VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI -590 014

A

Mini-Project Report

on

"IMAGE RECOGNITION WITH MACHINE LEARNING ALGORITHM"

Submitted for partial fulfillment of requirement for the award of Degree of

Bachelor of Engineering

in

Department of CSE (DATA SCIENCE)

Submitted by

MEHEK A	**[1RL22CD028]**
NASREEN T S	**[1RL22CD032]**
NAVJOT KAUR	**[1RL22CD033]**
SAYADA RUQAYYA	**[1RL22CD047]**

of

V Semester

Under the guidance of

Dr Mrutyunjaya M S

Associate Professor

Department of CSE (DATA SCIENCE)

R. L. JALAPPA INSTITUTE OF TECHNOLOGY

Doddaballapur, Bangalore Rural District-561 203

2023-24

TITLE OF CONTENT:

ABSTRACT

Image recognition using deep learning has become a cornerstone of modern computer vision, enabling machines to achieve human-level accuracy in identifying and classifying objects within images. Central to this advancement is the use of Convolutional Neural Networks (CNNs), which are particularly effective at capturing spatial hierarchies in images. Deep learning architectures such as AlexNet, VGG, ResNet, and Inception have significantly improved the precision of image recognition tasks by leveraging deeper network designs and techniques like residual connections to mitigate the vanishing gradient problem. These models have been trained on massive datasets, such as ImageNet, using advanced hardware like GPUs and TPUs, allowing for substantial gains in both accuracy and efficiency.

While deep learning has excelled in image recognition, challenges persist. Models are often data-hungry, requiring large labeled datasets for training, which can be difficult to acquire in some domains. Additionally, they can be vulnerable to adversarial attacks and tend to lack interpretability, which hinders their deployment in sensitive applications. Ongoing research focuses on improving robustness, data efficiency, and explainability of these models, as well as exploring novel architectures and transfer learning to make deep learning more accessible and practical across a wider range of industries.

INTRODUCTION:

Image recognition is a key aspect of computer vision, enabling machines to interpret and classify visual data using machine learning, particularly deep learning techniques like Convolutional Neural Networks (CNNs). The process involves collecting diverse labeled datasets, preprocessing images for better performance, training models to adjust parameters, and evaluating their accuracy. Applications span various fields, including healthcare diagnostics, autonomous vehicles, facial recognition, and retail analytics. Despite its promise, challenges such as data quality, computational requirements, and generalization to unseen data persist. As technology evolves, image recognition will increasingly transform how machines understand and interact with visual information.

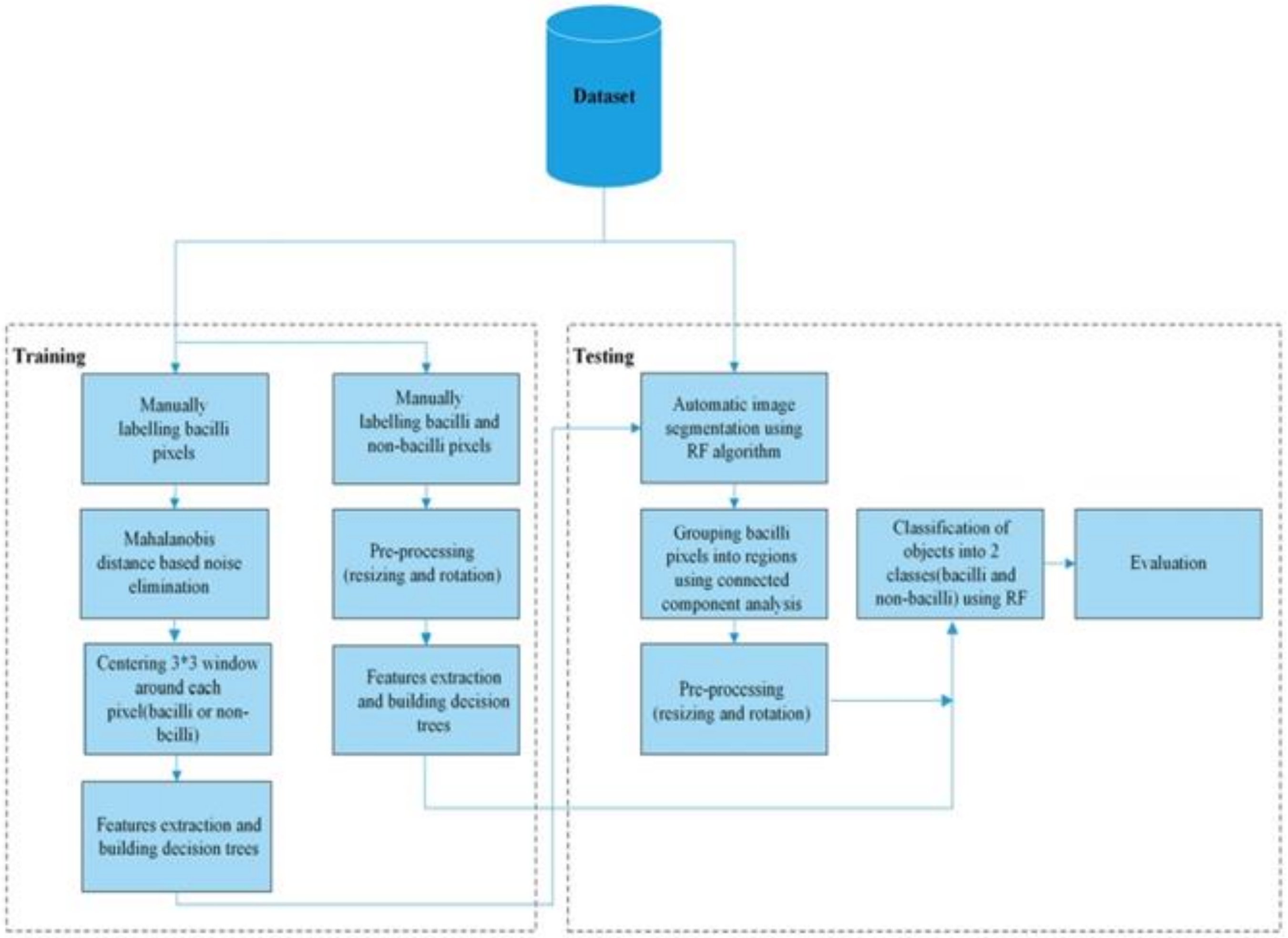
Nowadays, more and more people use images to represent and transmit information. It is convenient for us to learn a lot of information from images. Image recognition is an important research area for its widely applications. For the image recognition problem such as handwritten classification, we should know how to use the data to represent images. The data here is not the row pixels, but should be the feature of images which has high level representation.

The extensive use of deep learning algorithms has led to significant advancements in image recognition and classification, two crucial aspects of computer vision. Automated object recognition and classification in photos have extensive implications across multiple fields, such as self-driving vehicles, medical analysis, and security monitoring . Deep learning architectures have surpassed traditional image processing methods, which mainly rely on handmade features and shallow learning models. Large-scale annotated datasets, advanced neural network topologies, and significant advancements in computing power are primarily responsible for this paradigm change.

LITERATURE SURVEY:

Sl no	Title of Paper	Author	Methodology	Result	Drawbacks
1	Machine Learning and Deep Learning Based Computational Approaches in Automatic Microorganisms Image Recognition: Methodologies, Challenges, and Developments	• Priya Rani • Shallu Kotwal • Jatinder Manhas • Vinod Sharma • Sparsh Sharma	• Review protocol development • Research Questions framing • Search Process and Sources of Information • Inclusion and Exclusion Criteria • Data visualization	• Automated classification & identification • Feature extraction & selection • Development trends	• Lack of publicly available datasets • Overlapping micro-organism species • Long training times for DL models • Challenges in full automation
2	Deep Learning Techniques for Image Recognition and Classification	• Divya Nimma • Rajendar Nimma • Arjun Uddagiri	• Convolutional Neural Networks (CNNs) • Recurrent Neural Networks (RNNs) • Generative Adversarial Networks (GANs) • Hybrid Models	• Deep learning techniques overview • Comparative strengths	• Computational efficiency • Training stability • Complexity of hybrid models
3	Image recognition based on deep learning	• Meiyin Wu • Li Chen	• Overview of Image Recognition Technology • Deep Learning models • Comparative analysis	• The deep learning, especially CNNs demonstrated strong performance across several image recognition tasks • Learning captivity • Versatility	• Data dependency • Computational complexity • Generalization • Interpretability

DATA SET:



Dataset
Training
Manually labelling bacilli pixels
Mahalanobis distance based noise elimination
Centering 3*3 window around each pixel(bacilli or non-bcilli)
Features extraction and building decision trees
Manually labelling bacilli and non-bacilli pixels
Pre-processing (resizing and rotation)
Features extraction and building decision trees
Testing
Automatic image segmentation using RF algorithm
Grouping bacilli pixels into regions using connected component analysis
Pre-processing (resizing and rotation)
Classification of objects into 2 classes(bacilli and non-bacilli) using RF
Evaluation

DATAFLOW MODEL:

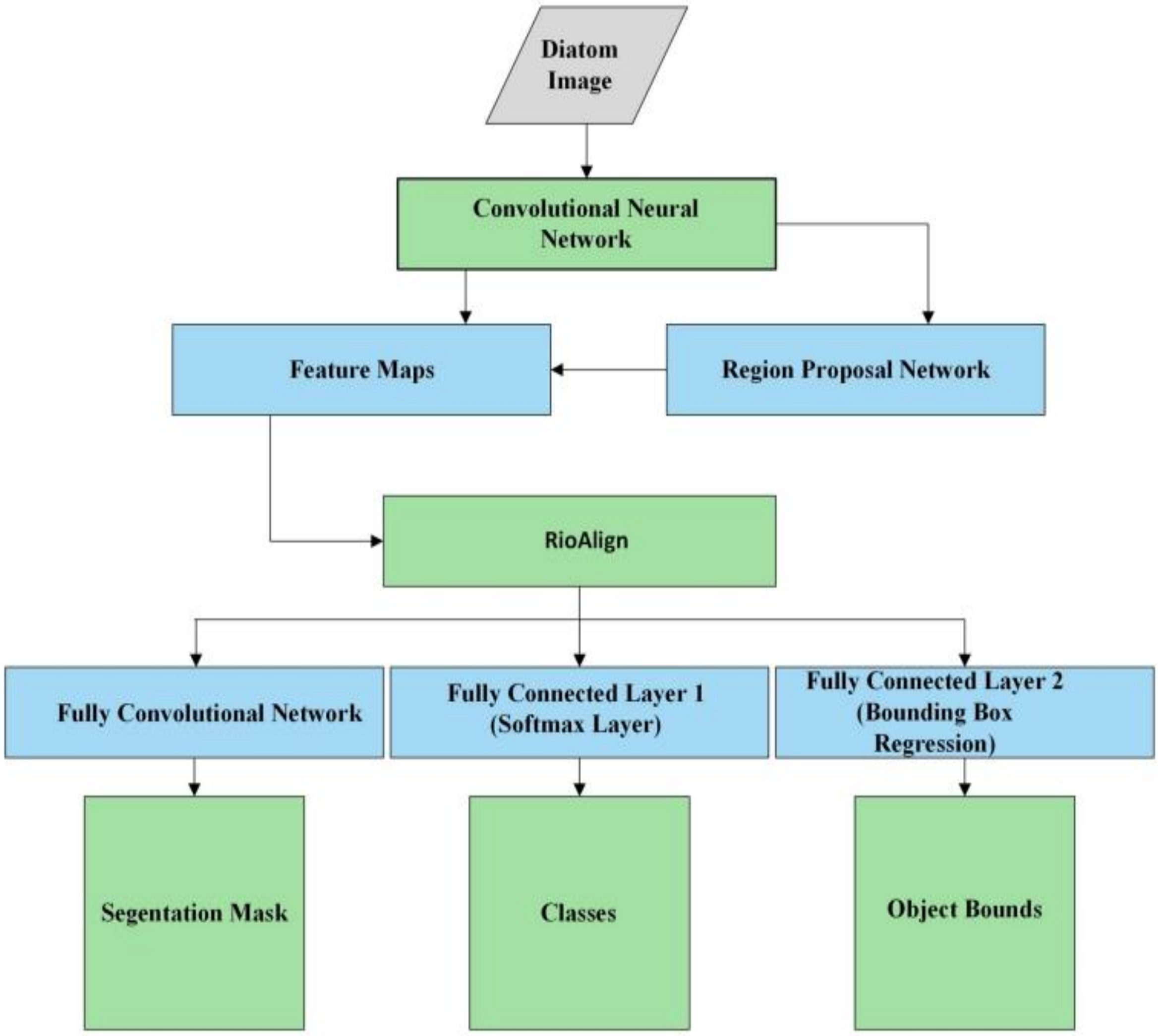

Diatom Image
Convolutional Neural Network
Feature Maps
Region Proposal Network
RioAlign
Fully Convolutional Network
Fully Connected Layer 1 (Softmax Layer)
Fully Connected Layer 2 (Bounding Box Regression)
Segentation Mask
Classes
Object Bounds

Fig. 9 Flowchart of Mask-RCNN for diatom image segmentation [87]

METHODOLOGY:

➢ Convolutional Neural Networks (CNNs)

Convolutional neural networks, or CNNs, are made to automatically and adaptably analyse input images and determine the spatial hierarchies of various features. In order to help in feature extraction and classification, they are made up of a number of layers, such as convolutional, pooling, and fully connected layers.

Algorithm:
A typical CNN algorithm consists of the following crucial steps:
● Convolutional Layer: Utilize the convolution technique to extract features.
● Activation Function: Implement ReLU activation to introduce nonlinearity.
● Pooling Layer: Downscale feature maps to decrease spatial dimensions.
● Fully Connected Layer: Condense and move through completely connected layers.
● SoftMax Layer: Probabilities of classification output.

Mathematical Model:
● Convolution operation:

$$h_{i,j}^{k} = \sum_{m=1}^{M} \square \sum_{n=1}^{N} \square\, W_{m,n}^{k} \cdot x_{i+m,j+n} + b^{k}$$

➢ Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) are made to process sequential data by storing information from earlier time steps in a hidden state. A variation of RNNs, Long Short-Term Memory (LSTM) networks enable the model to learn longterm dependencies by addressing the vanishing gradient problem.

Algorithm: LSTM Networks
The following steps are involved in an LSTM algorithm:
● **Input Gate**: Modify input states according to the input that is being received.
● **Forget Gate**: Discard any unnecessary data from earlier stages.
● **Cell State**: Utilizing input and historical state data, update the cell state.
● **Output Gate**: Generate output by using the most recent cell state.

Mathematical Model:
● **Input gate:**

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

● **Forget gate:**

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

● **Cell state update:**

$$C'_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * C'_t$$

● **Output gate:**

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

➢ Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are made up of two neural networks, a generator and a discriminator, that play a minimax game. The generator generates fake data because it is driven to offer more realistic data by the discriminator's attempt to distinguish between real and fake data.

Algorithm:
The following steps are involved in a GAN algorithm:
● **Generator:** Create fake samples by generating random noise.
● **Discriminator:** Determine whether samples are real or fake.
● **Loss Function:** Create an adversarial discriminator and generator.

Mathematical Model:
● **Generator:**

$$x' = G(z)$$

● **Discriminator:**

$$D(x) = \sigma(W_d \cdot x + b_d)$$

➢ Hybrid Models

Several neural network architectures are combined into hybrid models to address difficult challenges. CNNs are used for spatial feature extraction while RNNs are used for temporal dependency capture, for example, when CNNs and RNNs are combined .
Algorithm:
● **Architecture Fusion:** Combine various models' architectures (e.g., CNNs and RNNs).
● **Feature Fusion:** Merge features that have been taken out of various models or modalities.
● **Decision Fusion:** Voting or weighted averaging are used to combine predictions from several models.

Mathematical Model:
● **General Fusion Formula:**

$$y' = \alpha \cdot f_1(x) + (1 - \alpha) \cdot f_2(x)$$

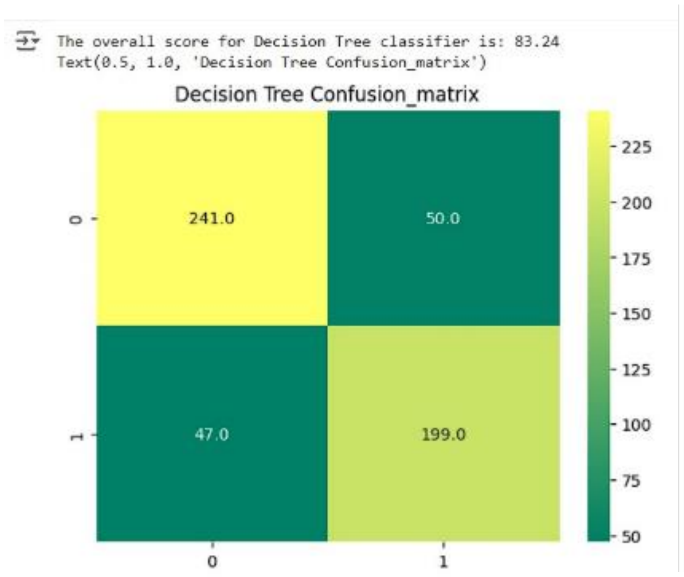
RESULTS AND DISCUSSION:

In this paper, we have reviewed the research done in implementing various ML techniques for image recognition of various microorganisms. After going through all the studies, it has been analyzed that many researchers have effectively used ML techniques to automate the traditional methods of microorganism classification and identification. Researchers have used different types of imaging techniques including microscopic imaging, hyper spectral imaging etc. for the identification, detection and classification of microorganisms. ML based research has also performed better on ZN stained sputum smear imaging for the identification of tuberculosis bacteria. This section aims to address the research questions (RQ2-RQ6) framed in section II, Subsection A

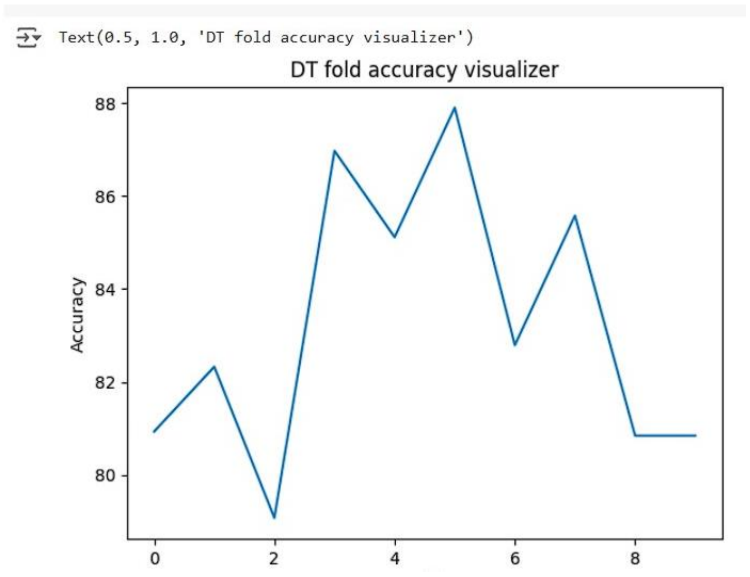
DECISION TREE CLASSIFIER (DT):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

hazel_df = pd.read_csv("file path.csv")

hazel_df.head()

#Feature selection
all_features = hazel_df.drop("CLASS",axis=1)
target_feature = hazel_df["CLASS"]
all_features.head()

#Dataset preprocessing
from sklearn import preprocessing
x = all_features.values.astype(float) #returns a numpy array of type float
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
scaled_features = pd.DataFrame(x_scaled)
scaled_features.head()


#Decision tree
from sklearn.model_selection import train_test_split #for split the data
from sklearn.metrics import accuracy_score  #for accuracy_score
from sklearn.model_selection import KFold #for K-fold cross validation
from sklearn.model_selection import cross_val_score #score evaluation
from sklearn.model_selection import cross_val_predict #prediction
from sklearn.metrics import confusion_matrix #for confusion matrix
import seaborn as sns

X_train,X_test,y_train,y_test =
train_test_split(scaled_features,target_feature,test_size=0.25,random_state=40)
X_train.shape,X_test.shape,y_train.shape,y_test.shape

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
model= DecisionTreeClassifier(criterion='gini',
                             min_samples_split=10,min_samples_leaf=1,
                             max_features='auto')
model.fit(X_train,y_train)
dt_pred=model.predict(X_test)
kfold = KFold(n_splits=10, random_state=None) # k=10, split the data into 10 equal parts
result_tree=cross_val_score(model,scaled_features,target_feature,cv=10,scoring='accuracy')
print('The overall score for Decision Tree classifier is:',round(result_tree.mean()*100,2))
y_pred = cross_val_predict(model,scaled_features,target_feature,cv=10)
sns.heatmap(confusion_matrix(dt_pred,y_test),annot=True,fmt=".1f",cmap='summer')
plt.title('Decision Tree Confusion_matrix')


#DT fold accuracy visualizer
_result_tree=[r*100 for r in result_tree]
plt.plot(_result_tree)
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.title('DT fold accuracy visualizer')

from sklearn.metrics import balanced_accuracy_score, accuracy_score, precision_score,
recall_score, f1_score
print('Micro Precision: {:.4f}'.format(precision_score(y_test, dt_pred, average='micro')))
print('Micro Recall: {:.4f}'.format(recall_score(y_test, dt_pred, average='micro')))
print('Micro F1-score: {:.4f}\n'.format(f1_score(y_test, dt_pred, average='micro')))

print('Macro Precision: {:.4f}'.format(precision_score(y_test, dt_pred, average='macro')))
print('Macro Recall: {:.4f}'.format(recall_score(y_test, dt_pred, average='macro')))
print('Macro F1-score: {:.4f}\n'.format(f1_score(y_test, dt_pred, average='macro')))

print('Weighted Precision: {:.4f}'.format(precision_score(y_test, dt_pred, average='weighted')))
print('Weighted Recall: {:.4f}'.format(recall_score(y_test, dt_pred, average='weighted')))
print('Weighted F1-score: {:.4f}'.format(f1_score(y_test, dt_pred, average='weighted')))

from sklearn.metrics import confusion_matrix, plot_confusion_matrix, classification_report
```

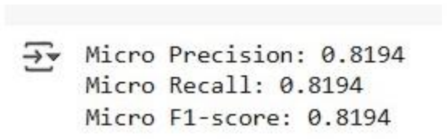
```
print('\n--------------- Decision Tree Classification Report ---------------\n')
print(classification_report(y_test, dt_pred))
#print('---------------------- Decison Tree ----------------------') # unnecessary fancy styling
```



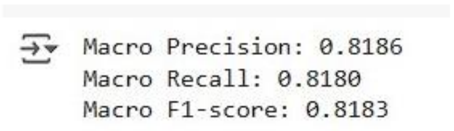
```
The overall score for Decision Tree classifier is: 83.24
Text(0.5, 1.0, 'Decision Tree Confusion_matrix')
```



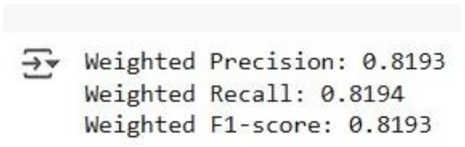
```
Text(0.5, 1.0, 'DT fold accuracy visualizer')
```



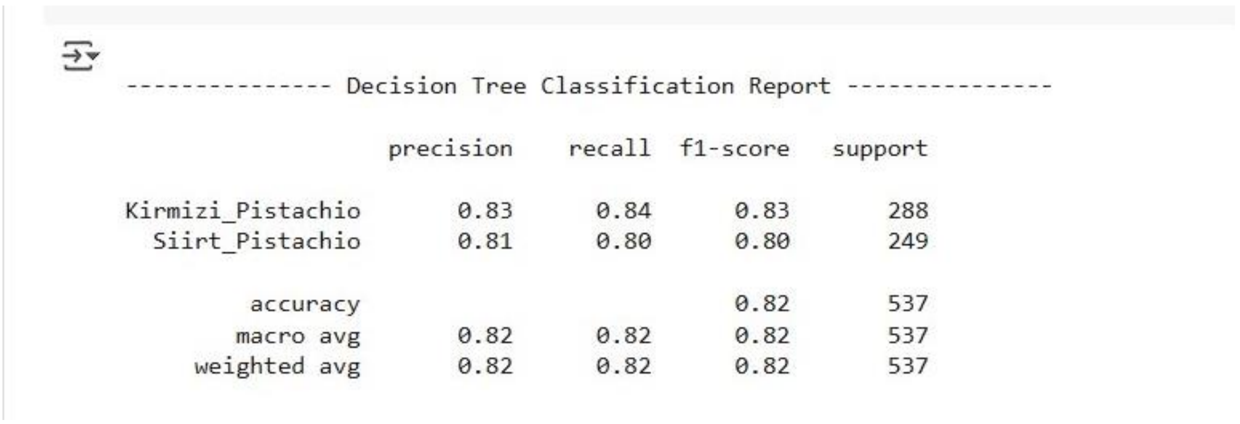
```
Micro Precision: 0.8194
Micro Recall: 0.8194
Micro F1-score: 0.8194
```



```
Macro Precision: 0.8186
Macro Recall: 0.8180
Macro F1-score: 0.8183
```



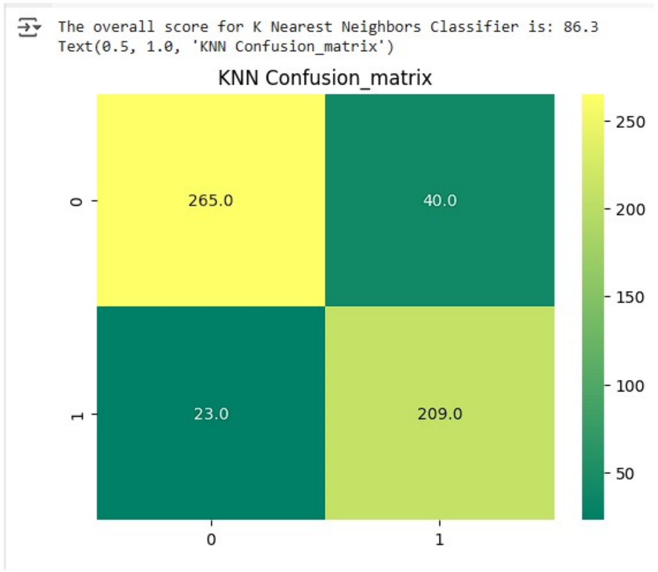
```
Weighted Precision: 0.8193
Weighted Recall: 0.8194
Weighted F1-score: 0.8193
```



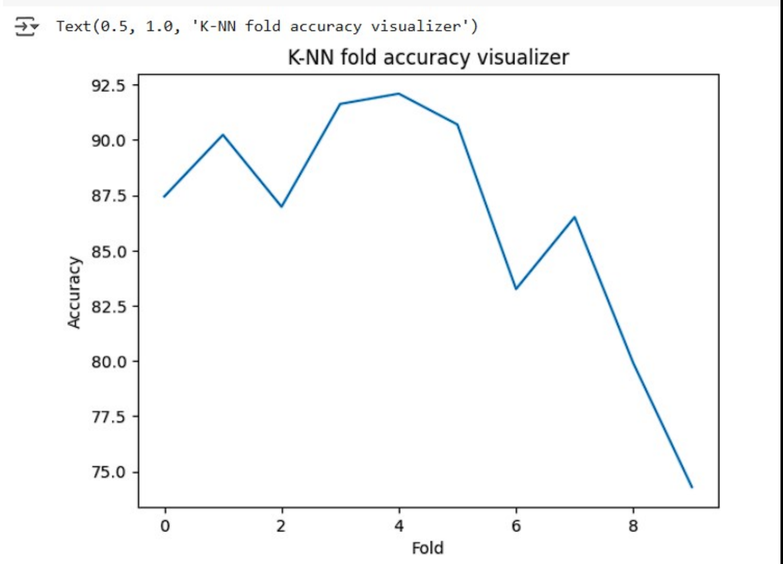
```
--------------- Decision Tree Classification Report ---------------

                   precision    recall  f1-score   support

Kirmizi_Pistachio       0.83      0.84      0.83       288
  Siirt_Pistachio       0.81      0.80      0.80       249

         accuracy                           0.82       537
        macro avg       0.82      0.82      0.82       537
     weighted avg       0.82      0.82      0.82       537
```

KNN CLASSIFIER:

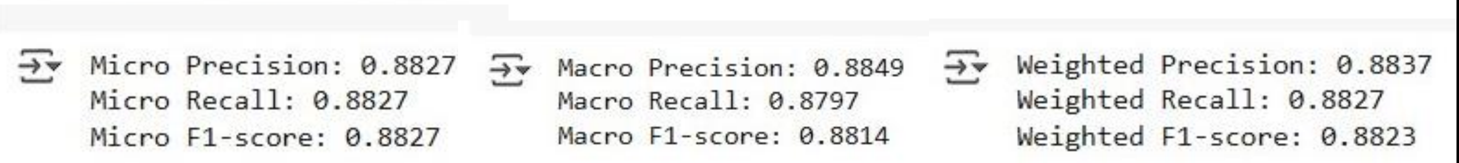
```
#KNN
from sklearn.model_selection import KFold #for K-fold cross validation
from sklearn.model_selection import cross_val_score #score evaluation
from sklearn.model_selection import cross_val_predict #prediction
from sklearn.metrics import confusion_matrix #for confusion matrix
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 25)
model.fit(X_train,y_train)
dt_knn=model.predict(X_test)
kfold = KFold(n_splits=10, random_state=None) # k=10, split the data into 10 equal parts=
result_knn=cross_val_score(model,scaled_features,target_feature,cv=kfold,scoring='accuracy')
print('The overall score for K Nearest Neighbors Classifier is:',round(result_knn.mean()*100,2))
y_pred = cross_val_predict(model,scaled_features,target_feature,cv=10)
sns.heatmap(confusion_matrix(dt_knn,y_test),annot=True,fmt=".1f",cmap='summer')
plt.title('KNN Confusion_matrix')


#KNN fold accuracy visualizer
_result_knn=[r*100 for r in result_knn]
plt.plot(_result_knn)
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.title('K-NN fold accuracy visualizer')


from sklearn.metrics import balanced_accuracy_score, accuracy_score, precision_score,
recall_score, f1_score
print('Micro Precision: {:.4f}'.format(precision_score(y_test, dt_knn, average='micro')))
print('Micro Recall: {:.4f}'.format(recall_score(y_test, dt_knn, average='micro')))
print('Micro F1-score: {:.4f}\n'.format(f1_score(y_test, dt_knn, average='micro')))

print('Macro Precision: {:.4f}'.format(precision_score(y_test, dt_knn, average='macro')))
print('Macro Recall: {:.4f}'.format(recall_score(y_test, dt_knn, average='macro')))
print('Macro F1-score: {:.4f}\n'.format(f1_score(y_test, dt_knn, average='macro')))

print('Weighted Precision: {:.4f}'.format(precision_score(y_test, dt_knn, average='weighted')))
print('Weighted Recall: {:.4f}'.format(recall_score(y_test, dt_knn, average='weighted')))
print('Weighted F1-score: {:.4f}'.format(f1_score(y_test, dt_knn, average='weighted')))
print('\n--------------- K-Nearest Neighbour Classification Report ---------------\n')
print(classification_report(y_test, dt_knn))
#print('---------------------- K-NN ----------------------') # unnecessary fancy styling
```



```
The overall score for K Nearest Neighbors Classifier is: 86.3
Text(0.5, 1.0, 'KNN Confusion_matrix')
```



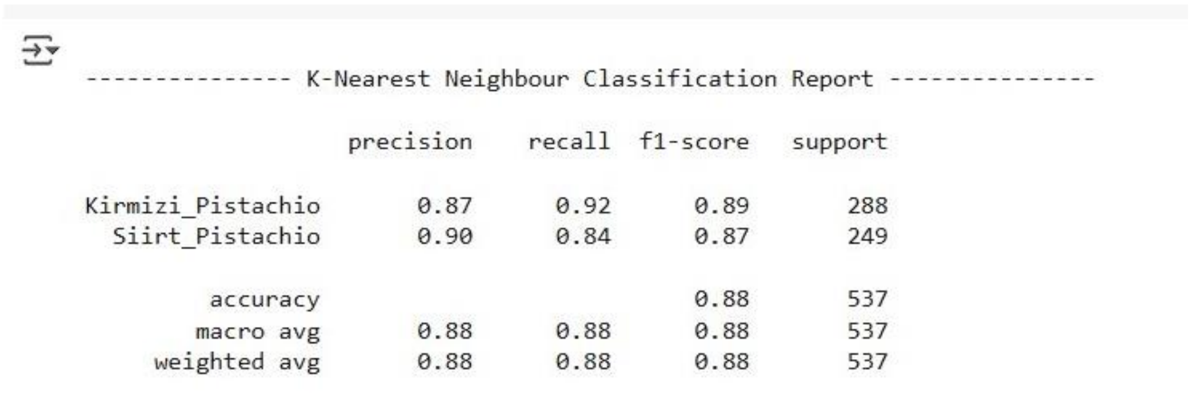
```
Text(0.5, 1.0, 'K-NN fold accuracy visualizer')
```



```
Micro Precision: 0.8827
Micro Recall: 0.8827
Micro F1-score: 0.8827
```

```
Macro Precision: 0.8849
Macro Recall: 0.8797
Macro F1-score: 0.8814
```

```
Weighted Precision: 0.8837
Weighted Recall: 0.8827
Weighted F1-score: 0.8823
```

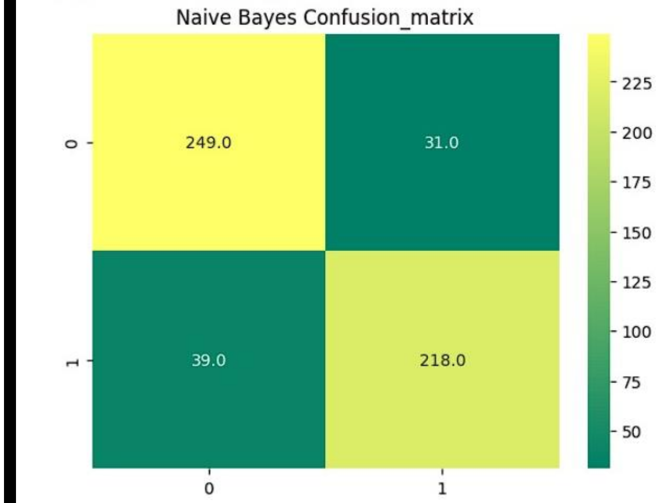


```
--------------- K-Nearest Neighbour Classification Report ---------------

                  precision    recall  f1-score   support

Kirmizi_Pistachio       0.87      0.92      0.89       288
  Siirt_Pistachio       0.90      0.84      0.87       249

         accuracy                           0.88       537
        macro avg       0.88      0.88      0.88       537
     weighted avg       0.88      0.88      0.88       537
```

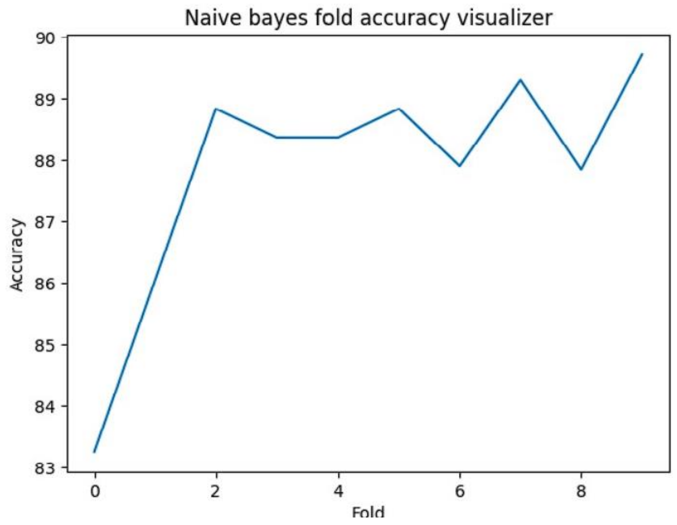
NAIVE BAYES:

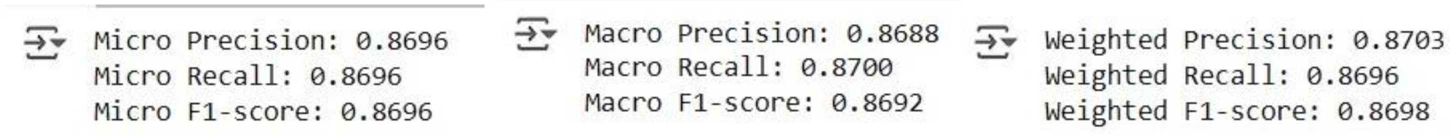
```
#Naive bayes
from sklearn.naive_bayes import GaussianNB
model= GaussianNB()
model.fit(X_train,y_train)
gnb_pred=model.predict(X_test)
kfold = KFold(n_splits=10, random_state=None) # k=10, split the data into 10 equal parts
result_gnb=cross_val_score(model,scaled_features,target_feature,cv=10,scoring='accuracy')
print('The overall score for Gaussian Naive Bayes classifier is:',round(result_gnb.mean()*100,2))
y_pred = cross_val_predict(model,scaled_features,target_feature,cv=10)
sns.heatmap(confusion_matrix(gnb_pred,y_test),annot=True,fmt=".1f",cmap='summer')
plt.title('Naive Bayes Confusion_matrix')


#Naive bayes fold accuracy visualizer
_result_gnb=[r*100 for r in result_gnb]
plt.plot(_result_gnb)
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.title('Accuracy')
plt.title('Naive bayes fold accuracy visualizer')


from sklearn.metrics import balanced_accuracy_score, accuracy_score, precision_score,
recall_score, f1_score
print('Micro Precision: {:.4f}'.format(precision_score(y_test, gnb_pred, average='micro')))
print('Micro Recall: {:.4f}'.format(recall_score(y_test, gnb_pred, average='micro')))
print('Micro F1-score: {:.4f}\n'.format(f1_score(y_test, gnb_pred, average='micro')))

print('Macro Precision: {:.4f}'.format(precision_score(y_test, gnb_pred, average='macro')))
print('Macro Recall: {:.4f}'.format(recall_score(y_test, gnb_pred, average='macro')))
print('Macro F1-score: {:.4f}\n'.format(f1_score(y_test, gnb_pred, average='macro')))


print('Weighted Precision: {:.4f}'.format(precision_score(y_test, gnb_pred, average='weighted')))
print('Weighted Recall: {:.4f}'.format(recall_score(y_test, gnb_pred, average='weighted')))
print('Weighted F1-score: {:.4f}'.format(f1_score(y_test, gnb_pred, average='weighted')))

print('\n---------------Naive Bayes Classification Report ---------------\n')
print(classification_report(y_test, gnb_pred))
#print('---------------------- Naive Bayes ----------------------') # unnecessary fancy styling
```

```
The overall score for Gaussian Naive Bayes classifier is: 87.85
Text(0.5, 1.0, 'Naive Bayes Confusion_matrix')
```



```
Text(0.5, 1.0, 'Naive bayes fold accuracy visualizer')
```

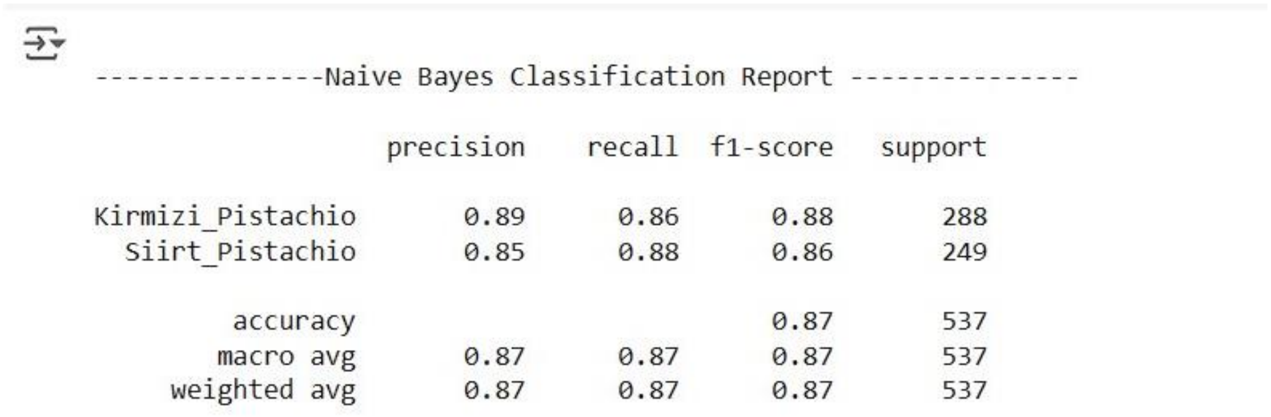




```
Micro Precision: 0.8696
Micro Recall: 0.8696
Micro F1-score: 0.8696
```

```
Macro Precision: 0.8688
Macro Recall: 0.8700
Macro F1-score: 0.8692
```

```
Weighted Precision: 0.8703
Weighted Recall: 0.8696
Weighted F1-score: 0.8698
```



```
---------------Naive Bayes Classification Report ---------------

                  precision    recall  f1-score   support

Kirmizi_Pistachio       0.89      0.86      0.88       288
  Siirt_Pistachio       0.85      0.88      0.86       249

         accuracy                           0.87       537
        macro avg       0.87      0.87      0.87       537
     weighted avg       0.87      0.87      0.87       537
```

CONCLUSION AND FUTURE SCOPE:

The integration of machine learning (ML) in microbiology, particularly for microorganism image analysis, holds great potential to enhance classification and identification accuracy. Despite challenges like limited publicly available datasets and performance degradation across diverse datasets, deep learning (DL) techniques have shown promise. Convolutional Neural Networks (CNNs) are highly effective for static image recognition, offering high accuracy and efficiency, while Recurrent Neural Networks (RNNs) excel with sequential data, albeit with higher computational demands. Generative Adversarial Networks (GANs) are valuable for generative tasks but require stable training. Hybrid models, combining multiple techniques, offer superior robustness and accuracy for complex tasks, though they come with increased processing needs. Future research should focus on developing more diverse datasets, optimizing DL techniques, and exploring hybrid and swarm intelligence-based approaches to push the boundaries of microorganism classification. Cross-disciplinary collaboration between informatics, medicine, and biology is essential to advance ML applications in microbiology and improve image recognition technologies.

Future scope Future research should focus on:

- **Efficiency Improvement:** RNNs and GANs' computational efficiency can be increased by improved architectures and hardware developments.
- **Training Stability:** Use cutting-edge strategies and loss functions to increase GAN training stability.
- **Hybrid Model Optimization:** Reduce complexity without sacrificing functionality in hybrid models.
- **Transfer Learning and Multitask Learning:** Applying knowledge across domains and empowering models to execute various tasks reduces training time and data needs.
- **Robustness and Generalization:** Make sure models can adapt well to different datasets and withstand adversarial attacks.
- **Interpretable and Explainable AI:** Provide techniques for improving the interpretability and comprehensibility of models.
- **Real-Time Processing:** Sophisticated models enable effective real-time picture identification, particularly in applications such as driverless cars and video surveillance.

REFRENCES:

➢ Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification withss deep convolutional neural networks. Advances in neural information processing systems, 25.

➢ Arjun Uddagiri,Pragada Eswar,Tummu Vineetha,"Enhancing Mobile security with Automated sim slot ejection system and authentication mechanism",2023

➢ Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2625-2634).

➢ Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. Advances in neural information processing systems, 27.

➢ Sharif Razavian, A., Azizpour, H., Sullivan, J. and Carlsson, S., 2014. CNN features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).

➢ Microorganism—Wikipedia (2021) En.wikipedia.org. https://en. wikipedia.org/wiki/Microorganism. Accessed: 20 Apr 2021

➢ COVID-19—Wikipedia (2021) En.wikipedia.org. https://en. wikipedia.org/wiki/Coronavirus_disease_2019. Accessed 20 Apr 2021

➢ Franco-Duarte R et al (2019) Advances in chemical and biological methods to identify microorganisms—from past to present. Microorganisms 7(5):130. https://doi.org/10.3390/microorgan isms7050130

➢ Londhe ND, Ahirwal MK, Lodha P (2016) Machine learning paradigms for speech recognition of an Indian dialect. In: 2016 international conference on communication and signal processing (ICCSP), pp 0780–0786. https://doi.org/10.1109/ICCSP. 2016.7754251

➢ Liu F, Yan J, Wang W, Liu J, Li J et al (2020) Scalable skin lesion multi-classifcation recognition system. Comput Mater Contin 62(2):801–816