CHAPTER - I

Introduction and Research Methodology

1.1 INTRODUCTION:

The study of glass classification problem was motivated by criminological investigation. In several different types and chemical compositions, glass is available. It can usually be found in windows or doors of a house, in kitchen utensils, in vehicles or in cars. Glass characteristics, particularly the refractive index, depend on the composition and treatment of the glass. The typical glass contains oxides of Si, Mg, Ba, Na and other oxides. In forensic investigation, glass fragments are investigated in order to determine whether the glass fragments obtained from an individual belong to window or non-window glass. Several procedures, based on compositions and refractive index (RI), are available to identify the glass types. In many cases it is difficult to get a well-defined boundary separating the window from non-window glass just looking at the composition and RI, as the composition and RI of window and non-window based glass overlap to a certain extent.

A wide range of classification or machine learning techniques applied to glass identification dataset have included Linear Regression, Decision Tree, Support Vector machine, Naive Bayes,and k- nearest neighbor (kNN). In order to investigate the relative performance of the classification methods.

In this project we are going to use Decision Tree And K- Nearest Neighbour algorithm to design classification models and comparing these models to suggest one.

1.2 Statement of the problem:

A frequent casework requirement is the comparison of glass from a crime scene with glass particles found to be associated with a suspect. Such glass particles are often exceedingly small. It is important to identify and compare these small glass fragments that may be significant in a forensic context. They are facing a problem to classify single fragments of glass based on their main components while investigating.

1.3 Objectives of the study:

a. To predict the Type of Glass based on various parameters.
b. To select the best machine learning model for prediction.

1.4 Need for the study:

The purpose of the project is to understand the concepts of Decision Tree and K-Nearest Neighbour algorithm by practical implementation on the data.

1.5 Scope of the study:

This project is targeted towards classification of Glass to help in investigation & same can be applied where multiclass classification is required.

1.6 Research methodology:

For the research **Secondary** data is collected.
Secondary data – The data is collected from the website named "UCI Repository"

1.7 Chapter Scheme:

The research student has categorized the present study under the five respective chapters. They are-

Chapter-I – Introduction and research methodology

In this chapter research students have covered all the introductory details of the study. i.e. statement of the problem, objectives of the study, scope of the study, research design, and limitations of the study.

Chapter II – Theoretical Framework of the subject

This chapter covers the basic concept and the subject concept of the study.

Chapter III – Data Analysis and Interpretations

This chapter covers the data visualization and machine learning algorithm of decision tree and KNN model.

Chapter IV – Findings and Suggestions

This chapter is totally based on data analysis and interpretations. Also overall observations have considered and made suggestions to choose one model that suits the above problem.

CHAPTER-II

Theoretical Background of the Subject

2.1 GLASS:

DEFINITION: Glass is an amorphous, hard, brittle, transparent or translucent super cooled liquid of infinite viscosity, having no definite melting point obtained by fusing a mixture of a number of metallic silicates or borates of Sodium, Potassium, Calcium, and Lead.

2.1.2 RI (Refractive index): This is a measure of how much the light is bent, or retracted, as it passes through the glass.

2.1.3 Elemental composition: The elements investigated are usually sodium, magnesium, aluminum, silicon, potassium, calcium, barium and iron.

2.2 CLASSIFICATION:

It is a process of sorting a given set of data into each different class. Classification can be implemented on both kinds of data structured as well as unstructured. Classes are often referred to as labels or targets which hold different classes. For example, classifying different fruits.

In order to do so, we have undergone two machine learning algorithms namely:

2.2.1. Decision Tree:

Decision tree build classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost

decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Terminology related to decision tree:

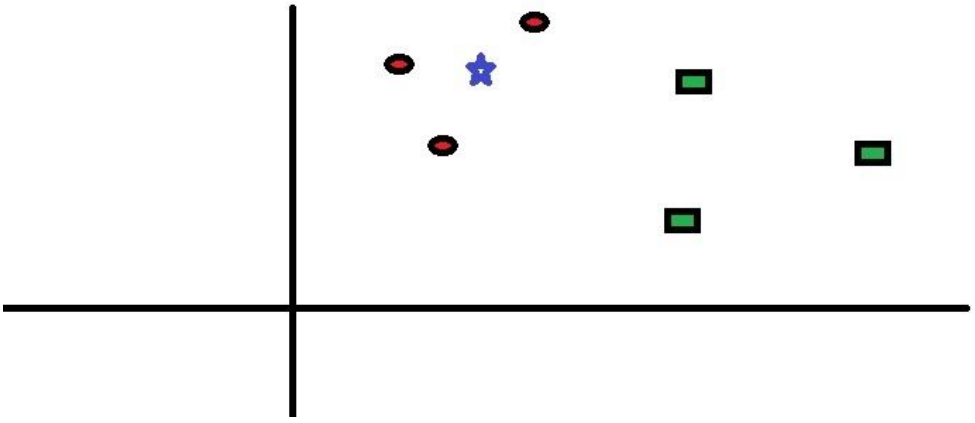
- Branches - Division of the whole tree is called branches.
- Root Node - Represent the whole sample that is further divided.
- Splitting - Division of nodes is called splitting.
- Terminal Node - Node that does not split further is called a terminal node.
- Decision Node - It is a node that also gets further divided into different sub-nodes being a sub node.
- Pruning - Removal of subnodes from a decision node.
- Parent and Child Node - When a node gets divided further then that node is termed as parent node whereas the divided nodes or the sub-nodes are termed as a child node of the parent node.

2.2.2 KNN (K-Nearest Neighbour):

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- The KNN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.
- KNN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using KNN algorithm.
- The KNN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

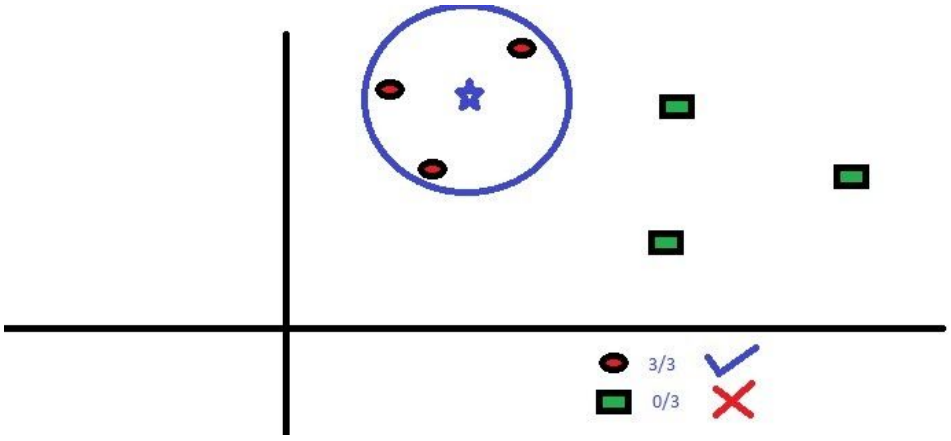
2.2.3 How does the KNN algorithm work?

Let’s take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS) :



You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The KNN algorithm is the nearest neighbor we wish to take the vote from.

Let's say K = 3. Hence, we will now make a circle with BS as the center just as big as to enclose only three data points on the plane. Refer to the following diagram for more details:



The three closest points to BS are all RC. Hence, with a good confidence level, we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

CHAPTER-III

Data Analysis and Interpretation

3.1 About Data:

This is a Glass Identification Data Set from UCI. It contains 10 attributes including id. The response is glass type(discrete 7 values):

Column Position	Attribute Name	Definition	Data Type
1	RI	Refractive Index	float64
2	Na	Sodium (unit measurement: weight percent in corresponding oxide)	float64
3	Mg	Magnesium (unit measurement: weight percent in corresponding oxide)	float64
4	Al	Aluminum (unit measurement: weight percent in corresponding oxide)	float64
5	Si	Silicon (unit measurement: weight percent in corresponding oxide)	float64
6	K	Potassium (unit measurement: weight percent in corresponding oxide)	float64
7	Ca	Calcium (unit measurement: weight percent in corresponding oxide)	float64
8	Ba	Barium (unit measurement: weight percent in corresponding oxide)	float64

9	Fe	Iron (unit measurement: weight percent in corresponding oxide)	float64
10	Type of Glass	Glas Type 1. building_windows_float_processed 2. building_windows_non_float_processed 3. vehicle_windows_float_processed 4. vehicle_windows_non_float_processed 5. containers 6. tableware 7. headlamps	int64

The goal is to clean and preprocess the data also to compare Decision tree and K-Nearest Neighbour. The structure is as follows:

1. **Prepare Problem**
    - Loading Libraries
    - Load and explore the shape of dataset
    - Checking missing values

2. **Summarize data**
    - Descriptive statistics
    - Data Visualization

3. **Prepare data**
    - Split out train test dataset
    - Data transformation

4. **Model Building**

    a. Decision tree classification
        - Fitting the model
        - Checking the training and testing score

- Prediction on testing set
- Checking accuracy
- Optimizing performance
- Checking accuracy after optimizing
- Plotting Decision tree

b. K-Nearest Neighbour

- Fitting the model
- checking the training and testing score
- Prediction on testing set
- Checking accuracy
- Optimizing performance
- Checking accuracy after optimizing

5. **Finalize model and conclusion.**

3.2 Data Cleaning:

Before we use a dataset for actual coding it's important to know is there any missing data. Here's some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

In this dataset there are no missing values as well as glass type in integer and remaining in the float type. Therefore there is no need to clean the dataset.

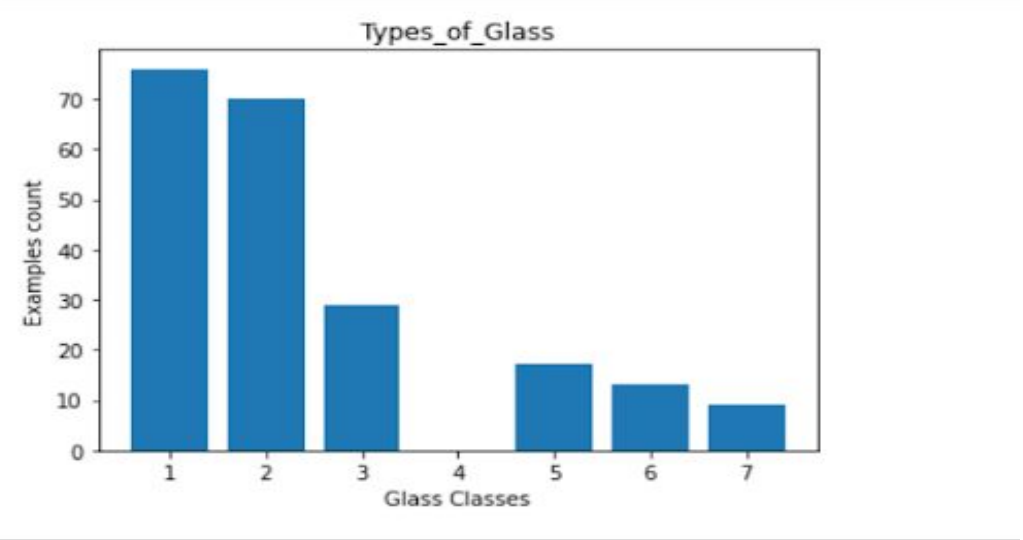
```
: #checking missing values in the data
  data.isnull().sum()

: RI      0
  Na      0
  Mg      0
  Al      0
  Si      0
  K       0
  Ca      0
  Ba      0
  Fe      0
  Type    0
  dtype: int64
```

Before going for the actual model we require to know some necessary information i.e. summary of given data and we get this by using function .**describe()**.

Summary of our dataset as given below:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516523	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

3.3 Data visualization:



Interpretation: Almost 60-67% of glasses are Type 1& 2. Remaining are type 3,5,6 & 7 also no type 4 glass is present in this data.

3.4 Preparing the model:

Before model building we need to split the data frame into two parts i.e dependent and independent variables and then we further use it for validation dataset using train_test_split function of sklearn model.

In this training set contains 80% of the data and the testing set contains 20% of data.

```
#importing sklearn and train_test_split to create validation set
import sklearn
from sklearn.model_selection import train_test_split
#creating the train and validation set
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, stratify=y, random_state = 70)
```

3.5 Model Building:

3.5.1 Decision Tree Classifier

Predictions on validation set:

Input:

```
: #predictions on validation set
  dt_predict=dt_model.predict(X_test)
```

Output:

```
]: array([1, 2, 7, 1, 2, 7, 2, 2, 1, 3, 7, 7, 1, 1, 6, 1, 1, 2, 6, 1, 1, 1,
          7, 2, 2, 1, 5, 2, 1, 1, 1, 2, 1, 1, 2, 6, 6, 2, 1, 2, 2, 1, 1],
         dtype=int64)
```

Accuracy score of validation set:

```
: from sklearn.metrics import accuracy_score
  accuracy_score(y_test,dt_predict)

: 0.6511627906976745
```

Interpretation: Accuracy of given model is 65% and we need to optimize the performance to improve the model.

Optimizing performance:

There are several methods of optimizing performance of decision tree classifier. For our model we are going to use max depth of tree method.

If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

In general, the deeper you allow your tree to grow, the more complex your model will become because you will have more splits and it captures more information about the data and this is one of the root causes of overfitting in decision trees because your model will fit perfectly for the training data and will not be able to generalize well on test set. So, if your model is overfitting, reducing the number for max_depth is one way to combat overfitting.It is also bad to have a very low depth because your model will underfit. So we need to find the best value for the depth of the tree.

Input:

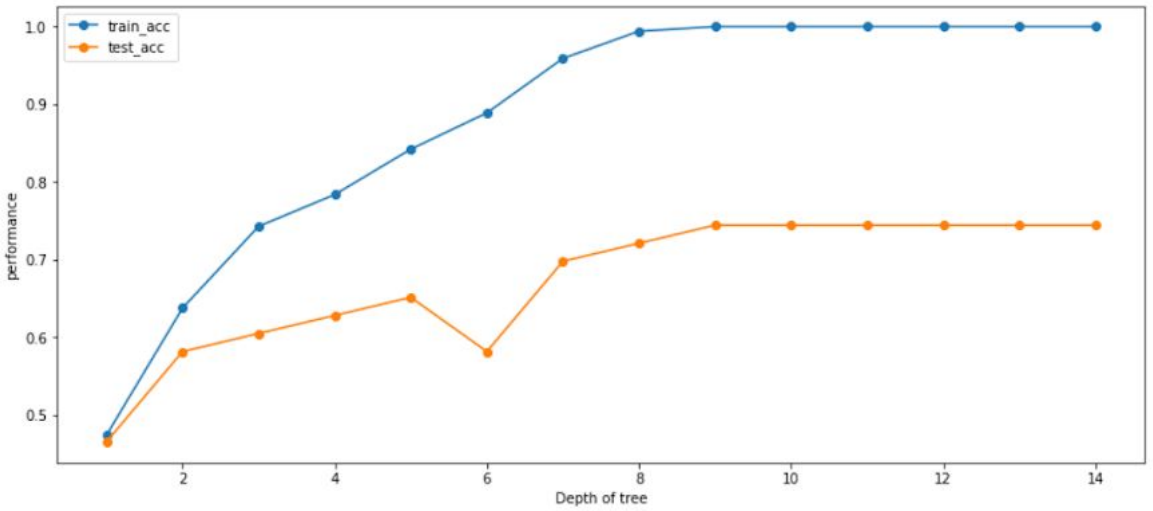
```
train_accuracy = []
validation_accuracy = []
for depth in range(1,15):
    dt_model = DecisionTreeClassifier(max_depth=depth, random_state=6)
    dt_model.fit(X_train, y_train)
    train_accuracy.append(dt_model.score(X_train, y_train))
    validation_accuracy.append(dt_model.score(X_test, y_test))
```

First 5 heads of training and test accuracy for max depth :

	max_depth	train_acc	valid_acc
0	1	0.473684	0.465116
1	2	0.637427	0.581395
2	3	0.742690	0.604651
3	4	0.783626	0.627907
4	5	0.842105	0.651163

Visualisation of training and testing accuracy:

```
<matplotlib.legend.Legend at 0x243b0e0bca0>
```



Interpretation: Max depth keeps on increasing both the training and testing accuracy increasing. In this case at max depth = 9 producing highest testing accuracy and after this accuracy remains the same. hence we are going to use max depth as 9.

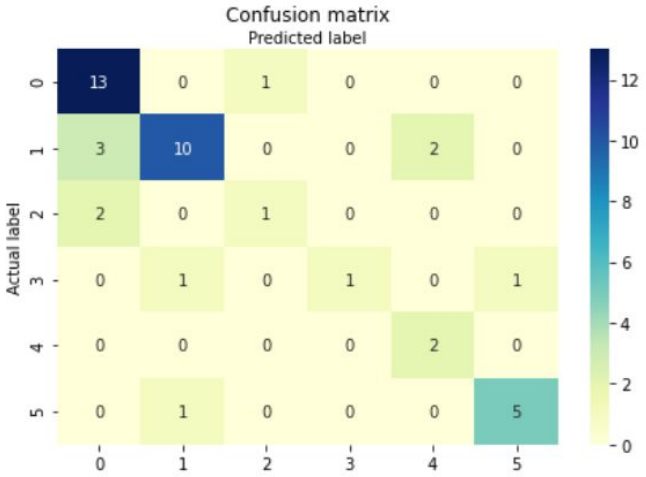
Accuracy after optimizing the performance:

```
accuracy_score(y_test,dt_predict1)
```

```
0.7441860465116279
```

Confusion Matrix:

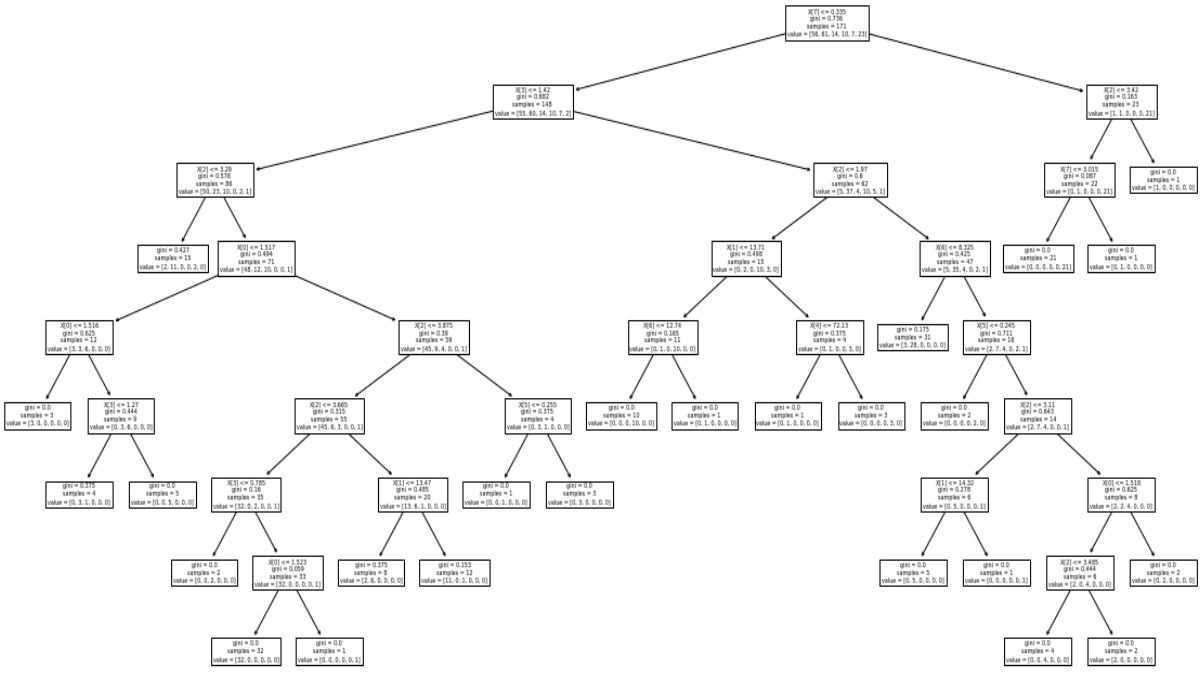
```
Text(0.5, 257.44, 'Predicted label')
```



Interpretation:

- 13 type 1 glass data points were correctly classified by the model also 3 & 2 data points were incorrectly classified as belonging to the type 2 and type 3 by the model respectively.
- 10 type 2 glass data points were correctly classified by the model also 1 data point was incorrectly classified as each belonging to the type 5 and type 7 by the model.
- 1 type 3 glass data points were correctly classified by the model and 1 data point was incorrectly classified as belonging to the type 1 by the model.
- 1 type 5 glass data points were correctly classified by the model.
- 2 type 6 glass data points were correctly classified by the model and 2 data points were incorrectly classified as belonging to the type 2 by the model.
- 5 type 7 glass data points were correctly classified by the model and 1 data point was incorrectly classified as belonging to the type 5 by the model.

Plotting Decision Tree:



3.5.2 K-Nearest Neighbors:

Accuracy score of validation set:

```
: from sklearn.metrics import accuracy_score
  accuracy_score(y_test,predict_type_n)
```

```
: 0.6976744186046512
```

To choose optimal value of K:

There is no one proper method of estimation of K value in KNN. No method is the rule of thumb but you should try considering following suggestions:

1. Square Root Method: Take square root of the number of samples in the training dataset.

2. Cross Validation Method: We should also use cross validation to find out the optimal value of K in KNN.By getting the range of expected k-value, but to get the exact k-value we need to test the model for each and every expected k-value. Start with K=1, run cross validation (5 to 10 fold), measure the accuracy and keep repeating till the results become consistent.

K=1, 2, 3... As K increases, the error usually goes down, then stabilizes, and then raises again. Pick the optimum K at the beginning of the stable zone. This is also called the Elbow Method.

Input:

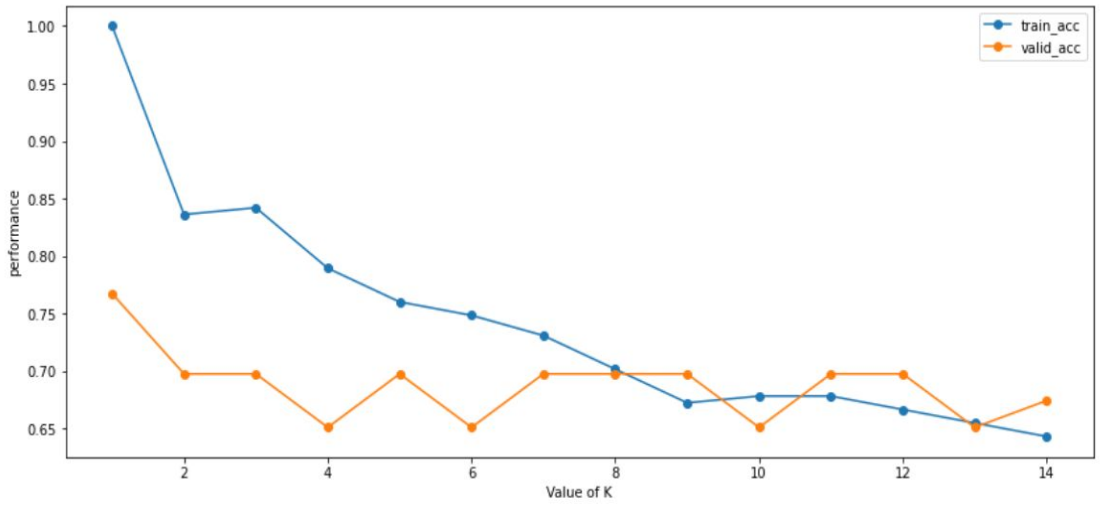
```
: train_accuracy = []
  validation_accuracy = []
  for i in range(1,15):
      clf = KNN(n_neighbors = i)
      clf.fit(X_train, y_train)
      train_accuracy.append(clf.score(X_train, y_train))
      validation_accuracy.append(clf.score(X_test, y_test))
```

First 10 heads of training and test accuracy for K value :

	n_neighbors	train_acc	valid_acc
0	1	1.000000	0.767442
1	2	0.836257	0.697674
2	3	0.842105	0.697674
3	4	0.789474	0.651163
4	5	0.760234	0.697674
5	6	0.748538	0.651163
6	7	0.730994	0.697674
7	8	0.701754	0.697674
8	9	0.672515	0.697674
9	10	0.678363	0.651163

Visualisation of training and testing accuracy:

```
: <matplotlib.legend.Legend at 0x1e3ef0fb670>
```



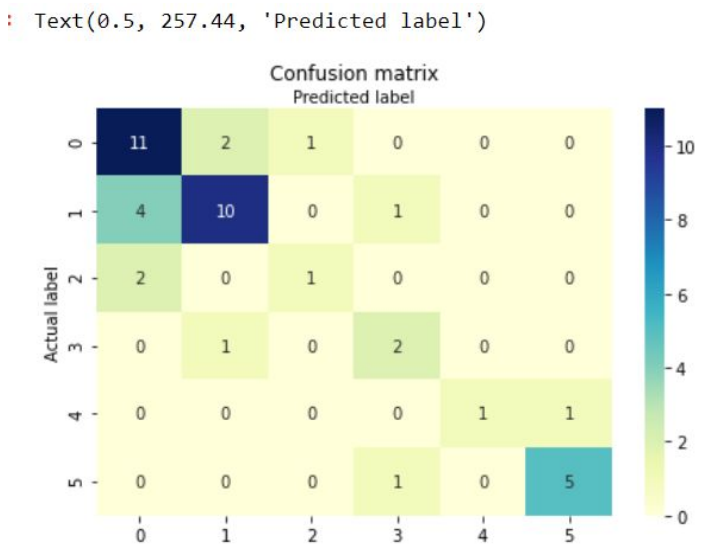
Interpretation: A very low value for K such as K=1 or K=2 can give high accuracy but it can be noisy and lead to the effects of outliers in the model. The above graph accuracy is the same for K value of 2,3,5,7 i.e. 0.697.

Here we use the value of K=8 because it also satisfies the “Square Root Method”.

Accuracy after optimizing the performance:

```
: accuracy_score(y_test,predict_type_n_2)
: 0.6976744186046512
```

Confusion Matrix:



Interpretation:

- 11 type 1 glass data points were correctly classified by the model and 4 & 2 data points were incorrectly classified as belonging to the type 2 and type 3 by the model respectively.
- 10 type 2 glass data points were correctly classified by the model and 2 & 1 data points were incorrectly classified as each belonging to the type 1 and type 5 by the model.
- 1 type 3 glass data points were correctly classified by the model and 1 data point was incorrectly classified as belonging to the type 1 by the model.
- 2 type 5 glass data points were correctly classified by the model and 1 data point were incorrectly classified as each belonging to the type 2 & and type 7 by the model.
- 1 type 6 glass data points were correctly classified by the model.
- 5 type 7 glass data points were correctly classified by the model and 1 data point was incorrectly classified as belonging to the type 6 by the model.

3.6 Final Output:

- Accuracy of Decision tree classifier: 0.744
- Accuracy of K-Nearest Neighbor classifier: 0.6976

- **Report of the DT Model:**

	precision	recall	f1-score	support
1	0.92	0.97	0.94	70
2	0.96	0.93	0.95	76
3	0.93	0.82	0.87	17
5	1.00	0.85	0.92	13
6	0.82	1.00	0.90	9
7	0.97	0.97	0.97	29
accuracy			0.94	214
macro avg	0.93	0.92	0.92	214
weighted avg	0.94	0.94	0.94	214

- **Report of the KNN(K-nearest neighbors) model:**

	precision	recall	f1-score	support
1	0.63	0.86	0.73	70
2	0.75	0.71	0.73	76
3	0.50	0.12	0.19	17
5	0.62	0.62	0.62	13
6	0.80	0.44	0.57	9
7	0.88	0.76	0.81	29
accuracy			0.70	214
macro avg	0.70	0.58	0.61	214
weighted avg	0.70	0.70	0.68	214

CHAPTER -IV

Findings & Conclusion

1. From the Final Report, Accuracy of the DT Model and K-nearest neighbors it can be concluded that :-

- Accuracy of Decision tree classifier: 0.744
- Accuracy of K-Nearest Neighbor classifier: 0.6976

Hence we can interpret that Decision tree classifier is more accurate than K-nearest neighbors classifier.

2. Now in the case of Comprehensive Report of the Decision Tree model & K-nearest neighbors model we can concluded that :-

- F-1 score accuracy for the decision tree model and K-nearest neighbors(K-NN) Model are 0.94 & 0.70 respectively.
- Macro average precision for the decision tree model and K-nearest neighbors(K-NN) Model are 0.93 & 0.70 respectively.
- Weighted average precision the decision tree model and K-nearest neighbors(K-NN) Model are 0.94 & 0.70 respectively.
- Macro average recall for the decision tree model and K-nearest neighbors(K-NN) Model in are 0.92 & 0.58 respectively.
- Weighted average recall for the decision tree model and K-nearest neighbors(K-NN) Model are 0.94 & 0.70 respectively.
- Macro average f1-score for the decision tree model and K-nearest neighbors(K-NN) Model are 0.92 & 0.61 respectively.
- Weighted average f1-score for the decision tree model and K-nearest neighbors (K-NN) Model are 0.94 & 0.68 respectively.

Hence we can interpret that in this case, The Decision Tree model performs much better than that of K-nearest neighbors(K-NN) Model and suggests to use the decision tree model over K-nearest neighbors model.

REFERENCES

1. https://www.kaggle.com

2. Course named “Introduction to Python” on Analytics vidhya.

3. https://analyticsindiamag.com

4. Mashael S. Aldayel, “K-Nearest Neighbor Classification for Glass Identification Problem” Department of Information Technology,King Saud University, Saud Arabia,Riyadh,researchgate Conference Paper, Issue: 10.11.09/ICCSII.2012.6454522.

5. Suchismita Goswami and Edward J. Wegman,“Comparison of Different Classification Methods on Glass Identification for Forensic Research”, Computational and Data Science, George Mason University, Fairfax VA 22030, Journal of Statistical Science and Application, April 2016, Vol. 4, No. 03-04, 65-84.

6. https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019