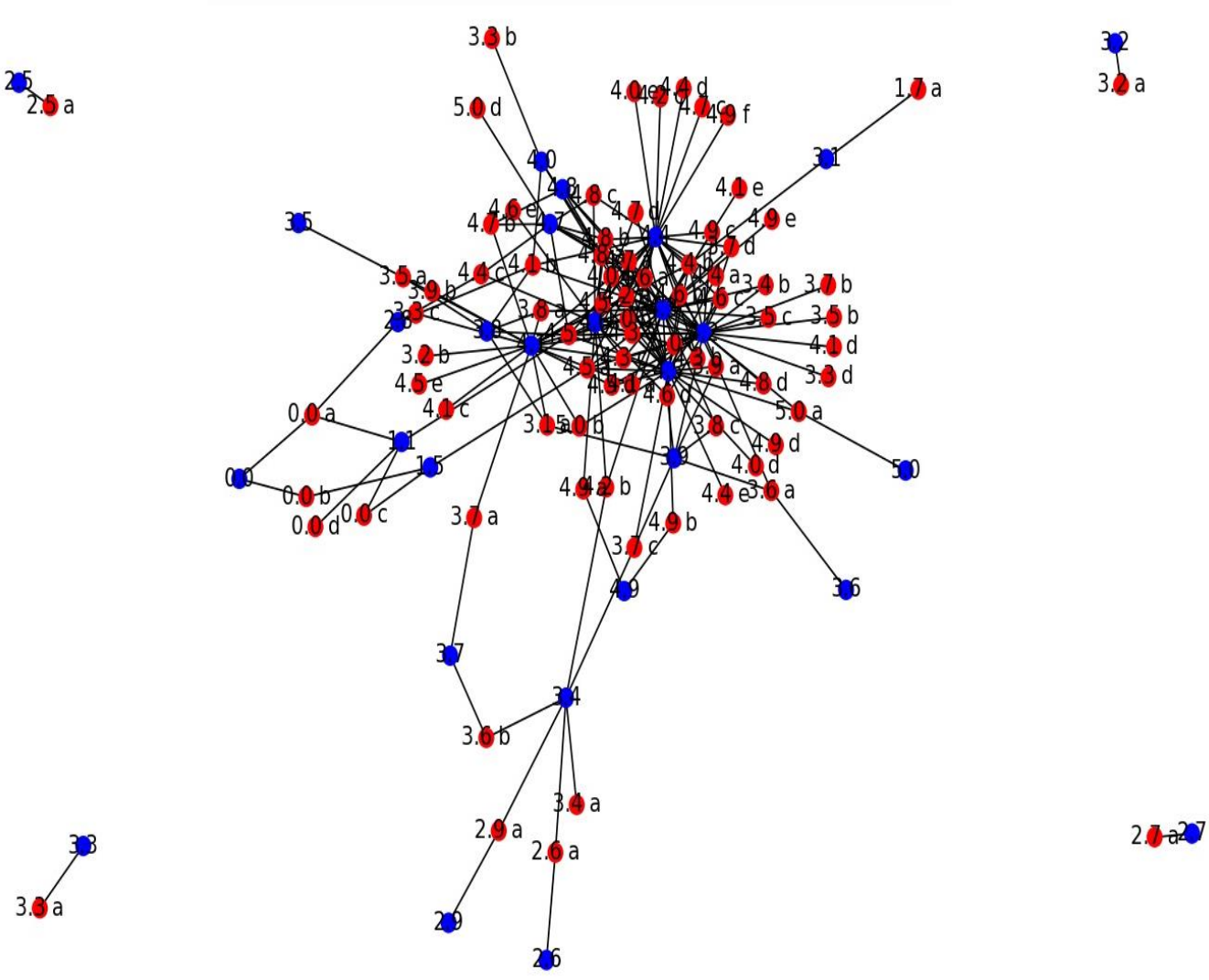


Figure 6: Blue Test outcomes connected with red Train respected elements

Module 5: Error Checking and Comparing

The last stage is to compare our modified KNN with traditional KNN to measure the error rate and find out the optimization context if there is any.

We will also check the Error rate in traditional KNN through a graphical representation. The accuracy is calculated manually and verified for both Modified KNN and traditional KNN individually.

[39]:
```
tp = 0
tn = 0
fp = 0
fn = 0

for t,p in zip(encoded_y, y_pred):
   if t == p:
       if p == 1:
           tp += 1
       else:
           tn += 1
   else:
       if p == 1:
           fn += 1
       else:
           fp += 1

accuracy = (tp + tn) / (tp + tn + fp + fn)
#precision = tp / (tp + fp)
print(accuracy)
```

```
0.061243144424131625
```

[40]:
```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(encoded_y, y_pred))
```

```
Accuracy: 0.061243144424131625
```

Figure 7: Accuracy of traditional KNN

```
tp = 0
tn = 0
fp = 0
fn = 0

for t,p in zip(train, test):
    if t == p:
        if p == 1:
            tp += 1
        else:
            tn += 1
    else:
        if p == 1:
            fn += 1
        else:
            fp += 1

accuracy = (tp + tn) / (tp + tn + fp + fn)
#precision = tp / (tp + fp)
print(accuracy)#modified KNN
```

```
1.0
```

Figure 8: Accuracy of Modified KNN

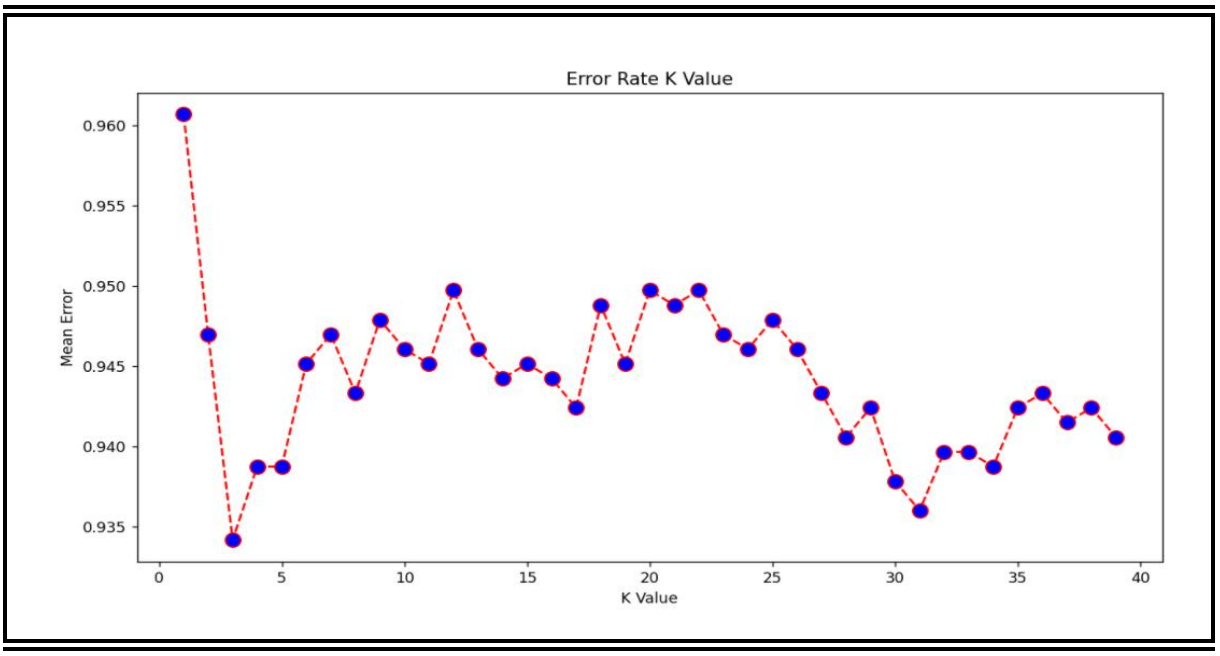


Figure 8: Error rate of traditional KNN

Implementation

Source Code

- **Data Read**

```
import numpy as np

import pandas as pd

import matplotlib as plt

import seaborn as sns

%matplotlib notebook


dataset=pd.read_csv('C:\\Users\\HP\\Desktop\\VIT\\WIN SEM\\Data Mining\\Project\\Google-Playstore.csv')

dataset.head()

dataset.info()

X=dataset.iloc[:6000,[2,4,5,6,7,8,9,11,12,13,16,17,18,20,21,22,3]]
```

- **Data pre-processing 0:**

```
# conversion of date to datetime datatype

X['Released'] = pd.to_datetime(X['Released'])

X['Last Updated'] = pd.to_datetime(X['Last Updated'])

X

X.info()
```

- **Data pre-processing 1:**

```
#conversion of category to integer value
```

```
z=list(X["Category"])
z1=set(z)
print(z1,len(z1))
lz=list(z1)
lz1=[x+1 for x in range(len(lz))]
print(lz1)
lz2=[]
for i in range(len(z)):
    for j in range(len(lz)):
        if(z[i]==lz[j]):
            z[i]=lz1[j]
print(z)
X["Category"]=z
#removal of string undesired with values from the dataset
X=X[~X['Size'].str.contains('Varies with device')]
X
```

- **Data Pre-processing 2:**

```
#removal of 'M' from Size and commas

ll=list(X['Size'])
for i in range(len(ll)):
    ll[i]=ll[i][0:len(ll[i])-1]
for i in range(len(ll)):
```

```
    if(ll[i][1]==","):
        print((ll[i]))
        ll[i]=ll[i][:1]+ll[i][2:]
        print((ll[i]))
ll=[float(i) for i in ll]
print(type(ll))
X['Size']=ll


X
```

- **Data Pre-processing 3:**

```
#removal of 'commas' from Installs
ll1=list(X['Installs'])
for i in range(len(ll1)):
    ll1[i]=str(ll1[i])
    ll1[i]=ll1[i][:len(ll1[i])-1]
    if(ll1[i].find(',')!=-1):
        li=ll1[i].split(',')
        s=''
        for j in li:
            s=s+j
        ll1[i]=float(s)
X['Installs']=ll1
X
```

- **Data Pre-processing 4:**

```
#conversion of true to 1 and false to 0, i.e. boolean to integer
ll2=list(X['Ad Supported'])
for i in range(len(ll2)):
    if(ll2[i]==False):
        ll2[i]=0
    else:
        ll2[i]=1
X['Ad Supported']=ll2
ll3=list(X['In App Purchases'])
for i in range(len(ll3)):
    if(ll3[i]==False):
        ll3[i]=0
    else:
        ll3[i]=1
X['In App Purchases']=ll3

ll3=list(X['Editors Choice'])
for i in range(len(ll3)):
    if(ll3[i]==False):
        ll3[i]=0
    else:
        ll3[i]=1
```

```
X['Editors Choice']=ll3
ll3=list(X['Free'])
for i in range(len(ll3)):
    if(ll3[i]==False):
        ll3[i]=0
    else:
        ll3[i]=1
X['Free']=ll3
X.info()


X
```

- **Data Pre-processing 5:**

```
#sort the table as per 'Last Updated' and 'Released'
details = pd.DataFrame(X)
# sort Dataframe rows based on "Name'
# column in Descending Order
rslt_df = details.sort_values(by = ['Last Updated','Released'], ascending = False)


# show the resultant Dataframe**********
rslt_df
```

- **Data Pre-processing 6:**

```
#conversion of sorted elements as integer values
Y=rslt_df
ll3=list(Y['Released'])
for i in range(len(ll3)):
    ll3[i]=i+1
Y['Released']=ll3
ll3=list(Y['Last Updated'])
for i in range(len(ll3)):
    ll3[i]=i+1
Y['Last Updated']=ll3
#**********************************************************************
**************************************************
Y
Y.info()
```

- **Data Pre-processing 7:**

```
#content rating i.e. String to respective integer values conversion
s=set(list(Y['Content Rating']))
print(s)
l=list(Y['Content Rating'])
for i in range(len(l)):
  if(l[i]=="Everyone"):
    l[i]=1
```

```
  elif(l[i]=="Everyone 10+"):
      l[i]=2
  elif(l[i]=="Teen"):
      l[i]=3
  elif(l[i]=="Mature 17+"):
      l[i]=4
  elif(l[i]=="Adults only 18+"):
      l[i]=5
  elif(l[i]=='Unrated'):
      l[i]=6
Y['Content Rating']=l
Y
```

- **Data Pre-processing 8:**

```
#optimizing
ll1=list(Y['Installs'])
for i in range(len(ll1)):
   if(ll1[i]!='' and ll1[i]!='na'):
       ll1[i]=float(ll1[i])*100
   else:
       ll1[i]=-1
s=set(list(Y['Installs']))
s1=set(ll1)
```

```
Y['Installs']=ll1

Y

Y.info()

Y=Y.iloc[:,[0,1,2,3,4,5,6,7,10,11,12,13,14,15,16]]

Y.info()

#test=Y.loc[Y['Rating']==NaN]

Y.loc[Y['Rating'].isnull(),'value_is_NaN']='Yes'

Y.loc[Y['Rating'].notnull(), 'value_is_NaN'] = 'No'

#test=pd.isnull(Y['Rating'])

s=set(Y['Rating'])

s1=set(Y['value_is_NaN'])

Y =Y[~Y.isin([np.nan, np.inf, -np.inf])]

Y


test=Y.iloc[5001:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

test =test[~test.isin([np.nan, np.inf, -np.inf])]

train=Y.iloc[:5000,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

train =train[~train.isin([np.nan, np.inf, -np.inf])]

test["Rating"]=-1

test.info()


train.info()

train
```

- **Data Processing with Modified KNN implementation:**

```
#Modified KNN algorithm implementation
def MKNN(X,Y,K,msg):
    G=nx.Graph()
    la=[]
    lb=[]
    k=0
    for i in X:
        la.append([])
        for j in X[i]:
            la[k].append(j)
        k+=1
    k=0
    for i in Y:
        lb.append([])
        for j in Y[i]:
            lb[k].append(j)
        k+=1
    n=len(la)
    n1=len(la[0])
    nn=len(lb)
    nn1=len(lb[0])
```

```
d100=[]
for i in range(nn1):
    for j in range(nn):
        if(lb[j][i]==-1.0):
            lb[j][i]=0.0
    d1=[]
    m=0
    for i1 in range(n1):
        d=0
        for j1 in range(n):
            d+=(lb[j1][i]-la[j1][i1])**2
        d1.append(d**(1/2))
    for i in range(len(d1)):
        if str(d1[i])=='nan':
            d1[i]=999999999999999
    m=set(list(d1))
    mi=list(m)
    mi.sort()
    dist=[]
    for i in range(len(d1)):
        if(d1[i]<=mi[0]+K):
            dist.append(i)
    r=0
```

```
    for i in dist:
        r+=la[n-1][i]
    r=r/len(dist)
    G.add_node((round(r,1)))
    d100.append(round(r,1))
    j='a'
    for i in dist:
        G.add_node(str(la[n-1][i])+" "+str(j))
        G.add_edge(round(r,1),str(la[n-1][i])+" "+str(j))
        j = chr(ord(j) + 1)
Y[msg]=d100
print(Y)
plt.figure(3,figsize=(15,15))
llll=list(G.nodes)
color_map = []
for node in llll:
    if isinstance(node,str):
        color_map.append('red')
    else:
        color_map.append('blue')
nx.draw(G, node_color=color_map, with_labels=True, node_size=80)
plt.show()
return Y
```

```
#pass the train test datasets, K-value and testing column or last column
test=MKNN(train,test,100,"Rating")
```

- **Test dataset after implementation:**

```
test
```

- **Train Dataset:**

```
train
```

- **Accuracy checking:**

```
#accuracy
tp = 0
tn = 0
fp = 0
fn = 0
for t,p in zip(train, test):
  if t == p:
     if p == 1:
        tp += 1
     else:
        tn += 1
  else:
     if p == 1:
        fn += 1
     else:
```

```
        fp += 1

accuracy = (tp + tn) / (tp + tn + fp + fn)

print(accuracy)#modified KNN


Y =Y[~Y.isin([np.nan, np.inf, -np.inf]).any(1)]

Y=Y.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

Y


h = Y.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]].values

k = Y.iloc[:, 14].values

h
```

- **Comparing with traditional KNN:**

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(h, k, test_size=0.20)


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X_train)


X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

y_train
```

```
from sklearn import preprocessing
from sklearn import utils
lab_enc = preprocessing.LabelEncoder()
encoded = lab_enc.fit_transform(y_train)
print(utils.multiclass.type_of_target(y_train))
print(utils.multiclass.type_of_target(y_train.astype('int')))
print(utils.multiclass.type_of_target(y_train))
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train,encoded)
y_pred = classifier.predict(X_test)
encoded_y = lab_enc.fit_transform(y_test)
len(encoded)
```

- **Error-calculation:**

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(encoded_y, y_pred))
print(classification_report(encoded_y, y_pred))
error = []
# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, encoded)
    pred_i = knn.predict(X_test)
```

```
    error.append(np.mean(pred_i != encoded_y))
error


from matplotlib import *
import sys
from pylab import *
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
tp = 0
tn = 0
fp = 0
fn = 0
for t,p in zip(encoded_y, y_pred):
   if t == p:
      if p == 1:
         tp += 1
      else:
         tn += 1
   else:
```

```
        if p == 1:
            fn += 1
        else:
            fp += 1
accuracy = (tp + tn) / (tp + tn + fp + fn)
#precision = tp / (tp + fp)
print(accuracy)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(encoded_y, y_pred))
```

RESULTS

Screenshots

In [83]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
%matplotlib notebook



dataset=pd.read_csv('C:\\Users\\HP\\Desktop\\VIT\\WIN SEM\\Data Mining\\Project\\Google-Playstore.csv')


dataset.head()


dataset.info()

X=dataset.iloc[:6000,[2,4,5,6,7,8,9,11,12,13,16,17,18,20,21,22,3]]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 23 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   App Name           1048574 non-null  object
 1   App Id             1048575 non-null  object
 2   Category           1048572 non-null  object
 3   Rating             1042319 non-null  float64
 4   Rating Count       1042319 non-null  float64
 5   Installs           1048424 non-null  object
 6   Minimum Installs   1048424 non-null  float64
 7   Maximum Installs   1048575 non-null  int64
 8   Free               1048575 non-null  bool
 9   Price              1048575 non-null  float64
 10  Currency           1048424 non-null  object
 11  Size               1048575 non-null  object
 12  Minimum Android    1046713 non-null  object
 13  Developer Id       1048574 non-null  object
 14  Developer Website  660774 non-null   object
 15  Developer Email    1048554 non-null  object
 16  Released           1041470 non-null  object
 17  Last Updated       1048575 non-null  object
 18  Content Rating     1048575 non-null  object
 19  Privacy Policy     906730 non-null   object
 20  Ad Supported       1048575 non-null  bool
 21  In App Purchases   1048575 non-null  bool
 22  Editors Choice     1048575 non-null  bool
dtypes: bool(4), float64(4), int64(1), object(14)
memory usage: 156.0+ MB
```

In [84]:
```
# conversion of date to datetime datatype
X['Released'] = pd.to_datetime(X['Released'])
X['Last Updated'] = pd.to_datetime(X['Last Updated'])
X
```

Out[84]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Last Updated	Content Rating	Ad Supported	Pu
0	Communication	2848.0	100,000+	100000.0	351560	True	0.0	2.7M	2.3 and up	Xavier Roche	2013-08-12	2017-05-20	Everyone	False	
1	Strategy	17297.0	1,000,000+	1000000.0	2161778	True	0.0	86M	5.1 and up	Skizze Games	2018-07-19	2020-11-26	Everyone 10+	True	
2	Tools	488639.0	50,000,000+	50000000.0	79304739	True	0.0	5.8M	4.1 and up	TheMauSoft	2016-03-07	2020-10-21	Everyone	True	
3	Business	1224420.0	100,000,000+	100000000.0	163660067	True	0.0	59M	4.4 and up	MobiSystems	2011-12-22	2020-11-23	Everyone	True	
4	Music & Audio	665.0	50,000+	50000.0	73463	True	0.0	29M	5.0 and up	Arthelion92	2016-09-24	2020-11-22	Everyone	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5995	Personalization	152.0	10,000+	10000.0	24186	True	0.0	5.2M	4.2 and up	Jolan Rensen	2018-12-12	2020-11-16	Everyone	False	
5996	Personalization	1690.0	100,000+	100000.0	278401	True	0.0	28M	5.0 and up	Smart Launcher Team	2016-04-19	2020-09-22	Everyone	False	
5997	Personalization	24713.0	1,000,000+	1000000.0	4627346	True	0.0	24M	5.0 and up	Kustom Industries	2015-10-14	2020-11-09	Everyone	True	
5998	Music & Audio	1184.0	100,000+	100000.0	132780	True	0.0	1.6M	4.0.3 and up	Lars Decker	NaT	2017-06-14	Everyone	False	
5999	Music & Audio	283.0	50,000+	50000.0	55626	True	0.0	3.9M	4.0.3 and up	Lars Decker	2011-02-01	2020-10-24	Everyone	False	

6000 rows × 17 columns

In [85]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Category          6000 non-null   object
 1   Rating Count      5950 non-null   float64
 2   Installs          5994 non-null   object
 3   Minimum Installs  5994 non-null   float64
 4   Maximum Installs  6000 non-null   int64
 5   Free              6000 non-null   bool
 6   Price             6000 non-null   float64
 7   Size              6000 non-null   object
 8   Minimum Android   5988 non-null   object
 9   Developer Id      6000 non-null   object
 10  Released          5936 non-null   datetime64[ns]
 11  Last Updated      6000 non-null   datetime64[ns]
 12  Content Rating    6000 non-null   object
 13  Ad Supported      6000 non-null   bool
 14  In App Purchases  6000 non-null   bool
 15  Editors Choice    6000 non-null   bool
 16  Rating            5950 non-null   float64
dtypes: bool(4), datetime64[ns](2), float64(4), int64(1), object(6)
memory usage: 632.9+ KB
```

In [86]:
```
#conversion of category to integer value
z=list(X["Category"])
z1=set(z)
print(z1,len(z1))
lz=list(z1)
lz1=[x+1 for x in range(len(lz))]
print(lz1)
lz2=[]
for i in range(len(z)):
    for j in range(len(lz)):
        if(z[i]==lz[j]):
            z[i]=lz1[j]
print(z)
X["Category"]=z
```

```
{'Art & Design', 'Parenting', 'Beauty', 'Board', 'Communication', 'Strategy', 'Business', 'Racing', 'Action', 'Adventure', 'D
ating', 'Libraries & Demo', 'Social', 'Arcade', 'Photography', 'Auto & Vehicles', 'Music', 'Travel & Local', 'Food & Drink',
'House & Home', 'Sports', 'Video Players & Editors', 'Puzzle', 'Weather', 'Educational', 'Personalization', 'Events', 'Role P
laying', 'Maps & Navigation', 'Entertainment', 'Word', 'Productivity', 'News & Magazines', 'Books & Reference', 'Health & Fit
ness', 'Card', 'Tools', 'Trivia', 'Casual', 'Shopping', 'Music & Audio', 'Comics', 'Casino', 'Lifestyle', 'Simulation', 'Educ
ation', 'Medical', 'Finance'} 48
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 3
4, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48]
[5, 6, 37, 7, 41, 29, 44, 25, 25, 37, 32, 7, 33, 32, 29, 19, 5, 15, 37, 37, 41, 41, 41, 7, 44, 32, 34, 7, 7, 32, 32, 32, 7, 3
7, 40, 40, 32, 4, 37, 37, 37, 44, 7, 32, 32, 32, 7, 7, 21, 37, 7, 7, 40, 37, 32, 37, 47, 32, 7, 37, 32, 32, 32, 32, 32, 37, 3
7, 37, 7, 37, 37, 32, 37, 37, 21, 35, 20, 5, 32, 32, 7, 7, 7, 37, 32, 7, 37, 7, 29, 40, 37, 35, 35, 47, 7, 7, 7, 7, 7, 7, 32,
32, 7, 37, 37, 46, 40, 37, 32, 35, 16, 7, 48, 7, 7, 7, 7, 32, 37, 32, 7, 37, 37, 7, 35, 35, 35, 47, 32, 7, 48, 7, 48, 7, 7, 3
2, 37, 37, 32, 32, 32, 32, 32, 35, 7, 35, 47, 32, 37, 7, 48, 7, 7, 16, 37, 37, 32, 7, 40, 7, 40, 35, 35, 7, 37, 48, 48, 7, 3
7, 37, 46, 46, 40, 7, 2, 48, 7, 35, 35, 35, 35, 47, 20, 5, 37, 46, 46, 37, 18, 46, 48, 48, 35, 35, 35, 35, 35, 35, 35, 35, 4
7, 32, 32, 37, 46, 32, 26, 48, 48, 40, 35, 35, 35, 35, 35, 35, 35, 35, 37, 37, 46, 37, 35, 7, 41, 33, 48, 21, 48, 47, 21, 35,
35, 21, 35, 37, 32, 32, 32, 47, 41, 41, 41, 48, 48, 48, 47, 35, 35, 35, 35, 35, 35, 21, 33, 37, 35, 41, 41, 41, 41, 7, 48, 4
8, 48, 35, 35, 35, 35, 35, 35, 35, 35, 44, 37, 41, 41, 41, 48, 48, 48, 48, 35, 35, 35, 35, 35, 35, 47, 35, 35, 35, 30, 37, 4
1, 48, 48, 48, 48, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 19, 37, 7, 15, 37, 37, 48, 48, 48, 35, 35, 35, 35, 19, 35, 47, 35,
37, 37, 37, 22, 37, 48, 48, 48, 48, 35, 35, 21, 35, 35, 35, 35, 35, 37, 37, 37, 37, 22, 48, 48, 48, 7, 35, 35, 35, 35, 35, 2
1, 21, 35, 37, 37, 37, 37, 44, 22, 48, 48, 47, 19, 35, 48, 35, 37, 21, 21, 35, 37, 37, 37, 37, 32, 44, 37, 26, 48, 48, 48, 3
```

In [87]:
```
#removal of string undesired with values from the dataset
X=X[~X['Size'].str.contains('Varies with device')]
X
```

Out[87]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Last Updated	Content Rating	Ad Supported	In A Purchas
0	5	2848.0	100,000+	100000.0	351560	True	0.0	2.7M	2.3 and up	Xavier Roche	2013-08-12	2017-05-20	Everyone	False	Fal
1	6	17297.0	1,000,000+	1000000.0	2161778	True	0.0	86M	5.1 and up	Skizze Games	2018-07-19	2020-11-26	Everyone 10+	True	Tr
2	37	488639.0	50,000,000+	50000000.0	79304739	True	0.0	5.8M	4.1 and up	TheMauSoft	2016-03-07	2020-10-21	Everyone	True	Fal
3	7	1224420.0	100,000,000+	100000000.0	163660067	True	0.0	59M	4.4 and up	MobiSystems	2011-12-22	2020-11-23	Everyone	True	Tr
4	41	665.0	50,000+	50000.0	73463	True	0.0	29M	5.0 and up	Arthelion92	2016-09-24	2020-11-22	Everyone	False	Fal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5995	26	152.0	10,000+	10000.0	24186	True	0.0	5.2M	4.2 and up	Jolan Rensen	2018-12-12	2020-11-16	Everyone	False	Tr
5996	26	1690.0	100,000+	100000.0	278401	True	0.0	28M	5.0 and up	Smart Launcher Team	2016-04-19	2020-09-22	Everyone	False	Fal
5997	26	24713.0	1,000,000+	1000000.0	4627346	True	0.0	24M	5.0 and up	Kustom Industries	2015-10-14	2020-11-09	Everyone	True	Tr
5998	41	1184.0	100,000+	100000.0	132780	True	0.0	1.6M	4.0.3 and up	Lars Decker	NaT	2017-06-14	Everyone	False	Fal
5999	41	283.0	50,000+	50000.0	55626	True	0.0	3.9M	4.0.3 and up	Lars Decker	2011-02-01	2020-10-24	Everyone	False	Fal

5479 rows × 17 columns

In [89]:
```
#removal of 'M' from Size and commas

ll=list(X['Size'])
for i in range(len(ll)):
    ll[i]=ll[i][0:len(ll[i])-1]
for i in range(len(ll)):
    if(ll[i][1]==","):
        print((ll[i]))
        ll[i]=ll[i][:1]+ll[i][2:]
        print((ll[i]))


ll=[float(i) for i in ll]
print(type(ll))
```

```
1,008
1008
1,004
1004
1,016
1016
1,006
1006
1,008
1008
<class 'list'>
```

In [90]:
```
X['Size']=ll

X
```

```
<ipython-input-90-a657ca5b9ee4>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Size']=ll
```

Out[90]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Up
0	5	2848.0	100,000+	100000.0	351560	True	0.0	2.7	2.3 and up	Xavier Roche	2013-08-12	
1	6	17297.0	1,000,000+	1000000.0	2161778	True	0.0	86.0	5.1 and up	Skizze Games	2018-07-19	202
2	37	488639.0	50,000,000+	50000000.0	79304739	True	0.0	5.8	4.1 and up	TheMauSoft	2016-03-07	
3	7	1224420.0	100,000,000+	100000000.0	163660067	True	0.0	59.0	4.4 and up	MobiSystems	2011-12-22	202
4	41	665.0	50,000+	50000.0	73463	True	0.0	29.0	5.0 and up	Arthelion92	2016-09-24	202
...	...	...	...	...	...	...	...	...	...	...	...	...
5995	26	152.0	10,000+	10000.0	24186	True	0.0	5.2	4.2 and up	Jolan Rensen	2018-12-12	202

In [91]:
```
#removal of 'commas' from Installs

ll1=list(X['Installs'])
for i in range(len(ll1)):
    ll1[i]=str(ll1[i])
    ll1[i]=ll1[i][:len(ll1[i])-1]
    if(ll1[i].find(',')!=-1):
        li=ll1[i].split(',')
        s=''
        for j in li:
            s=s+j

        ll1[i]=float(s)



X['Installs']=ll1
X
```

```
<ipython-input-91-20146ce04a75>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
rsus-a-copy
  X['Installs']=ll1
```

Out[91]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Last Updated	Content Rating	A Supporte
0	5	2848.0	100000.0	100000.0	351560	True	0.0	2.7	2.3 and up	Xavier Roche	2013-08-12	2017-05-20	Everyone	Fals
									5.1 and	Skizze	2018-07-	2020-11-	Everyone	

In [92]:
```
#conversion of true to 1 and false to 0, i.e. boolean to integer

ll2=list(X['Ad Supported'])
for i in range(len(ll2)):
    if(ll2[i]==False):
        ll2[i]=0
    else:
        ll2[i]=1
X['Ad Supported']=ll2

ll3=list(X['In App Purchases'])
for i in range(len(ll3)):
    if(ll3[i]==False):
        ll3[i]=0
    else:
        ll3[i]=1
X['In App Purchases']=ll3

ll3=list(X['Editors Choice'])
for i in range(len(ll3)):
    if(ll3[i]==False):
        ll3[i]=0
    else:
        ll3[i]=1
X['Editors Choice']=ll3

ll3=list(X['Free'])
for i in range(len(ll3)):
    if(ll3[i]==False):
        ll3[i]=0
    else:
        ll3[i]=1
X['Free']=ll3

X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5479 entries, 0 to 5999
```

In [94]:
```
#sort the table as per 'Last Updated' and 'Released'

details = pd.DataFrame(X)

# sort Dataframe rows based on "Name'
# column in Descending Order
rslt_df = details.sort_values(by = ['Last Updated','Released'], ascending = False)

# show the resultant Dataframe*********************************************************************************
rslt_df
```

Out[94]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Last Updated	Content Rating	Ad Supported	In Ap Purchase
5344	23	45.0	5000.0	5000.0	9900	1	0.00	45.0	4.1 and up	HealingJam	2020-10-26	2020-11-28	Everyone	1	
2520	15	311.0	50000.0	50000.0	51755	1	0.00	44.0	4.4 and up	NewG App	2020-08-12	2020-11-28	Everyone	1	
4781	30	4464.0	500000.0	500000.0	702542	1	0.00	31.0	4.3 and up	TLiveGames	2020-07-18	2020-11-28	Teen	1	
606	35	512.0	50000.0	50000.0	54964	1	0.00	19.0	7.0 and up	Team GoodApp	2020-07-14	2020-11-28	Everyone	1	
620	46	21.0	1000.0	1000.0	3682	1	0.00	11.0	5.0 and up	Altaïr Studios	2020-06-08	2020-11-28	Everyone	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4722	26	555.0	10000.0	10000.0	12828	0	1.99	288.0	2.1 and up	IIIIIT	2011-01-22	2013-02-10	Everyone	0	
2281	21	0.0	100	100.0	318	1	0.00	523.0	2.1 and up	SIDEARM Sports	2011-10-21	2012-08-01	Everyone	1	
4740	30	576.0	100000.0	100000.0	121116	1	0.00	273.0	2.0 and up	IIIIIT	2012-01-02	2012-01-03	Everyone	0	
4741	30	9758.0	500000.0	500000.0	802217	1	0.00	76.0	2.0 and up	IIIIIT	2011-08-03	2011-12-29	Everyone	0	

In [95]:
```
#conversion of sorted elements as integer values

Y=rslt_df
ll3=list(Y['Released'])
for i in range(len(ll3)):
        ll3[i]=i+1
Y['Released']=ll3

ll3=list(Y['Last Updated'])
for i in range(len(ll3)):
        ll3[i]=i+1
Y['Last Updated']=ll3
#*************************************************************************************************************
Y
```

Out[95]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Last Updated	Content Rating
5344	23	45.0	5000.0	5000.0	9900	1	0.00	45.0	4.1 and up	HealingJam	1	1	Everyone
2520	15	311.0	50000.0	50000.0	51755	1	0.00	44.0	4.4 and up	NewG App	2	2	Everyone
4781	30	4464.0	500000.0	500000.0	702542	1	0.00	31.0	4.3 and up	TLiveGames	3	3	Teen
606	35	512.0	50000.0	50000.0	54964	1	0.00	19.0	7.0 and up	Team GoodApp	4	4	Everyone
620	46	21.0	1000.0	1000.0	3682	1	0.00	11.0	5.0 and up	Altaïr Studios	5	5	Everyone
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4722	26	555.0	10000.0	10000.0	12828	0	1.99	288.0	2.1 and up	IIIIIT	5475	5475	Everyone
2281	21	0.0	100	100.0	318	1	0.00	523.0	2.1 and up	SIDEARM Sports	5476	5476	Everyone
4740	30	576.0	100000.0	100000.0	121116	1	0.00	273.0	2.0 and	IIIIIT	5477	5477	Everyone

In [97]:
```
#content rating i.e. String to respective integer values conversion
s=set(list(Y['Content Rating']))
print(s)
l=list(Y['Content Rating'])
for i in range(len(l)):
   if(l[i]=="Everyone"):
       l[i]=1
   elif(l[i]=="Everyone 10+"):
       l[i]=2
   elif(l[i]=="Teen"):
       l[i]=3
   elif(l[i]=="Mature 17+"):
       l[i]=4
   elif(l[i]=="Adults only 18+"):
       l[i]=5
   elif(l[i]=='Unrated'):
       l[i]=6
Y['Content Rating']=l
Y
```

```
{'Mature 17+', 'Teen', 'Everyone 10+', 'Everyone'}
```

Out[97]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Minimum Android	Developer Id	Released	Last Updated	Co R
5344	23	45.0	5000.0	5000.0	9900	1	0.00	45.0	4.1 and up	HealingJam	1	1	
2520	15	311.0	50000.0	50000.0	51755	1	0.00	44.0	4.4 and up	NewG App	2	2	
4781	30	4464.0	500000.0	500000.0	702542	1	0.00	31.0	4.3 and up	TLiveGames	3	3	
606	35	512.0	50000.0	50000.0	54964	1	0.00	19.0	7.0 and up	Team GoodApp	4	4	
620	46	21.0	1000.0	1000.0	3682	1	0.00	11.0	5.0 and up	Altaïr Studios	5	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	...

In [100]:
```
Y=Y.iloc[:,[0,1,2,3,4,5,6,7,10,11,12,13,14,15,16]]
```

In [101]:
```
Y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5479 entries, 5344 to 4739
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Category          5479 non-null   int64
 1   Rating Count      5469 non-null   float64
 2   Installs          5479 non-null   float64
 3   Minimum Installs  5477 non-null   float64
 4   Maximum Installs  5479 non-null   int64
 5   Free              5479 non-null   int64
 6   Price             5479 non-null   float64
 7   Size              5479 non-null   float64
 8   Released          5479 non-null   int64
 9   Last Updated      5479 non-null   int64
 10  Content Rating    5479 non-null   int64
 11  Ad Supported      5479 non-null   int64
 12  In App Purchases  5479 non-null   int64
 13  Editors Choice    5479 non-null   int64
 14  Rating            5469 non-null   float64
dtypes: float64(6), int64(9)
memory usage: 684.9 KB
```

In [102]:
```
#test=Y.loc[Y['Rating']==NaN]
Y.loc[Y['Rating'].isnull(),'value_is_NaN']='Yes'
Y.loc[Y['Rating'].notnull(), 'value_is_NaN'] = 'No'
#test=pd.isnull(Y['Rating'])
s=set(Y['Rating'])
s1=set(Y['value_is_NaN'])
Y =Y[~Y.isin([np.nan, np.inf, -np.inf])]
Y
```

```
c:\users\hp\desktop\new folder (2)\lib\site-packages\pandas\core\indexing.py:159
A value is trying to be set on a copy of a slice from a DataFrame.
```

In [103]:
```
test=Y.iloc[5001:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
test =test[~test.isin([np.nan, np.inf, -np.inf])]
train=Y.iloc[:5000,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
train =train[~train.isin([np.nan, np.inf, -np.inf])]
test["Rating"]=-1
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 478 entries, 5167 to 4739
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Category         478 non-null    int64
 1   Rating Count     478 non-null    float64
 2   Installs         478 non-null    float64
 3   Minimum Installs 478 non-null    float64
 4   Maximum Installs 478 non-null    int64
 5   Free             478 non-null    int64
 6   Price            478 non-null    float64
 7   Size             478 non-null    float64
 8   Released         478 non-null    int64
 9   Last Updated     478 non-null    int64
 10  Content Rating   478 non-null    int64
 11  Ad Supported     478 non-null    int64
 12  In App Purchases 478 non-null    int64
 13  Editors Choice   478 non-null    int64
 14  Rating           478 non-null    int64
dtypes: float64(5), int64(10)
memory usage: 59.8 KB
```

In [104]:
```
train.info()
train
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 5344 to 4705
Data columns (total 15 columns):
```

In [221]:
```
#Modified KNN algorithm


def MKNN(X,Y,K,msg):

    G=nx.Graph()

    la=[]
    lb=[]
    k=0
    for i in X:
        #print(i)
        la.append([])
        for j in X[i]:
            la[k].append(j)
        k+=1

    k=0
    for i in Y:
        #print(i)
        lb.append([])
        for j in Y[i]:
            lb[k].append(j)
        k+=1
    #print(lb)

    n=len(la)
    n1=len(la[0])

    nn=len(lb)
    nn1=len(lb[0])

    #if(n==nn):
        #print("AAAAAAAAAAAAAAA",n1,nn1)
    #print(la[n-1][0])
    d100=[]
    for i in range(nn1):
        for j in range(nn):
```

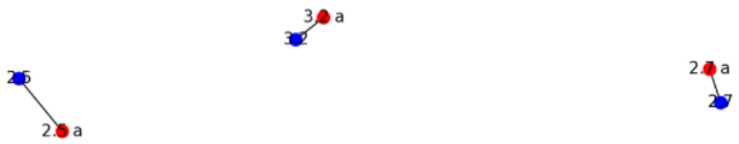
```
for i in range(nn1):
    for j in range(nn):


        if(lb[j][i]==-1.0):
            lb[j][i]=0.0

    d1=[]
    m=0
    for i1 in range(n1):
        d=0
        for j1 in range(n):

            #print(la[j][i],i,j)



            d+=(lb[j1][i]-la[j1][i1])**2

        d1.append(d**(1/2))

    for i in range(len(d1)):
        if str(d1[i])=='nan':
            d1[i]=999999999999999

    m=set(list(d1))

    mi=list(m)
    #m.remove(np.nan)
    mi.sort()
    #print(mi)
    dist=[]
    for i in range(len(d1)):
        if(d1[i]<=mi[0]+K):
            dist.append(i)
    r=0
    #print(dist)

    for i in dist:
```

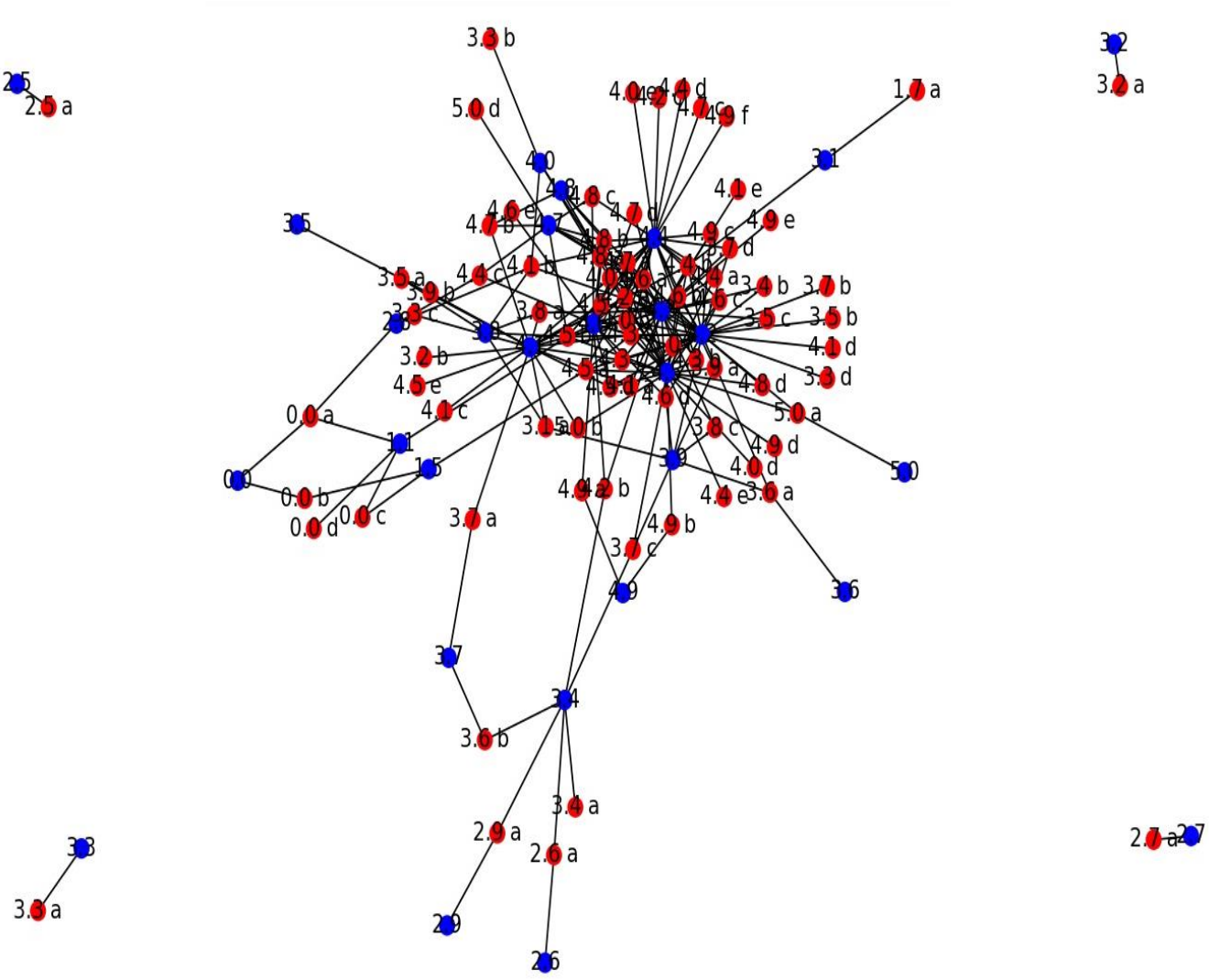
```
        for i in dist:

            r+=la[n-1][i]
            #print(c13[i])

        r=r/len(dist)
        G.add_node((round(r,1)))
        d100.append(round(r,1))
        j='a'
        for i in dist:

            G.add_node(str(la[n-1][i])+" "+str(j))
            G.add_edge(round(r,1),str(la[n-1][i])+" "+str(j))
            #print(r,la[n-1][i])
            j = chr(ord(j) + 1)

    #train.iloc[255:1555,:]
    Y[msg]=d100

    print(Y)
    plt.figure(3,figsize=(15,15))

    llll=list(G.nodes)
    color_map = []
    for node in llll:
        if isinstance(node,str):
            color_map.append('red')
        else:
            color_map.append('blue')
    nx.draw(G, node_color=color_map, with_labels=True, node_size=80)


    plt.show()
    return Y
```

In [223]:
```
#pass the train test datasets, K-value and testing column or last column
test=MKNN(train,test,100,"Rating")
```

```
      Category  Rating Count    Installs  Minimum Installs  Maximum Installs  \
5167        31        1579.0   5000000.0           50000.0             84014
4333        18         109.0    100000.0            1000.0              4383
3793        21         424.0   1000000.0           10000.0             35167
5767        46         702.0   1000000.0           10000.0             13282
5831        24         613.0  10000000.0          100000.0            111467
...        ...           ...         ...               ...               ...
4722        46         555.0   1000000.0           10000.0             12828
2281        22           0.0     10000.0             100.0               318
4740        39         576.0  10000000.0          100000.0            121116
4741        39        9758.0  50000000.0          500000.0            802217
4739        28          35.0    100000.0            1000.0              4491

      Free  Price   Size  Released  Last Updated  Content Rating  \
5167     1   0.00   71.0      5002          5002               3
4333     0   2.99    5.7      5003          5003               1
3793     1   0.00   55.0      5004          5004               1
5767     0   1.49   48.0      5005          5005               1
5831     1   0.00    8.5      5006          5006               1
...    ...    ...    ...       ...           ...             ...
4722     0   1.99  288.0      5475          5475               1
2281     1   0.00  523.0      5476          5476               1
4740     1   0.00  273.0      5477          5477               1
4741     1   0.00   76.0      5478          5478               1
4739     1   0.00  337.0      5479          5479               1

      Ad Supported  In App Purchases  Editors Choice  Rating
5167             1                 1               0     4.2
4333             0                 0               0     4.3
3793             1                 0               0     4.3
5767             0                 0               0     4.8
5831             1                 1               0     4.2
...            ...               ...             ...     ...
4722             0                 0               0     4.7
2281             1                 0               0     1.5
```

```
      Ad Supported  In App Purchases  Editors Choice  Rating
5167             1                 1               0     4.2
4333             0                 0               0     4.3
3793             1                 0               0     4.3
5767             0                 0               0     4.8
5831             1                 1               0     4.2
...            ...               ...             ...     ...
4722             0                 0               0     4.7
2281             1                 0               0     1.5
4740             0                 0               0     3.4
4741             0                 0               0     4.7
4739             0                 0               0     3.9

[478 rows x 15 columns]
```





In [139]:
```
test
```

Out[139]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Released	Last Updated	Content Rating	Ad Supported	In App Purchases	Editors Choice	Rating
5167	4	1579.0	5000000.0	50000.0	84014	1	0.00	71.0	5002	5002	3	1	1	0	4.2
4333	44	109.0	100000.0	1000.0	4383	0	2.99	5.7	5003	5003	1	0	0	0	4.3
3793	10	424.0	1000000.0	10000.0	35167	1	0.00	55.0	5004	5004	1	1	0	0	4.3
5767	26	702.0	1000000.0	10000.0	13282	0	1.49	48.0	5005	5005	1	0	0	0	4.8
5831	47	613.0	10000000.0	100000.0	111467	1	0.00	8.5	5006	5006	1	1	1	0	4.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4722	26	555.0	1000000.0	10000.0	12828	0	1.99	288.0	5475	5475	1	0	0	0	4.7
2281	21	0.0	10000.0	100.0	318	1	0.00	523.0	5476	5476	1	1	0	0	1.5
4740	30	576.0	10000000.0	100000.0	121116	1	0.00	273.0	5477	5477	1	0	0	0	3.4
4741	30	9758.0	50000000.0	500000.0	802217	1	0.00	76.0	5478	5478	1	0	0	0	4.7
4739	5	35.0	100000.0	1000.0	4491	1	0.00	337.0	5479	5479	1	0	0	0	3.9

478 rows × 15 columns

In [140]:
```
train
```

Out[140]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Released	Last Updated	Content Rating	Ad Supported	In App Purchases	Editors Choice	Rating
5344	23	45.0	500000.0	5000.0	9900	1	0.00	45.0	1	1	1	1	1	0	4.6
2520	15	311.0	5000000.0	50000.0	51755	1	0.00	44.0	2	2	1	1	1	0	4.1
4781	30	4464.0	50000000.0	500000.0	702542	1	0.00	31.0	3	3	3	1	1	0	4.5
606	35	512.0	5000000.0	50000.0	54964	1	0.00	19.0	4	4	1	1	1	0	4.4
620	46	21.0	100000.0	1000.0	3682	1	0.00	11.0	5	5	1	1	1	0	4.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1897	10	4449.0	10000000.0	100000.0	275466	1	0.00	24.0	4996	4996	3	1	1	0	4.0
1669	28	274.0	100000.0	1000.0	4045	0	2.99	70.0	4997	4997	3	0	0	0	4.6
2890	48	274.0	1000000.0	10000.0	36541	1	0.00	18.0	4998	4998	1	0	0	0	4.1
1968	44	14.0	100000.0	1000.0	1514	1	0.00	7.1	4999	4999	1	0	0	0	4.9
4705	25	68.0	5000000.0	50000.0	66480	1	0.00	25.0	5000	5000	1	0	0	0	4.3

5000 rows × 15 columns

In [27]:
```
#accuracy
tp = 0
tn = 0
fp = 0
fn = 0

for t,p in zip(train, test):
   if t == p:
       if p == 1:
           tp += 1
       else:
           tn += 1
   else:
       if p == 1:
           fn += 1
       else:
           fp += 1

accuracy = (tp + tn) / (tp + tn + fp + fn)
precision = tp / (tp + fp)
print(accuracy)#modified KNN
```

```
1.0
```

In [28]:
```
Y =Y[~Y.isin([np.nan, np.inf, -np.inf]).any(1)]

Y=Y.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
Y
```

Out[28]:

	Category	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Size	Released	Last Updated	Content Rating	Ad Supported	In App Purchases	Editors Choice	Rating
5344	23	45.0	500000.0	5000.0	9900	1	0.00	45.0	1	1	1	1	1	0	4.6
2520	15	311.0	5000000.0	50000.0	51755	1	0.00	44.0	2	2	1	1	1	0	4.1

In [29]:
```
h = Y.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]].values
k = Y.iloc[:, 14].values
h
```

Out[29]:
```
array([[2.300e+01, 4.500e+01, 5.000e+05, ..., 1.000e+00, 0.000e+00,
        4.600e+00],
       [1.500e+01, 3.110e+02, 5.000e+06, ..., 1.000e+00, 0.000e+00,
        4.100e+00],
       [3.000e+01, 4.464e+03, 5.000e+07, ..., 1.000e+00, 0.000e+00,
        4.500e+00],
       ...,
       [3.000e+01, 5.760e+02, 1.000e+07, ..., 0.000e+00, 0.000e+00,
        4.300e+00],
       [3.000e+01, 9.758e+03, 5.000e+07, ..., 0.000e+00, 0.000e+00,
        4.300e+00],
       [5.000e+00, 3.500e+01, 1.000e+05, ..., 0.000e+00, 0.000e+00,
        3.800e+00]])
```

In [30]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(h, k, test_size=0.20)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
y_train
```

Out[31]:
```
array([3.9, 4.7, 4.5, ..., 4.1, 3. , 4.5])
```

In [ ]:
```
```

In [32]:
```
#comparing with traditional KNN

from sklearn import preprocessing
from sklearn import utils

lab_enc = preprocessing.LabelEncoder()
encoded = lab_enc.fit_transform(y_train)


print(utils.multiclass.type_of_target(y_train))

print(utils.multiclass.type_of_target(y_train.astype('int')))


print(utils.multiclass.type_of_target(y_train))
```

```
continuous
multiclass
continuous
```

In [33]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train,encoded)
```

Out[33]:
```
KNeighborsClassifier()
```

In [34]:
```
y_pred = classifier.predict(X_test)

encoded_y = lab_enc.fit_transform(y_test)

len(encoded)
```

Out[34]:
```
4375
```

In [35]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [35]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(encoded_y, y_pred))
print(classification_report(encoded_y, y_pred))
```

```
[[29  0  0 ...  0  0  0]
 [ 1  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 ...
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]]
```

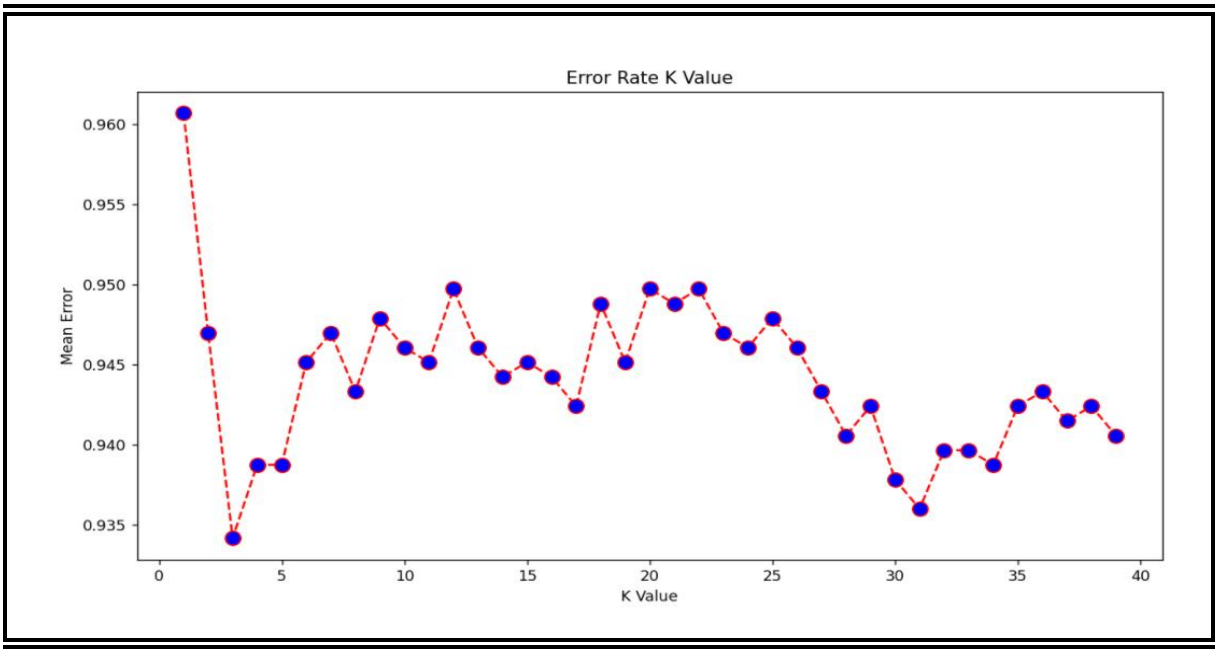
	precision	recall	f1-score	support
0	0.97	1.00	0.98	29
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	4
4	0.00	0.00	0.00	2
5	0.00	0.00	0.00	3
6	0.00	0.00	0.00	3
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	5
9	0.50	0.25	0.33	4
10	0.00	0.00	0.00	3
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	3
13	0.00	0.00	0.00	7
14	0.00	0.00	0.00	6
15	0.00	0.00	0.00	8
16	0.50	0.08	0.14	12
17	0.25	0.06	0.09	18
18	0.14	0.05	0.08	19
19	0.00	0.00	0.00	27
20	0.00	0.00	0.00	26
21	0.00	0.00	0.00	35
22	0.04	0.02	0.03	42
23	0.10	0.05	0.07	39
24	0.17	0.08	0.11	62
28	0.05	0.02	0.02	111
29	0.06	0.06	0.06	94
30	0.03	0.03	0.03	104
31	0.03	0.06	0.04	72
32	0.03	0.11	0.04	27
33	0.01	0.07	0.01	14
34	0.00	0.00	0.00	0
35	0.00	0.00	0.00	0
36	0.00	0.00	0.00	0
37	0.00	0.00	0.00	0
accuracy			0.06	1094
macro avg	0.08	0.05	0.06	1094
weighted avg	0.08	0.06	0.06	1094

In [36]:
```
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, encoded)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != encoded_y))
```

In [37]:
```
error
```

Out[37]:
```
[0.9606946983546618,
 0.946983546617916,
 0.9341864716636198,
 0.9387568555758684,
 0.9387568555758684,
 0.9451553930530164,
 0.946983546617916,
 0.943327239488117,
 0.9478976234003657,
 0.9460694698354661,
 0.9451553930530164,
 0.9497257769652651,
 0.9460694698354661,
 0.9442413162705667,
 0.9451553930530164,
 0.9442413162705667,
 0.9424131627056673,
 0.9488117001828154,
 0.9451553930530164,
 0.9497257769652651,
 0.9488117001828154,
 0.9497257769652651,
 0.946983546617916,
 0.9460694698354661,
 0.9478976234003657,
 0.9460694698354661,
```

In [42]:
```
#error visualization
from matplotlib import *
import sys
from pylab import *
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```



In [39]:
```
tp = 0
tn = 0
fp = 0
fn = 0

for t,p in zip(encoded_y, y_pred):
    if t == p:
        if p == 1:
            tp += 1
        else:
            tn += 1
    else:
        if p == 1:
            fn += 1
        else:
            fp += 1

accuracy = (tp + tn) / (tp + tn + fp + fn)
#precision = tp / (tp + fp)
print(accuracy)
```

```
0.061243144424131625
```

In [40]:
```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(encoded_y, y_pred))
```

```
Accuracy: 0.061243144424131625
```

Conclusion

KNN is a widely accepted algorithm for data prediction, we have simply modified it with some optimization. This optimized version can be replaced with any occurrence of traditional KNN without any restriction. For our approach the user have to properly define the K-value; the K-value is the soul of this whole operation and one cannot utilized the optimized functionality of this algorithm without proper K-value.

Further research and development can be carried for the said approach for less time complexity and more fruitful results.

REFERENCES

1. Aulia, Tiara Fatehana, Wijaya, Dedy Rahman, Hernawati, Elis, Hidayat, Wahyu. (2020). Poverty Level Prediction Based on E-Commerce Data Using K-Nearest Neighbor and Information-Theoretical-Based Feature Selection. *2020 3rd International Conference on Information and Communications Technology (ICOIACT) Information and Communications Technology (ICOIACT).*

2. Gupta, Subhash Chandra, Goel, Noopur. (2020). Enhancement of Performance of K-Nearest Neighbors Classifiers for the Prediction of Diabetes Using Feature Selection Method. *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA).*

3. Ye, H., Wu, P., Zhu, T. Xiao, Z.Viana,. (2021). Diagnosing Coronavirus Disease 2019 (COVID-19): Efficient Harris Hawks-Inspired Fuzzy K-Nearest Neighbor Prediction Methods. *IEEE Access Access, IEEE. 9:17787-17802 2021*

4. Guo, Z., Shui, P.Lash,. (2020). Anomaly Based Sea-Surface Small Target Detection Using K-Nearest Neighbor Classification. *IEEE Transactions on Aerospace and Electronic Systems.*

5. Suhadi, Suhadi, Dina Atika, Prima, Sugiyatno, Sugiyatno, Panogari, Ahmad, Trias Handayanto, Rahmadya, Herlawat. (2020). Mobile-based Fish Quality Detection System Using K-Nearest Neighbors Method. *2020 Fifth International Conference on Informatics and Computing (ICIC)*

6. Irawaty, Irene, Andreswari, Rachmadita, Pramesti, Dita. (2020). Development of Youttube Sentiment Analysis Application using K-Nearest Neighbors (Nokia Case Study). *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*

7. Du, Jinhua. (2021). Global Epidemic Classification Based on K-Nearest Neighbor Algorithm. *2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA)*

8. Cai, Chang, Wang, Li. (2020). Application of improved k-means k-nearest neighbor algorithm in the movie recommendation system. *2020 13th International Symposium on Computational Intelligence and Design (ISCID)*

9. Sadrabadi, Aireza Naser, Znjirchi, Seyed Mahmood, Abadi, Habib Zare Ahmad, Hajimoradi, Ahmad. (2020). An optimized K-Nearest Neighbor algorithm based on Dynamic Distance approach. *2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*

10. Bahanshal, Soha, Kim, Byung. (2020). Hybrid Fuzzy Weighted K-Nearest Neighbor to Predict Hospital Readmission for Diabetic Patients. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*