Assignment No: 03
Classification and Learning [Decision Tree]

CSE-0408 Summer 2021

Joy Sarker
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
joysarker39@gmail.com

***Abstract*—Decision tree classifiers are widely recognized as one of the most well-known approaches for representing data classification in classifiers. The topic of extending a decision tree using existing data has been studied by researchers from diverse domains and backgrounds, including machine learning, pattern recognition, and statistics. Decision tree classifiers have been proposed in a variety of disciplines, including medical disease analysis, text categorization, user smartphone classification, pictures, and many others. The decision trees are discussed in depth in this Assignment. Furthermore, the contents of the Assignment, such as the algorithms/approaches employed, data-sets, and outcomes achieved, are thoroughly analyzed and discussed. Furthermore, all of the methodologies examined and determine the most accurate classifiers.**

***Index Terms*—Artificial Intelligence, Machine Learning, Supervised, Classification, Decision Tree.**

I. Introduction

Technology has advanced significantly in recent years, particularly in the field of Machine Learning (ML), which is effective for minimizing human labor. In the field of artificial intelligence, machine learning (ML) combines statistics and computer science to create algorithms that become more efficient when given relevant data rather than precise instructions. ML is the study of computing methods that are automatically enhanced through experience, in addition to speech recognition, picture identification, text localisation, and so on.

It is classified as a subset of artificial intelligence. ML algorithms generate a model population based on a sample, known as 'training data,' in order to predict or make decisions without being explicitly trained to do so. ML algorithms are used in a wide range of applications, such as email filtering and computer vision, when creating traditional algorithms to implement essential functionalities is difficult or prohibitive. There are numerous applications for machine learning, the most popular of which being predictive data mining.

A decision tree is a tree-based technique in which any path from the root to the leaf node is described by a data separating sequence until a Boolean outcome is reached. It's a hierarchical representation of knowledge relationships with nodes and connections. Nodes represent purposes when using relations to classify data.

In this project, a thorough implementation of the most recent and most effective techniques to decision trees in several fields of machine learning that have been developed by researchers over the last three years is carried out. Making a decision tree is also a part of the process.

II. Literature Review

A decision tree was utilized as a classifier in numerous machine learning and data mining projects. Several recent works on the DT are discussed in this research.

For diabetes mellitus prediction, Zou et al. used decision trees, Random Forest (RF), and neural network techniques. Physical research data for hospitals in Luzhou, China is included in the dataset. There are 14 different characteristics to consider. The training array extracts data from 68994 healthy humans and diabetic patients at random. To reduce dimensionality, they exploited the full significance of minimum Redundancy Maximum Relevance (mRMR) and Principal Component Analysis (PCA). In certain instances, the effects of RF, as opposed to the other classifiers, appeared to be larger. Furthermore, in the Luzhou data collection, 0.8084 is the best result.

Assegie and Nair used the DT classification technique to categorize the handwritten digits in the kaggle digits standard data set and assess the model's accuracy for each digit from 0 to 9. The kaggle features comprise 42,000 rows and 720 columns for machine learning, as well as vector characteristics for digital image pixels. They applied machine learning algorithms to map the classifier's success rate graph in the reality of handwritten digits using a highly efficient language called "python programming." The 83.4 percent accuracy and decision tree classifier had an impact on handwritten number recognition, according to the findings.

III. DECISION TREE ALGORITHM

Systems that produce classifiers are one of the most extensively utilized strategies in data mining. Classification algorithms are capable of handling a large amount of data in data mining. It can be used to make categorical class name assumptions, categorize knowledge based on training sets and

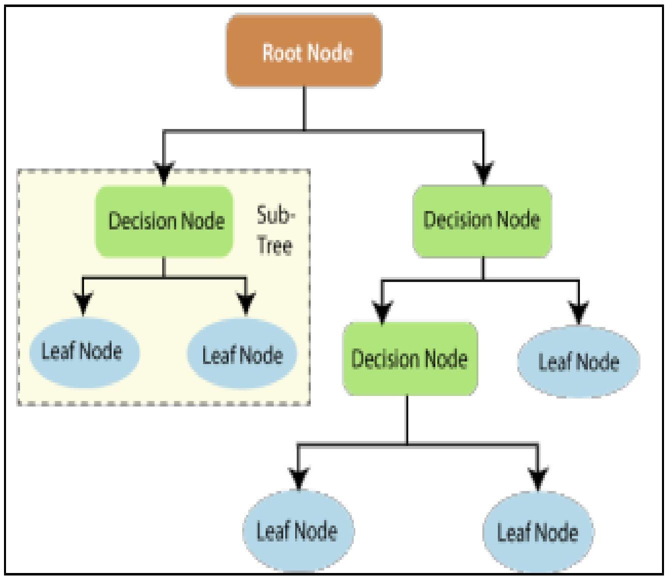


Fig. 1. Decision Tree

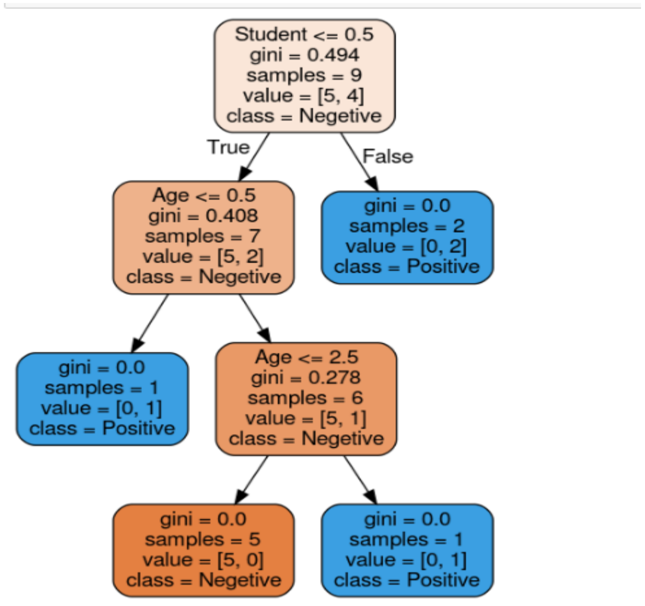


Fig. 2. Example on Decision Tree

class labels, and classify newly available data. Machine learning classification techniques include a variety of algorithms, however this research focuses on the decision tree algorithm in particular. The structure of DT is depicted in Figure 1.

Decision trees are a strong tool that may be utilized in a variety of domains, including machine learning, image processing, and pattern recognition. DT is a sequential model that effectively and cohesively connects a series of fundamental tests in which a numeric feature is compared to a threshold value in each test. The numerical weights in the neural network of connections between nodes are far more difficult to construct than the conceptual rules. DT is primarily used for grouping purposes. Furthermore, in Data Mining, DT is an often used classification model. Each tree is made up of its nodes and branches. Each subset defines a value that the node can take, whereas each node represents features in a category to be categorised. Decision trees offer a wide range of applications due to their straightforward analysis and precision across many data types. An example of DT is shown in Figure 2.

Libraries Requirements

- *pandas*
- *sklearn*
- *IPython*
- *matplotlib*

Pandas is used to take input data sets, **sklearn** is used to develop and train our models, as well as **IPython** and **matplotlib** are used to visualize our decision trees graphically.

IV. Conclusion

This assignment is based on a graphic representation of a decision tree. A data-set is given for the training and visualization of this decision tree.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] D. Abdulqader, A. Mohsin Abdulazeez, and D. Zeebaree, “Machine Learning Supervised Algorithms of Gene Selection: A Review,” Apr. 2020.

[2] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Machine learning: a review of classification and combining techniques,” Artif Intell Rev, vol. 26, no. 3, pp. 159–190, Nov. 2006, doi: 10.1007/s10462-007-9052-3.

[3] Anuradha and G. Gupta, “A self explanatory review of decision tree classifiers,” in International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), Jaipur, India, May 2014, pp. 1–7, doi: 10.1109/ICRAIE.2014.6909245.

[4] S. Taneja, C. Gupta, K. Goyal, and D. Gureja, “An enhanced k-nearest neighbor algorithm using information gain and clustering,” in 2014 Fourth International Conference on Advanced Computing & Communication Technologies, 2014, pp. 325–329.

Decision Tree

September 19, 2021

1 Decision Tree

[1]:
```
import pandas as pd
df = pd.read_csv("Income.csv")
df.head()
```

[1]:
```
     Age  Income Student     Credit    effect
0    <30    high      no       fair  negative
1   <=30    high      no  excellent  negative
2  31-40    high      no       fair  positive
3    >40  medium      no       fair  positive
4   <=30    high      no       fair  negative
```

[2]:
```
inputs = df.drop('effect', axis = 'columns')
target = df['effect']
```

[3]:
```
inputs.head()
```

[3]:
```
     Age  Income Student     Credit
0    <30    high      no       fair
1   <=30    high      no  excellent
2  31-40    high      no       fair
3    >40  medium      no       fair
4   <=30    high      no       fair
```

1.1 Label string with random value using LabelEncoder

[4]:
```
from sklearn.preprocessing import LabelEncoder
```

[5]:
```
new_age = LabelEncoder()
new_Income = LabelEncoder()
new_Student = LabelEncoder()
new_Credit = LabelEncoder()
new_d = LabelEncoder()
```

[6]:
```
inputs['age_n'] = new_age.fit_transform(inputs['Age'])
inputs['Income_n'] = new_Income.fit_transform(inputs['Income'])
```

```
inputs['Student_n'] = new_Student.fit_transform(inputs['Student'])
inputs['Credit_n'] = new_Credit.fit_transform(inputs['Credit'])
target = new_d.fit_transform(target)
inputs.head()
```

[6]:
```
     Age  Income Student     Credit  age_n  Income_n  Student_n  Credit_n
0    <30    high      no       fair      1         0          0         1
1   <=30    high      no  excellent      2         0          0         0
2  31-40    high      no       fair      0         0          0         1
3    >40  medium      no       fair      3         2          0         1
4   <=30    high      no       fair      2         0          0         1
```

[7]:
```
inputs_n = inputs.drop(['Age', 'Income', 'Student', 'Credit'], axis = 'columns')
```

[8]:
```
inputs_n.head()
```

[8]:
```
   age_n  Income_n  Student_n  Credit_n
0      1         0          0         1
1      2         0          0         0
2      0         0          0         1
3      3         2          0         1
4      2         0          0         1
```

[9]:
```
from sklearn import tree
```

[10]:
```
model = tree.DecisionTreeClassifier()
```

[11]:
```
model.fit(inputs_n, target)
```

[11]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

[12]:
```
model.score(inputs_n,target)
inputs.head()
```

[12]:
```
     Age  Income Student     Credit  age_n  Income_n  Student_n  Credit_n
0    <30    high      no       fair      1         0          0         1
1   <=30    high      no  excellent      2         0          0         0
2  31-40    high      no       fair      0         0          0         1
3    >40  medium      no       fair      3         2          0         1
4   <=30    high      no       fair      2         0          0         1
```

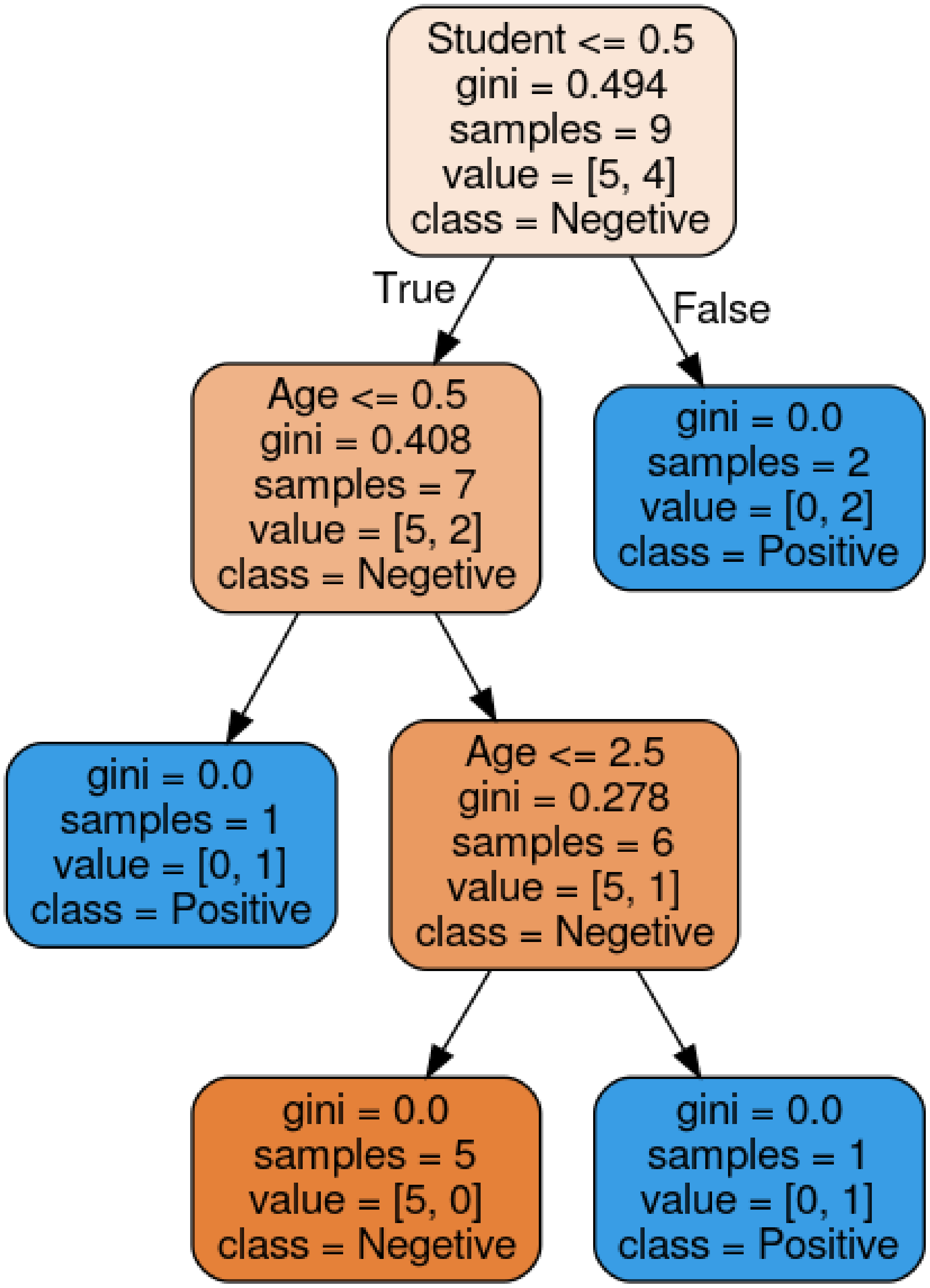
[13]:
```
model.predict([[2,2,0,0]])
```

```
[13]: array([0])
```

1.2 Genarating Decision Tree Visualization

```
[14]: from sklearn import metrics,model_selection
      from IPython.display import Image, display
      import matplotlib.pyplot as plt,pydotplus
```

```
[15]: ddata=tree.export_graphviz(model,out_file=None,filled=True,rounded=True,
                                 feature_names=['Age','Income','Student','Credit'],
                                 class_names=['Negetive','Positive'])
      graph=pydotplus.graph_from_dot_data(ddata)
      display(Image(graph.create_png()))
```


Student <= 0.5
gini = 0.494
samples = 9
value = [5, 4]
class = Negetive
True
False
Age <= 0.5
gini = 0.408
samples = 7
value = [5, 2]
class = Negetive
gini = 0.0
samples = 2
value = [0, 2]
class = Positive
gini = 0.0
samples = 1
value = [0, 1]
class = Positive
Age <= 2.5
gini = 0.278
samples = 6
value = [5, 1]
class = Negetive
gini = 0.0
samples = 5
value = [5, 0]
class = Negetive
gini = 0.0
samples = 1
value = [0, 1]
class = Positive

```
[ ]:
```

Assignment No: 04 Classification and Learning [K- Nearest Neighbors]

CSE-0408 Summer 2021

Joy Sarker
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
joysarker39@gmail.com

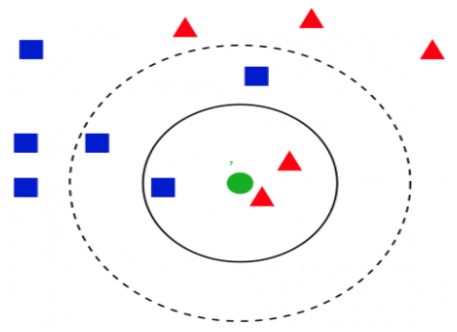


Fig. 1. KNN

***Abstract*—K-Nearest Neighbors method is one of methods used for classification which calculate a value to find out the closest in distance. In this Assignment we are going to implement K-Nearest Neighbors using Jupyter Notebook.**

***Index Terms*—Machine Learning, Supervised, Classification, K nearest neighbors.**

I. Introduction

In the four years of my data science career, I have built more than 80% classification models and just 15-20% regression models. These ratios can be more or less generalized throughout the industry. The reason behind this bias towards classification models is that most analytical problems involve making a decision.

In this assignment, we will implement another widely used machine learning classification technique called K-nearest neighbors (KNN). Our focus will be primarily on how does the algorithm work and how does the input parameter affects the output/prediction.

II. KNN ALGORITHM

We can implement a KNN model by following the below steps:

1) Load the data
2) Initialise the value of k
3) For getting the predicted class, iterate from 1 to total number of training data points
   a) Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
   b) Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
   c) Sort the calculated distances in ascending order based on distance values
   d) Get top k rows from the sorted array
   e) Get the most frequent class of these rows
   f) Return the predicted class

Libraries Requirements

- *pandas*
- *sklearn*
- *matplotlib*

Pandas is used to take input data sets, **sklearn** is used to develop and train our models, as well as **matplotlib** are used to visualize our K-Nearest Neighbors accuracy graphically.

III. Conclusion

This assignment is based on a graphic representation of a KNN model accuracy. A data-set is given for the training and visualization of this KNN model accuracy.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] Solichin, A. (2019, September). Comparison of Decision Tree, Naïve Bayes and K-Nearest Neighbors for Predicting Thesis Graduation. In 2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 217-222). IEEE.

KNN

September 19, 2021

1 Kth Nearest Neighbors

[1]:
```
import pandas as pd  # dataframe
import matplotlib.pyplot as plt #represtation
```

[2]:
```
dataset = pd.read_csv("Income.csv")
dataset.head()
```

[2]:
```
     Age  Income Student     Credit    effect
0    <30    high      no       fair  negative
1   <=30    high      no  excellent  negative
2  31-40    high      no       fair  positive
3    >40  medium      no       fair  positive
4   <=30    high      no       fair  negative
```

[3]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Age      9 non-null      object
 1   Income   9 non-null      object
 2   Student  9 non-null      object
 3   Credit   9 non-null      object
 4   effect   9 non-null      object
dtypes: object(5)
memory usage: 488.0+ bytes
```

[4]:
```
X = dataset.drop(['effect'], axis = 'columns')
Y = dataset['effect']
```

2 Labeling Data

[5]:
```
from sklearn.preprocessing import LabelEncoder #Labeling string with number
```

[6]:
```
new_age = LabelEncoder()
new_Income = LabelEncoder()
new_Student = LabelEncoder()
new_Credit = LabelEncoder()
new_d = LabelEncoder()
```

[7]:
```
X['age_n'] = new_age.fit_transform(X['Age'])
X['Income_n'] = new_Income.fit_transform(X['Income'])
X['Student_n'] = new_Student.fit_transform(X['Student'])
X['Credit_n'] = new_Credit.fit_transform(X['Credit'])
Y = new_d.fit_transform(Y)
X.head()
```

[7]:
```
     Age  Income Student     Credit  age_n  Income_n  Student_n  Credit_n
0    <30    high      no       fair      1         0          0         1
1   <=30    high      no  excellent      2         0          0         0
2  31-40    high      no       fair      0         0          0         1
3    >40  medium      no       fair      3         2          0         1
4   <=30    high      no       fair      2         0          0         1
```

[8]:
```
X = X.drop(['Age','Income','Student','Credit'], axis = 'columns')
```

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, test_size = .25,␣
↪random_state = 5)
```

3 Training and Testing

[11]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[12]:
```
KNN = KNeighborsClassifier()
```

[13]:
```
KNN.fit(X_train, Y_train)
```

[13]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

[14]:
```
Y_pred = KNN.predict(X_test)
```

4 Classification Accuracy Report

[15]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

[16]:
```
print('Calssification',classification_report(Y_test,Y_pred))
```

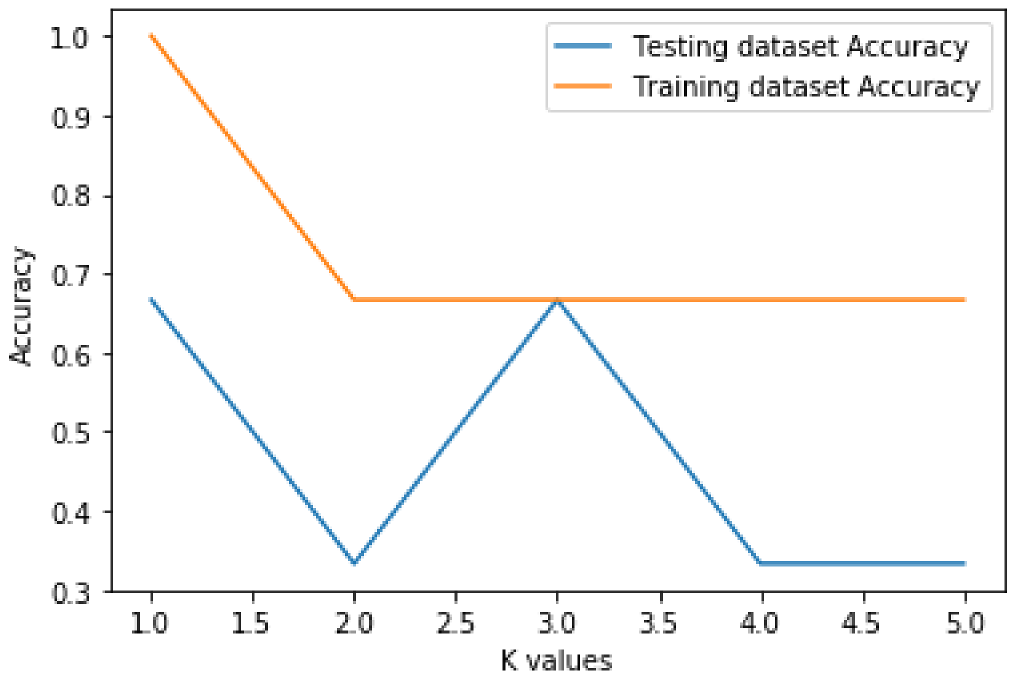
```
Calssification               precision    recall  f1-score   support

           0       0.33      1.00      0.50         1
           1       0.00      0.00      0.00         2

    accuracy                           0.33         3
   macro avg       0.17      0.50      0.25         3
weighted avg       0.11      0.33      0.17         3


/home/darkzero/anaconda3/lib/python3.7/site-
packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

5 Accuracy with different value of K

[17]:
```
train_accuracy = []
test_accuracy = []
```

[18]:
```
for k in range(5):
    KNN = KNeighborsClassifier(n_neighbors=k+1)
    KNN.fit(X_train, Y_train)
    train_accuracy.append(KNN.score(X_train, Y_train))
    test_accuracy.append(KNN.score(X_test, Y_test))
```

[19]:
```
plt.plot([1,2,3,4,5], test_accuracy, label = 'Testing dataset Accuracy')
plt.plot([1,2,3,4,5], train_accuracy, label = 'Training dataset Accuracy')
plt.legend()
plt.xlabel('K values')
plt.ylabel('Accuracy')
plt.show()
```


Testing dataset Accuracy
Training dataset Accuracy
Accuracy
K values
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

```
[ ]:
```