GLASS CLASSIFICATION

Mohammed Ali Shaik
Department of Computer Science& Artificial Intelligence, SR University, Warangal, Telangana State, India

Moola Harshitha Reddy

Department of Computer Science & Engineering,

SR University, Warangal, Telangana State, India 2203a51430@sru.edu

I. ABSTRACT

Glass type identication other than other readings to be important, despite diverse opposition. Differences with vignettes will help understanding uses, especially in criminal inspections. Yet, the usual classifications may be off and tiresome. Thus, more trustworthy approaches are a necessity. This research throws light on machine learning tactics on online datasets to enhance classification. Specifically, Support Vector Machines, random woods technique, Statistical Analysis, AdaBoost, and enhancing grades classifiers are involved. These tactics will learn from the dataset's features to classify crystal sortings precisely. Of all these tests, the method of random woods has emerged as the most powerful classifier for classification. Its effectiveness outstripped other computational methods, showcasing its knowledge in spotting crystal features. More so, the research examines ML's potential to ease the classification process. These techniques not just enhance the precision and efficiency of glass divisions but also provide glimpses into glass productivity. By adopting these methods, companies can cater to various sectors like mechanics, building, and solar energy demands. Ultimately, this research underlines the significance of innovation to cater to progressing requirements in tech and native industry.

II.INTRODUCTION

Glass enterprise be a a pillar of production in nowaday life, encloosing a large range of products from vehicle windshields to beverage boxes to specialized shielding substances. Its ubiquity across numerous sectors underscores its significance in daily life. However, inside this huge landscape lies a complicated net of compositions, houses, and uhm applications, prompting the want for rigorous investigation. Understanding the particular kind of glass found in diverse contexts, especially forensic analyses, remains a pivotal venture.

The breadth of the glass industry warrants interest, given its pervasive effect on numerous fields. Previous research has delved into numerous factors, such as the chemical composition of different glass types, their optical residences. While present research offers valuable insights, sure questions persist, leaving room for further exploration.Despite the extensive body of studies, there stays an ambiguity surrounding the identity of glass kinds in forensic eventualities. This gap in understanding presents a possibility to cope with the particular demanding situations encountered in correctly discerning the composition of glass samples from crime scenes. By leveraging advanced technology consisting of gadget gaining knowledge, the study tells to bridge this gap among and provide to the advancement of forensic technology practices. In the pursuit of this goal, the present research employs learning algorithms, inclusive of SVM, RF, LR, AdaBoost, and GBoosting classifiers. Through analysis of elemental compositions and refractive indices, we purpose the develope a robust classifications framework.

III. OBJECTIVE OF THE PROJECT

The method incorporates (SVM), (KNN), Random Forest, and Logistic Regression techniques.We furthermore hold Principal Component Analysis (PCA) and different spatiality discount solutions. Python's Pandas package deal is implemented to analyze and observe the dataset's dimensions, seen with visualization using the Matplotlib library.The dataset is then cut up into train and take a look at units with a 5:1 ratio. Following the

cut-up, we conscientiously inspect the dataset's qualities to alleviate problems related to version nice or overfitting.To tackle these dilemmas, we use dimensionality discount techniques such as PCA and XGBoost. After vast measurement discount, the version is trained and evaluated the usage of various class techniques. The option of the most effective method is governed by precision metrics.Through this approach, our enterprise caters to the extraordinary necessities of the car, manufacturing, and solar electricity industries. Furthermore, due to the fact each product and application demand specific sorts of glass, our enterprise area gives a spectrum of alternatives to simplify lifestyles each at home and within the place of work.

IV.LITERATURE REVIEW

In the pursuit of assisting criminal investigations, there was an evaluation of glass categorization flaws. The aim was to efficiently identify leftover glass as potential evidence in criminal cases. The categorization of glass fragments from crime scenes, often small in size, is a common procedural necessity. Assessing and evaluating dose minute glass shard within forensic framework is vital, given d diverse composition and refractive indices of different glass types. Despite its importance, glass categorization has received limited attention in research. Mashael S. And odar researchers have explored algorithms like KNN, but improving model accuracy requires combination of additional algorithms selected from literature. The rarity of a specific type of glass can be determined by comparing retrieved pieces with reference samples of refractive index. The process often relies on computer databases containing refractive index measurement to finding for suitable samples assessed by the laboratory. However, database information may be biased due to the disproportionate representation of glass samples, including those from smashed windows, compared to other sources. Therefore, caution is warranted when interpreting database information in forensic glass analysis.

Paper [1]: Glass-Type Classification Using Support Vector Machines (SVM) and Random Forests

Dis study explores d classification for the varieties of glass based on their chemical compositions. Utilizing data on d refractive index and chemical elements from d UCI Glass Identification Database, study employs Support Vector Machines (SVM) and Random Forest algorithms for classification. Feature significance analysis reveals that certain chemical elements play a crucial role in distinguishing glass types. This is study evaluates classification along with accuracies, precisions and f1score, demonstrating d effectiveness of both SVM and Random Forests in accurately classifying glass types.

Paper [2]: Glass Identification Using Convolutional Neural Networks (CNN)

In dis work, Convolutional Neural Networks (CNN) are used for glass identification based on images of glass samples. The study collects a dataset comprising high-resolution images of different glass types and employs transfer learning techniques using pre-trained CNN models. The research fine-tunes d CNN architectures to classify glass samples accurately. Data techniques are implemented to enhance model with generalization, and research evaluates model performance using classification accuracy and confusion matrices. The results demonstrate effectiveness of CNNs in accurately identifying glass types from images.

Paper [3]: Analysis of Machine Learning Algorithms for Glass Classification

The study conducts a comparison of various ML techniques for glass classification tasks. Algorithm such as (KNN), Decision Trees, Naive Bayes, and Gradient Boosting Machines (GBM) are evaluated using a dataset containing both chemical composition of glass sample. Selection methods applied for identify their most informative feature for classification. This research systematically compares the performance of various algorithms using cross-validation and statistical tests, providing insights into the strength and weakness of each approach for glass type tasks.

Paper [4]: Glass-Type Prediction Using Ensemble Learning

Ensemble learning methods are explored in this study for predicting glass types based on multiple features, including chemical composition, refractive index, and physical properties. The research utilizes ensemble methods such as AdaBoost, Gradient Boosting, and Random Forests to combine the predictions of multiple base classifiers. Feature importance analysis is conducted to understand the contribution of each

feature to ensemble predictions. The study evaluates the ensemble models' performances with accuracy, AUC-ROC, and precision-recall curves, demonstrating superiority of ensemble learning for glass-type prediction tasks.

Paper [5]: Glass Classification Using Deep Learning and Spectroscopic Data

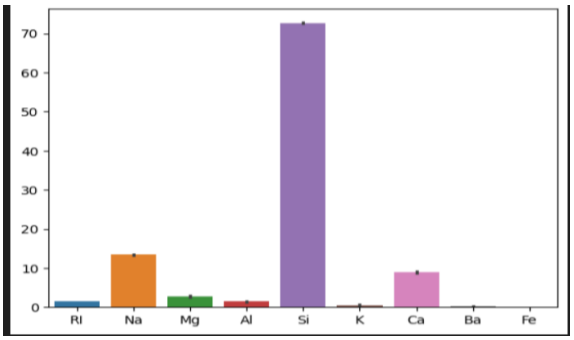
Dis research investigates the Deep learning models for glass classification tasks using spectroscopic data. This study collects spectral data from different glass samples and designs deep neural network architectures, including (CNN) and (RNN), to process the spectral data and predict glass types. Transfer learning techniques are employed to leverage pre-trained models and enhance classification performance. The study evaluates model performance using spectral analysis techniques.

V.PROPOSED METHODOLOGY

For crystal category, a crystal sorting dataset is from a UCI database, which contains categorial information, usually viewed as qualitative info. Our approach integrates various classifying methods, encount chainlug support vector machinery (SVM) algorithm, K. -nearest neighboring (KNN) algorithm, random -In addition forest dealt with, fake neural networks (ANN), and logistic regression alogarithms, including dimensional managing skills like principal component analytical (PCA) were activated. Commen our job by executing the dataset operating the Python Pandas library and analyzizing its parameters. To make the data set visible, we connected car loading the Matplotlib library. Afterwards, we broke down the data set into teaches and experimentistic cliques pursuant to a 5:1 ratio. Post-segmentation, the features of the dataset were thoroughly scanned. A surfeit of features in the dataset presented a hazard of vagueness or overestimating in the template. To tackle this worry, dimension reduction techniques like PCA and XGBoost were administered. Following a rewarding dimension reduction, the template to educated with classifying blueprints.

A. DATABASE: The Glass Categorization data deposited from (UCI). It encloses 10 characteristics with identifier. Drinking glass classifications (discrete 7 measures).

SNS BAR PLOT FOR DATASET:



B. DATA PRE-PROCESSING: Useful algorithms were imported, and the Dataset was loaded into a panda DataFrame. Moving forward, issues with any missing values were checked and handled by removing them. Following that, relevant features for classification were picked and made standard through feature scaling to have an average of 0 and variation of 1. These steps of preprocessing assure that the data is cleansed, standardized, and all set for training machine learning models

C.ALGORITHMS:

1.Support Vector Machine (SVM) algorithm: Incorporating both liner SVM and kerb SVM. Regression and kinds of trouble can be solving with SVM algorithms. SVM reading is serves up one supervised ML strategies. Deep models in complex data may recognize using SVM compared with other algorithms. Primarily, SV is for to resolve classifications problems. For construct the plots, let us consider 'N' variables in the info. Express each factor as data in an n-dimensional realm and the feature value coordinate for each function. We proceed with classifications by calculating the hyperplanes that separate the 2 class from another depending on the variables. The decisions boundary is categorized from hyperplanes. Various

points of training are located from looking at data points on different sides of the planes. Hyperplane's dimensions depend on feature counts.

2. K-Nearest Neighbors (KNN): "Like regression troubles could be resolved by the KNN programming recipe. Herein we examine splendor. An array of outcomes with KNN approaches without preliminary guesses of the distribution of records may address a major portion of class predicaments. KNN operates by computing the gaps among a question and every observation in the records, selecting the most favored sizable type specimen (K) that is nearest to the question, then tossing the least weird unlabeled ballot or lower the integral of the points (in the regression scenario) In this examination, we implemented KNN, and they granted us incredibly imprecise outcomes."

3. Random Forest: The system learning set of rules is named the Ensemble technique, it is the type of attaching various classifier to explain a complicated issue. This utilizes a major voting technique for predicting the data outcome. Voting occurs among the trees for the predictive classes. The decision is taken on classes having the greater number of voting. The accuracy of models is proportionate to quantity of tree. Additionally, they resolve high variance issue. Observed a significantly minor training time compared to other procedures. The accuracies of forecast is excessive despite an extensive datasets.

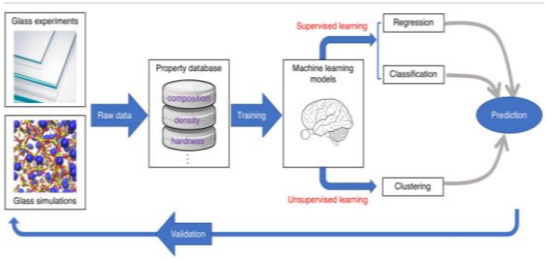
4.Neural Network (NN): Being a deep dive into the code and mainly about the operating of natural thoughts this is a really responsible stage. The multilayer structure is also composed of the layer that is input, the output and the layers for the hidden. Among these are the neurons that generate various sensory and non-sensory activities. Each of these processes is housed in a different layer of the brain. The structure of each and every neuron of the neural network includes a nonlinearity operator which generates a transformation composed from the signals received from the neurons of the preceding layer to an output sign that is sent forward to the neurons of the next layer. This way broadcasters do not differ from the other competitor and act not in their interest, they relay the others rather than transmit their shows.

5.Logistic Regression: Comparable to Linear Regression, Logistic Regression is a supervised learning algorithm however utilized for classification as opposed to regressions. The outcomes variables incongruous, that effective systematic approach is implemented. Thus suggests the corelations among their unbiased variable & classify them into dual formation.Then equations are utilized for their fundamental logit models.

It suggests the linear family members among the impartial variable

& classify them into binary shape! The equation used by the fundamental logistic model.

MODEL BUILDING:



VI.RESULTS AND OBSERVATIONS

TABLE:

ALGORITHM	TRAINING ACC	TESTING ACC	MSE	RMSE	MAE
LOGISTIC REGRESSION	0.625	0.697	1.65	1.28	0.58
RANDOM FOREST	1.0	0.88	2.72	1.64	2.72
DECISION TREE	1.0	0.76	0.97	0.98	0.37
KNN	0.76	0.69	1.0	1.0	0.4
NAÏVE BAYES	0.31	0.35	1.21	1.42	0.34

GRAPHS:

KNN Accuracy graph: It plots the accuracy scores against the values of k using Matplotlib. This visualization in determining the values of 'k' and yields the highest accuracy.

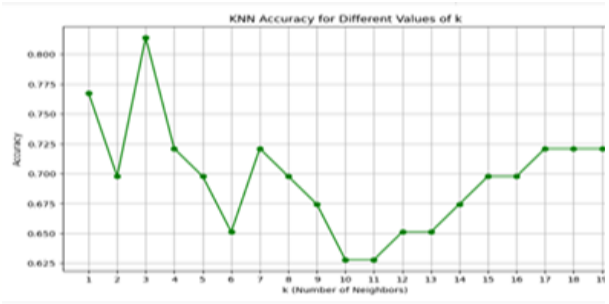


FIG (1):

ACCURACY COMPARISON: This code creates a bar chart showing the test accuracy scores of different classification algorithms. The bar tells the test accuracy of a given algorithm, and the X-axis labels indicate their names of algorithms.

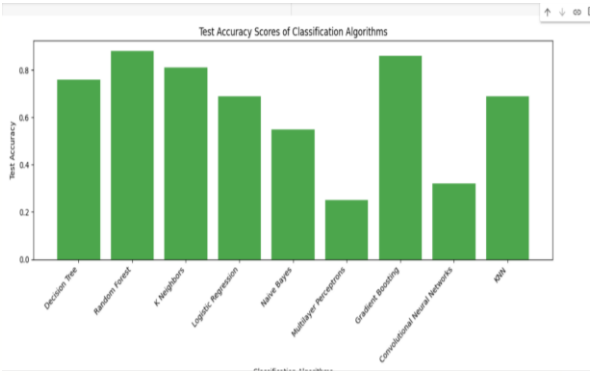


FIG (2):

Graph between Si and Fe: (SCATTER PLOT): This code segment creates a scatter plot to visualize the relationship between the'Si' and 'Fe' features.

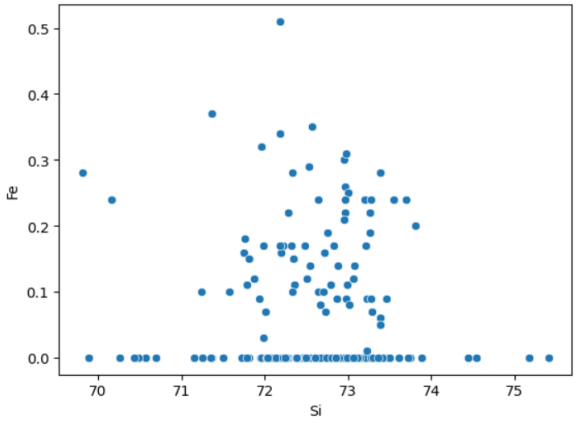


FIG (3):

Different glass types:(PAIRPLOT):

Pair plot to explore the relationship b/w all the features in Dataset, and the points coloured by their glass types.

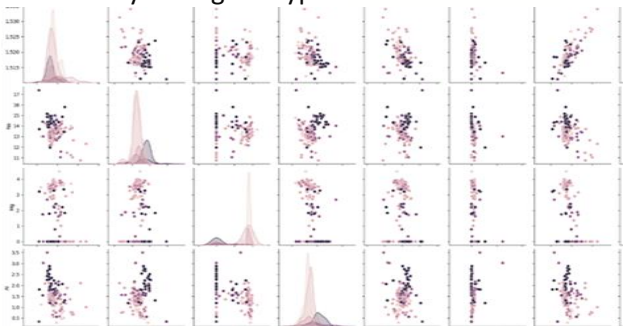


FIG (4):

HEATMAP: It displays the correlation matrix of the features from data sets. Each cell in the heat map represents the correlation coefficient of the two components.

FIG (5):

CONVOLUTIONAL NEURAL NETWORKS: Plotted graph illustrates the model training and accuracy over epochs. It provides insight into how well the model learn from the Training data and generalizes to unseen data.

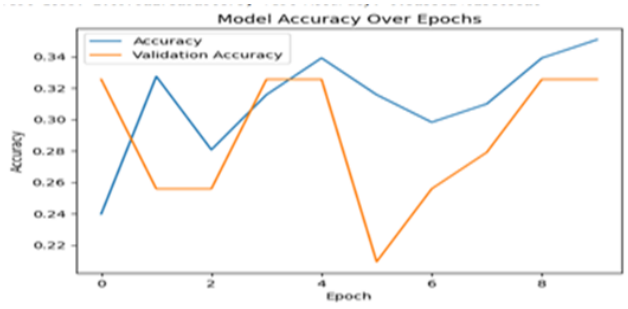


FIG (6):

Scatter plots for each column in the Data Frame df against the "RI" (Refractive Index) column using the seaborn library in Python.

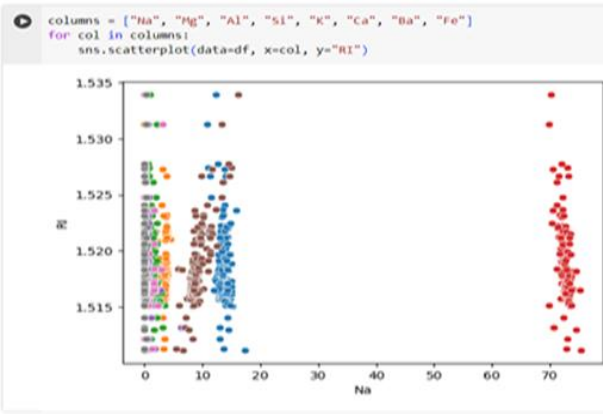


FIG (7):

This is 3D scatter plot to visualize the differrence b/w the actual (y_test) and predicted (y_pred test) values of a target variable.

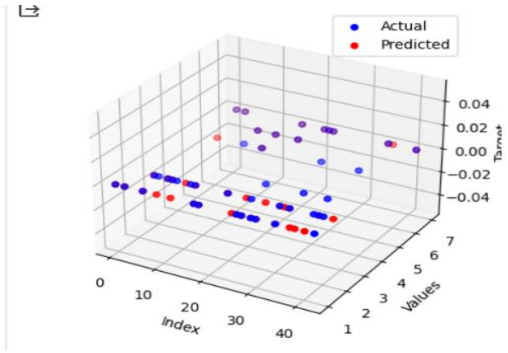
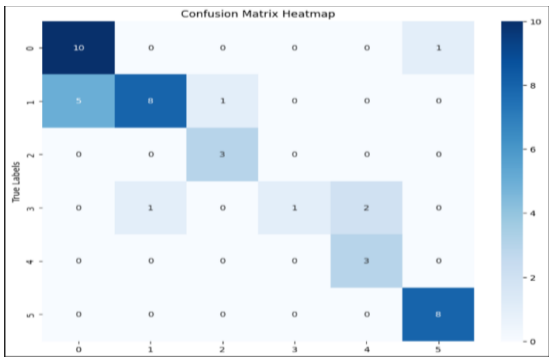


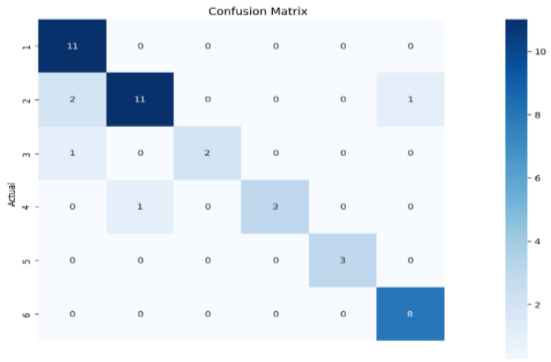
FIG (8):

CONFUSION MATRIX FOR ALGORITHMS USED:

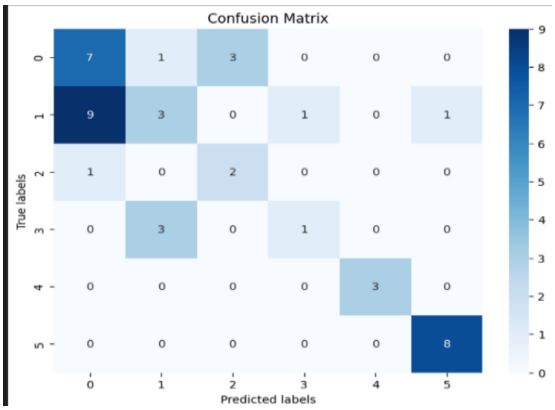
CONFUSION MATRIX FOR DECISION TREE:



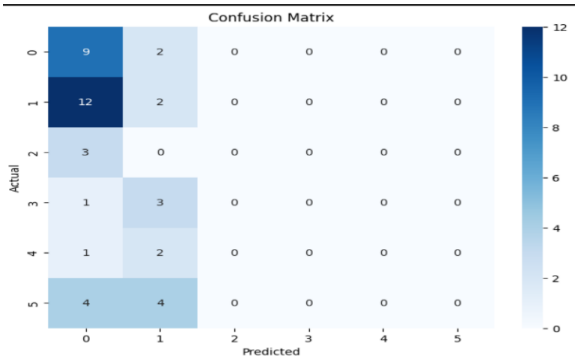
RANDOM FOREST:



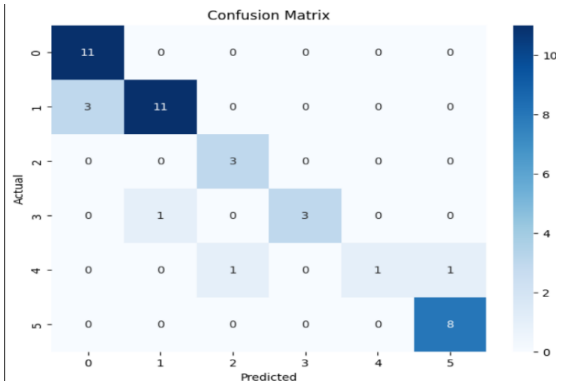
NAÏVE BAYES:



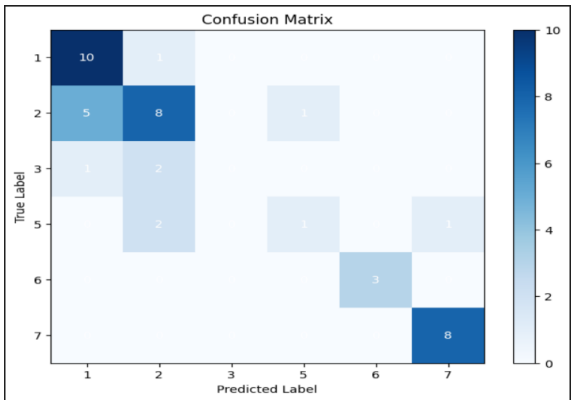
MULTILAYER PERCEPTRONS:



GRADIENT BOOSTING:



KNN:



VII. CONCLUSION

Involved several ML algorithm to forecast different glass types.Then decided the suitable algorithms to answer this kind of problems,in accordance with accuracy of the algorithm's models. the random forest model performed better with an accuracy of 88.37%, subsequently by support vector machine with an accuracy 77.77% .

VIII.REFERENCES

[1].B.German 2019). UCI Machine Learning Repository[https://archive.ics.uci.edu/ml/datasets/Glass+Identification]. Central Research Establishment Home Office Forensic Science Service Aldermeston

[2].J E T Akinsola. Supervised Machine Learning Algorithms: Classification and Comparison International Journal of Computer Trends and Technology (IJCTT) – Volume 48 Number 3 June 2017

[3].Nagaraju Kolla & M. Giridhar Kumar. Supervised Learning Algorithms of Machine Learning: Prediction of Brand Loyalty. International Journal of Innovative Technology and Exploring
Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-11, September 2019

[4].Aruna S, Rajagopalan SP. A novel SVM based CSSFFS feature selection algorithm for detecting breast cancer. Int J Comput Appl. 2011;31(8):14–20.R. Nicole, “Title of paper with only first

[5].Chao-Ying Joanne Peng, Kuk Lida Lee and Gary M. Ingersoll. The Journal of Educational Research.96(1), 3-14. DOI: 10.1080/ 00220670209598786

[6].Silverman D (2006) Interpreting qualitative data: methods for analyzing talk, text and interaction. Sage, Beverly Hills

[7].Deza MM, Deza E (2014) Encyclopedia of distances. Springer, Berlin ISBN 9783662443422

[8].Cunningham P, Delany SJ (2007) K-nearest neighbor classifers. Mult Classif Syst 34:1–17

[9].Devroye, L. (1981) "On the equality of Cover and Hart in nearest neighbor discrimination", IEEE Trans. Pattern Anal. Mach. Intell. 3: 75- 78.

[10] Ziad Al-Halah; Andrew Aitken; Wenzhe Shi; Jose Caballero “Emoji Embedding for Visual Sentiment Analysis” 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW) Seoul, Korea (South), pp.3-5,DOI: 10.1109/ICCVW.2019.00550

[11] b.gnanapriya “emoji based sentiment analysis using knnInternational Journal of Scientific Research and Review Volume 07, Issue 04, April 2019, Annāmalainagar, India.

[12] M. A. Shaik, M. Y. Sree, S. S. Vyshnavi, T. Ganesh, D. Sushmitha and N. Shreya, “Fake News Detection using NLP”, 2023 International Conference on Inovative Data Communication Technologies and Application (ICIDCA), Uttarakhand, India, 2023, pp. 399-405, doi: 10.1109/ICIDCA56705.2023.10100305.

[13] Mohammed Ali Shaik, M. Varshith, S. SriVyshnavi, N. Sanjana and R. Sujith, “Laptop Price Prediction using Machine Learning Algorithms”, 2022 International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS), Nagpur, India, 2022, pp. 226-231, doi: 10.1109/ICETEMS56252.2022.10093357.

[14] Mohammed Ali Shaik, Dhanraj Verma, P Praveen, K Ranganath and Bonthala Prabhanjan Yadav, (2020), RNN based prediction of spatiotemporal data mining, 2020 IOP Conf. Ser.: Mater. Sci. Eng. 981 022027, doi.org/10.1088/1757-899X/981/2/022027

glass-classification-code

April 29, 2024

[ ]:
```
import pandas as pd
df= pd.read_csv('/content/glass.csv')
print(df)
```

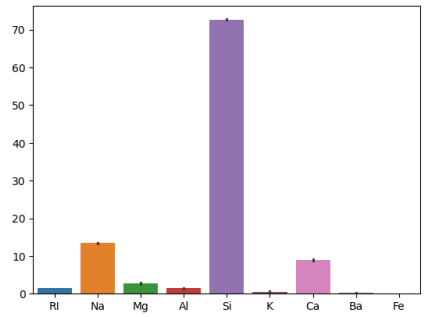
```
          RI     Na   Mg    Al     Si     K    Ca    Ba   Fe  Type
0    1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0     1
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     1
2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     1
3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     1
4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     1
..       ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     7
210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     7
211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     7
212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     7
213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     7

[214 rows x 10 columns]
```

[ ]:
```
df.isnull().sum()
```

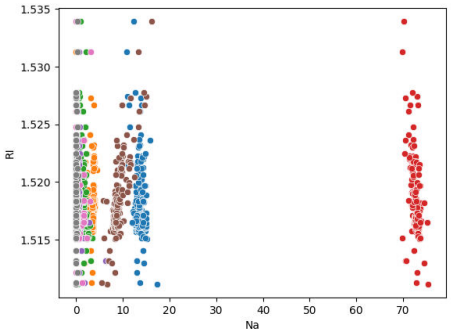
[ ]:
```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

[ ]:
```
sns.barplot(X)
```

[ ]:
```
<Axes: >
```

[ ]:
```
columns = ["Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe"]
for col in columns:
    sns.scatterplot(data=df, x=col, y="RI")
```

DECISION TREE CLASSIFIER

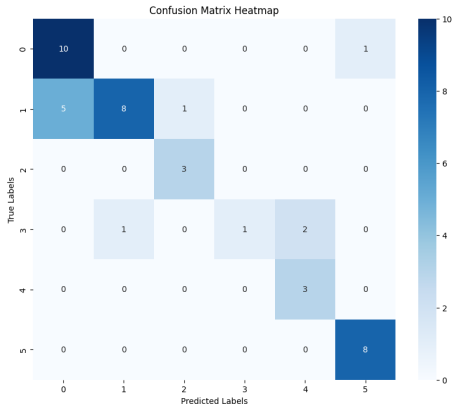
[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score,␣
↪confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Assuming X, y, and df are defined

X = df.drop('Type', axis=1)
y = df['Type']

# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Decision Tree Classifier
dt_classifier = DecisionTreeClassifier()
```



```
dt_classifier.fit(X_train, y_train)

# Predictions
y_train_pred = dt_classifier.predict(X_train)
y_pred = dt_classifier.predict(X_test)

# Accuracy
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_pred)

print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Plot Confusion Matrix
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.7674418604651163

Classification Report:
              precision    recall  f1-score   support

           1       0.67      0.91      0.77        11
           2       0.89      0.57      0.70        14
           3       0.75      1.00      0.86         3
           5       1.00      0.25      0.40         4
           6       0.60      1.00      0.75         3
           7       0.89      1.00      0.94         8

    accuracy                           0.77        43
   macro avg       0.80      0.79      0.74        43
weighted avg       0.81      0.77      0.75        43
```

```
Confusion Matrix:
[[10  0  0  0  0  1]
 [ 5  8  1  0  0  0]
 [ 0  0  3  0  0  0]
 [ 0  1  0  1  2  0]
 [ 0  0  0  0  3  0]
 [ 0  0  0  0  0  8]]
```




```
[ ]: from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Assuming X and y are defined
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

dt_regressor = DecisionTreeRegressor()
dt_regressor.fit(X_train, y_train)
y_pred = dt_regressor.predict(X_test)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Squared Error (MSE): 0.9767441860465116
Mean Absolute Error (MAE): 0.3720930232558139
Root Mean Squared Error (RMSE): 0.9883036912035246
```

RANDOM FOREST

```
[ ]: from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,␣
↪confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming X and y are defined

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)

# Predictions on training set
y_train_pred = rf_classifier.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
print("Training Accuracy:", train_accuracy)

# Predictions on test set
y_pred = rf_classifier.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)
```

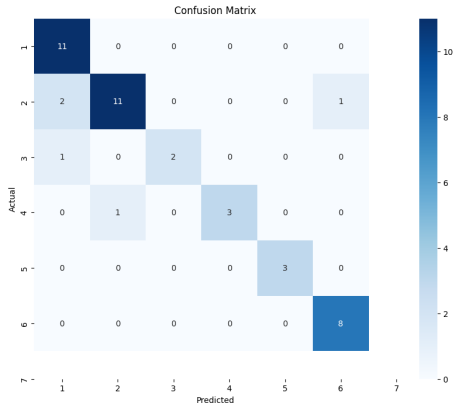


```
print("Testing Accuracy:", test_accuracy)

# Printing accuracy percentage
accuracy_percentage = test_accuracy * 100
print("Accuracy Percentage:", accuracy_percentage)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Plotting confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", xticklabels=range(1, 8),␣
↪yticklabels=range(1, 8))
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.8837209302325582
Accuracy Percentage: 88.37209302325581

Classification Report:
              precision    recall  f1-score   support

           1       0.79      1.00      0.88        11
           2       0.92      0.79      0.85        14
           3       1.00      0.67      0.80         3
           5       1.00      0.75      0.86         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.88        43
   macro avg       0.93      0.87      0.89        43
weighted avg       0.90      0.88      0.88        43


Confusion Matrix:
[[11  0  0  0  0  0]
 [ 2 11  0  0  0  1]
 [ 1  0  2  0  0  0]
 [ 0  1  0  3  0  0]
 [ 0  0  0  0  3  0]
```



```
 [ 0  0  0  0  0  8]]
```




```
[ ]: from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Assuming X and y are defined

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)

# Predictions on test set
```

```
y_pred_proba = rf_classifier.predict_proba(X_test)
y_pred = np.argmax(y_pred_proba, axis=1)  # Convert probabilities to class labels
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
test_mae = mean_absolute_error(y_test, y_pred)
test_mse = mean_squared_error(y_test, y_pred)

print("Testing RMSE:", test_rmse)
print("Testing MAE:", test_mae)
print("Testing MSE:", test_mse)
```

```
Testing RMSE: 1.649524244307473
Testing MAE: 1.5116279069767442
Testing MSE: 2.7209302325581395
```

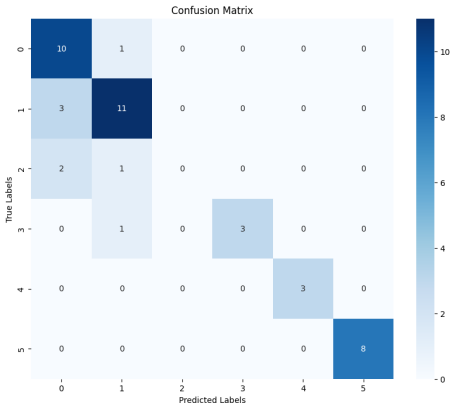
K NEIGHBORS

```
[ ]: from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Assuming X and y are defined

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)

# Predictions on training set
y_train_pred = knn.predict(X_train_scaled)
train_accuracy = accuracy_score(y_train, y_train_pred)
print("Training Accuracy:", train_accuracy)

# Predictions on test set
y_pred = knn.predict(X_test_scaled)
test_accuracy = accuracy_score(y_test, y_pred)
print("Testing Accuracy:", test_accuracy)
# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)

# Plotting confusion matrix
plt.figure(figsize=(10, 8))
```

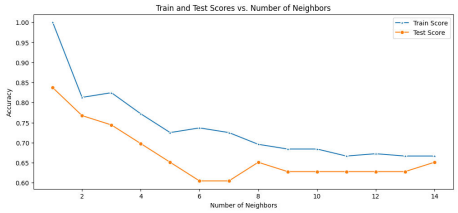


```
sns.heatmap(conf_matrix, annot=True, cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

```
Training Accuracy: 0.8245614035087719
Testing Accuracy: 0.813953488372093

Confusion Matrix:
[[10  1  0  0  0  0]
 [ 3 11  0  0  0  0]
 [ 2  1  0  0  0  0]
 [ 0  1  0  3  0  0]
 [ 0  0  0  0  3  0]
 [ 0  0  0  0  0  8]]
```




```
[ ]: import matplotlib.pyplot as plt
import seaborn as sns
test_score_list = []
train_score_list = []

for i in range(1, 15):
    knn2 = KNeighborsClassifier(n_neighbors=i)
    knn2.fit(X_train, y_train)
    test_score_list.append(knn2.score(X_test, y_test))
    train_score_list.append(knn2.score(X_train, y_train))

plt.figure(figsize=(12, 5))
sns.lineplot(x=range(1, 15), y=train_score_list, marker='*', label='Train Score')
sns.lineplot(x=range(1, 15), y=test_score_list, marker='o', label='Test Score')
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.title('Train and Test Scores vs. Number of Neighbors')
plt.legend()
plt.show()
```



LOGISTIC REGRESSION

```
[ ]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Assuming you already have X_train, X_test, y_train, and y_test defined

# Instantiate the model with a higher max_iter value
```



```
model = LogisticRegression(max_iter=1000)

# Train the model
model.fit(X_train, y_train)

# Predictions on training set
y_train_pred = model.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
print('Accuracy on training set using logistic regression:', train_accuracy)

# Predictions on test set
y_pred = model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)
print('Accuracy on testing set using logistic regression:', test_accuracy)
```

```
Accuracy on training set using logistic regression: 0.6257309941520468
Accuracy on testing set using logistic regression: 0.6976744186046512

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
[ ]: from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Mean Squared Error (MSE)
def mse(y_true, y_pred):
    return mean_squared_error(y_true, y_pred)

# Root Mean Squared Error (RMSE)
def rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))

# Mean Absolute Error (MAE)
def mae(y_true, y_pred):
    return mean_absolute_error(y_true, y_pred)

mse_value = mse(y_test, y_pred)
rmse_value = rmse(y_test, y_pred)
mae_value = mae(y_test, y_pred)
```

```
print('MSE:', mse_value)
print('RMSE:', rmse_value)
print('MAE:', mae_value)
```

```
MSE: 1.6511627906976745
RMSE: 1.2849757938177957
MAE: 0.5813953488372093
```

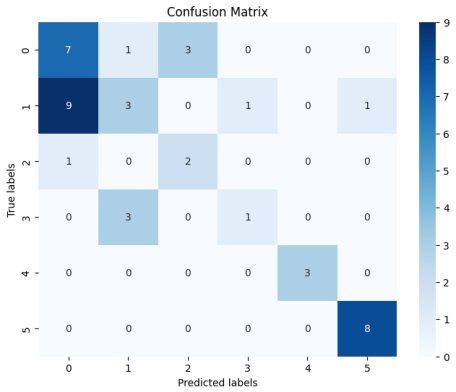
NAIVE BAYES

[ ]:
```
from sklearn.naive_bayes import GaussianNB
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred = nb_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(f'Accuracy Percentage: {accuracy * 100:.2f}%')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.5581395348837209
Accuracy Percentage: 55.81%
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43

[[7 1 3 0 0 0]
 [9 3 0 1 0 1]
 [1 0 2 0 0 0]
 [0 3 0 1 0 0]
```



```
[0 0 0 0 3 0]
[0 0 0 0 0 8]]
```



SVM

[ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Train the model
model = SVC()
model.fit(X_train, y_train)

# Predictions on training set
y_train_pred = model.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)

# Predictions on testing set
y_pred = model.predict(X_test)
```



```
test_accuracy = accuracy_score(y_test, y_pred)

# Convert accuracy to percentage
train_accuracy_percentage = train_accuracy * 100
test_accuracy_percentage = test_accuracy * 100

# Print accuracies
print(f'Training Accuracy: {train_accuracy_percentage}')
print(f'Testing Accuracy: {test_accuracy_percentage}')
```

```
Training Accuracy: 36.25730994152047
Testing Accuracy: 32.55813953488372
```

[ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Train the model
model = SVC()
model.fit(X_train, y_train)

# Predictions on training set
y_train_pred = model.predict(X_train)

# Convert categorical labels to numerical values
y_train_numeric = np.arange(len(np.unique(y_train)))
y_train_numeric_dict = dict(zip(np.unique(y_train), y_train_numeric))
y_train_numeric = np.array([y_train_numeric_dict[label] for label in y_train])
y_train_pred_numeric = np.array([y_train_numeric_dict[label] for label in␣
↪y_train_pred])

# Calculate regression metrics
train_rmse = np.sqrt(mean_squared_error(y_train_numeric, y_train_pred_numeric))
train_mae = mean_absolute_error(y_train_numeric, y_train_pred_numeric)
train_mse = mean_squared_error(y_train_numeric, y_train_pred_numeric)

# Predictions on testing set
y_pred = model.predict(X_test)

# Convert categorical labels to numerical values
y_test_numeric = np.array([y_train_numeric_dict[label] for label in y_test])
y_pred_numeric = np.array([y_train_numeric_dict[label] for label in y_pred])

# Calculate regression metrics
test_rmse = np.sqrt(mean_squared_error(y_test_numeric, y_pred_numeric))
test_mae = mean_absolute_error(y_test_numeric, y_pred_numeric)
test_mse = mean_squared_error(y_test_numeric, y_pred_numeric)
```



```
# Print regression metrics
print(f'Training RMSE: {train_rmse}')
print(f'Training MAE: {train_mae}')
print(f'Training MSE: {train_mse}')
print(f'Testing RMSE: {test_rmse}')
print(f'Testing MAE: {test_mae}')
print(f'Testing MSE: {test_mse}')
```

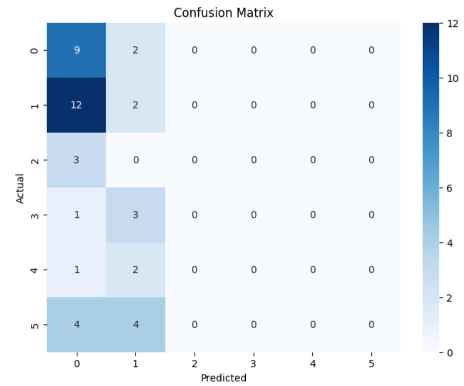
```
Training RMSE: 1.7082531003837693
Training MAE: 1.128654970760234
Training MSE: 2.91812865497076
Testing RMSE: 2.07420480700324
Testing MAE: 1.4651162790697674
Testing MSE: 4.3023255813953485
```

MULTILAYER PERCEPTRONS

[ ]:
```
from sklearn.neural_network import MLPClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('Accuracy Percentage: {:.2f}%'.format(accuracy * 100))
print('Classification Report:')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.2558139534883721
Accuracy Percentage: 25.58%
Classification Report:
              precision    recall  f1-score   support

           1       0.30      0.82      0.44        11
           2       0.15      0.14      0.15        14
           3       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         4
           6       0.00      0.00      0.00         3
```

```
           7       0.00      0.00      0.00         8

    accuracy                           0.26        43
   macro avg       0.08      0.16      0.10        43
weighted avg       0.13      0.26      0.16        43

[[ 9  2  0  0  0  0]
 [12  2  0  0  0  0]
 [ 3  0  0  0  0  0]
 [ 1  3  0  0  0  0]
 [ 1  2  0  0  0  0]
 [ 4  4  0  0  0  0]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
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))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
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))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
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))
```

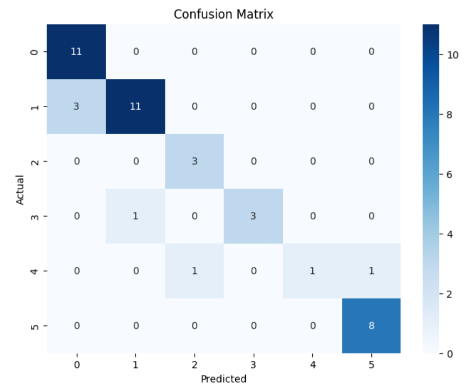
GRADIENT BOOSTING

```
[ ]: from sklearn.ensemble import GradientBoostingClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 ↪random_state=42)
clf = GradientBoostingClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(f'Accuracy Percentage: {accuracy * 100:.2f}%')
print('Classification Report:')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues')
plt.xlabel('Predicted')
```



```
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.8604651162790697
Accuracy Percentage: 86.05%
Classification Report:
              precision    recall  f1-score   support

           1       0.79      1.00      0.88        11
           2       0.92      0.79      0.85        14
           3       0.75      1.00      0.86         3
           5       1.00      0.75      0.86         4
           6       1.00      0.33      0.50         3
           7       0.89      1.00      0.94         8

    accuracy                           0.86        43
   macro avg       0.89      0.81      0.81        43
weighted avg       0.88      0.86      0.85        43

[[11  0  0  0  0  0]
 [ 3 11  0  0  0  0]
 [ 0  0  3  0  0  0]
 [ 0  1  0  3  0  0]
 [ 0  0  1  0  1  1]
 [ 0  0  0  0  0  8]]
```

CONVOLUTIONAL NEURAL NETWORKS

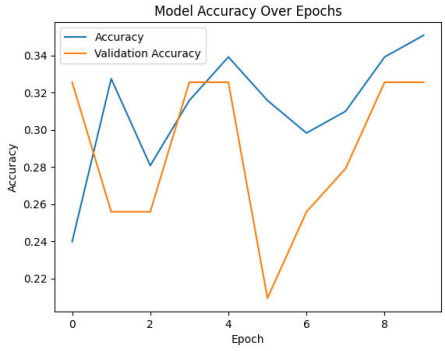
```
[ ]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense,
 ↪Dropout
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv("/content/glass.csv")

# Encode the Type column to numerical values
label_encoder = LabelEncoder()
df['Type'] = label_encoder.fit_transform(df['Type'])
```

```
# Split data into features and labels
X = df.drop('Type', axis=1).values
y = df['Type'].values

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Reshape data for 1D convolutional layer
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Define the model
model = Sequential([
    Conv1D(32, 3, activation='relu', input_shape=(X_train.shape[1], 1),␣
↪padding='same'),
    MaxPooling1D(2),
    Conv1D(64, 3, activation='relu', padding='same'),
    MaxPooling1D(2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy'])

# Train the model and store training history
history = model.fit(X_train, y_train, epochs=10, batch_size=32,␣
↪validation_data=(X_test, y_test))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}, Test Accuracy: {accuracy}")

# Plotting the accuracy graph
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.legend()
plt.show()
```

```
Epoch 1/10
```



```
6/6 [==============================] - 1s 43ms/step - loss: 6.5914 - accuracy:
0.2398 - val_loss: 4.1519 - val_accuracy: 0.3256
Epoch 2/10
6/6 [==============================] - 0s 6ms/step - loss: 3.6148 - accuracy:
0.3275 - val_loss: 2.2876 - val_accuracy: 0.2558
Epoch 3/10
6/6 [==============================] - 0s 6ms/step - loss: 2.8652 - accuracy:
0.2807 - val_loss: 1.7415 - val_accuracy: 0.2558
Epoch 4/10
6/6 [==============================] - 0s 6ms/step - loss: 1.9100 - accuracy:
0.3158 - val_loss: 1.7901 - val_accuracy: 0.3256
Epoch 5/10
6/6 [==============================] - 0s 6ms/step - loss: 1.7797 - accuracy:
0.3392 - val_loss: 1.8127 - val_accuracy: 0.3256
Epoch 6/10
6/6 [==============================] - 0s 9ms/step - loss: 1.6729 - accuracy:
0.3158 - val_loss: 1.7053 - val_accuracy: 0.2093
Epoch 7/10
6/6 [==============================] - 0s 10ms/step - loss: 1.6955 - accuracy:
0.2982 - val_loss: 1.7125 - val_accuracy: 0.2558
Epoch 8/10
6/6 [==============================] - 0s 7ms/step - loss: 1.7195 - accuracy:
0.3099 - val_loss: 1.7033 - val_accuracy: 0.2791
Epoch 9/10
6/6 [==============================] - 0s 8ms/step - loss: 1.6297 - accuracy:
0.3392 - val_loss: 1.6917 - val_accuracy: 0.3256
Epoch 10/10
6/6 [==============================] - 0s 6ms/step - loss: 1.6636 - accuracy:
0.3509 - val_loss: 1.6979 - val_accuracy: 0.3256
2/2 [==============================] - 0s 5ms/step - loss: 1.6979 - accuracy:
0.3256
Test Loss: 1.6979217529296875, Test Accuracy: 0.3255814015865326
```

KNN

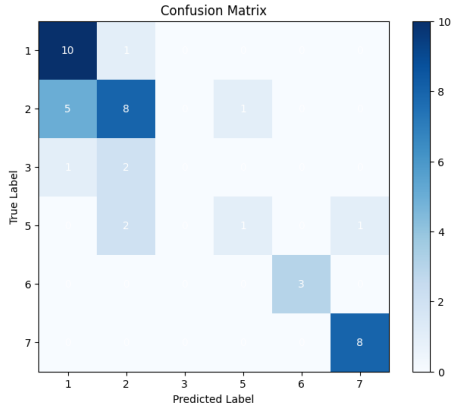
[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Load the dataset
data = pd.read_csv('/content/glass.csv')

# Split data into features (X) and target variable (y)
X = data.drop('Type', axis=1)
y = data['Type']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```



```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Implement KNN algorithm
k = 5  # number of neighbors
classifier = KNeighborsClassifier(n_neighbors=k)
classifier.fit(X_train, y_train)

# Predict the test set results
y_pred = classifier.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Create a confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xticks(np.arange(len(data['Type'].unique())), data['Type'].unique())
plt.yticks(np.arange(len(data['Type'].unique())), data['Type'].unique())
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
for i in range(len(data['Type'].unique())):
    for j in range(len(data['Type'].unique())):
        plt.text(j, i, cm[i, j], ha='center', va='center', color='white')
plt.show()
from sklearn.metrics import accuracy_score

# Train set accuracy
train_pred = classifier.predict(X_train)
train_accuracy = accuracy_score(y_train, train_pred)
print("Training Accuracy:", train_accuracy)

# Test set accuracy
test_accuracy = accuracy_score(y_test, y_pred)
print("Testing Accuracy:", test_accuracy)
```

```
Accuracy: 0.6976744186046512
Confusion Matrix:
```

```
[[10  1  0  0  0  0]
 [ 5  8  0  1  0  0]
 [ 1  2  0  0  0  0]
 [ 0  2  0  1  0  1]
 [ 0  0  0  0  3  0]
 [ 0  0  0  0  0  8]]
```

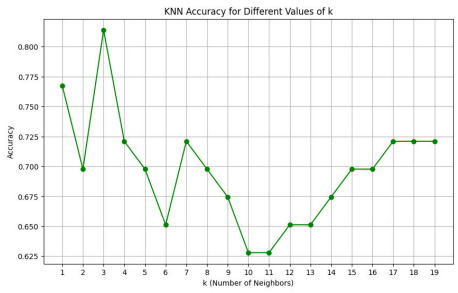



```
Training Accuracy: 0.7660818713450293
Testing Accuracy: 0.6976744186046512
```

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```



```
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv('/content/glass.csv')

# Split data into features (X) and target variable (y)
X = data.drop('Type', axis=1)
y = data['Type']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Experiment with different values of k
k_values = range(1, 20)
accuracy_scores = []

for k in k_values:
    # Implement KNN algorithm
    classifier = KNeighborsClassifier(n_neighbors=k)
    classifier.fit(X_train, y_train)

    # Predict the test set results
    y_pred = classifier.predict(X_test)

    # Calculate the accuracy
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

# Plot the accuracy scores for different values of k
plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracy_scores, marker='o', linestyle='-', color='g')
plt.title('KNN Accuracy for Different Values of k')
plt.xlabel('k (Number of Neighbors)')
plt.ylabel('Accuracy')
plt.xticks(k_values)
plt.grid(True)
plt.show()
```

[ ]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Calculate RMSE, MAE, and MSE for the testing set
test_rmse = mean_squared_error(y_test, y_pred, squared=False)  # RMSE
test_mae = mean_absolute_error(y_test, y_pred)  # MAE
test_mse = mean_squared_error(y_test, y_pred)  # MSE

# Print the results
print("Testing RMSE:", test_rmse)
print("Testing MAE:", test_mae)
print("Testing MSE:", test_mse)
```

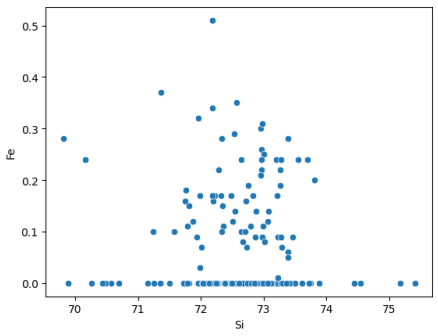
```
Testing RMSE: 1.0
Testing MAE: 0.4883720930232558
Testing MSE: 1.0
```

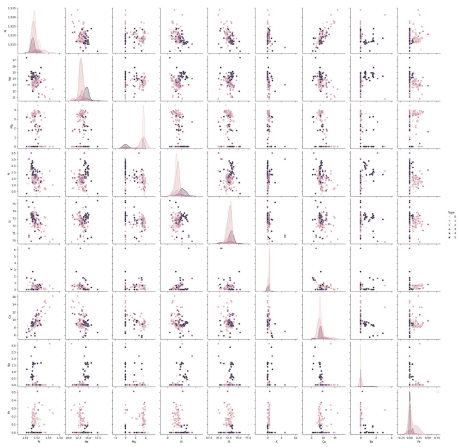
SCATTERPLOT BETWEEN DIFFERENT GLASS TYPES

[ ]:
```
x = df.drop('Type' , axis = 1)
y = df['Type']

sns.scatterplot(x = x.Si, y = x.Fe)
sns.pairplot(df, hue = "Type")
plt.legend()
plt.show()
```



```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
```

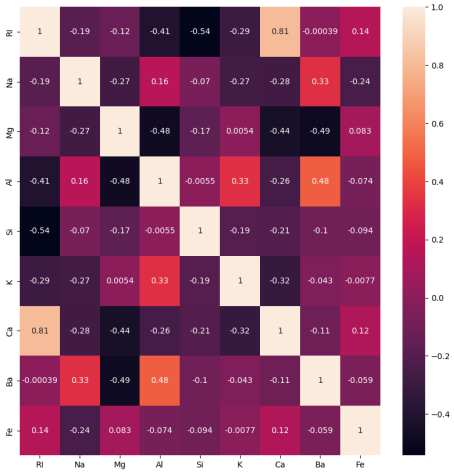




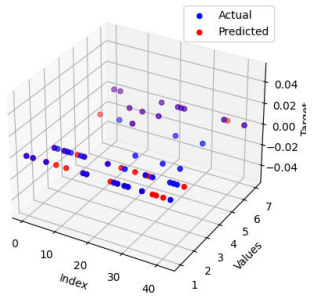
HEATMAP FOR GLASS TYPES

[ ]:
```
plt.figure(figsize=(10,10))
sns.heatmap(x.corr(), annot= True)
```

[ ]: <Axes: >



[ ]:
```
index = np.arange(len(y_test))
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(index, y_test, c='blue', label='Actual')
ax.scatter(index, y_pred, c='red', label='Predicted')
ax.set_xlabel('Index')
ax.set_ylabel('Values')
ax.set_zlabel('Target')
ax.legend()
plt.show()
```

[11]:
```
import matplotlib.pyplot as plt

# Algorithm names
algorithm_names = ['Decision Tree', 'Random Forest', 'K Neighbors', 'Logistic␣
↪Regression',
                   'Naive Bayes', 'Multilayer Perceptrons',
                   'Gradient Boosting', 'Convolutional Neural Networks', 'KNN']

# Test accuracy scores (replace with actual test accuracy scores)
test_accuracy_scores = [0.76, 0.88, 0.81, 0.69, 0.55,  0.25, 0.86, 0.32, 0.69]

plt.figure(figsize=(12, 6))

# Bar chart for test accuracy scores
plt.bar(algorithm_names, test_accuracy_scores, color='green', alpha=0.7)
plt.xlabel('Classification Algorithms')
plt.ylabel('Test Accuracy')
plt.title('Test Accuracy Scores of Classification Algorithms')
plt.xticks(rotation=45, ha='right')  # Rotate x labels for better readability
plt.tight_layout()

# Displaying the plot
plt.show()
```