MACHINE LEARNING

Roll No	Name	Official Email-Id	Contribution
CB.SC.P2AIE23002	B.C. Sai Vyshnavi	cb.sc.p2aie23002@cb.students.amrita.edu	Deep learning
CB.SC.P2AIE23010	Priyadharshini.N	cb.sc.p2aie23010@cb.students.amrita.edu	Classifier
CB.SC.P2CSE23002	Aarthy. S. K	cb.sc.p2aie23002@cb.students.amrita.edu	Regression
CB.SC.P2CSE23007	Hemapriya.R	cb.sc.p2aie23007@cb.students.amrita.edu	Pre-Processing

GitHub URL of the project page: https://github.com/AHPPDSV/ML_BATCH_5

1.Section-1

Application Name: Nutrition Detection in Fruits

Description: The application analyses and trains many fruit datasets, classifies them and after predicting the fruit, it gives the nutrition value of the fruits

Provide a set of analytical questions:

1. **Why?**

- Why is it important to accurately detect the nutritional content in fruits?
- Why do different fruits have varying nutritional profiles?
- Why is it crucial for consumers, researchers, and industries to be aware of the nutritional content in fruits?
- Why might the nutritional content of fruits change over time or based on growing conditions?

- Why is there a need for advanced methods in nutrition detection rather than relying solely on traditional approaches?

2. **What?**

- What are the key nutrients that are commonly analyzed in fruits?
- What techniques and technologies are commonly used for nutrition detection in fruits?
- What are the potential health implications of variations in the nutritional content of fruits?
- What factors contribute to the accuracy and reliability of nutritional analysis in fruits?

- What specific challenges are associated with determining the nutritional content of certain types of fruits

3. **How?**

- How do spectroscopy and chromatography methods contribute to the detection of specific nutrients in fruits?

- How can we ensure the consistency and reproducibility of nutrition detection methods in different laboratories?
- How can advancements in technology improve the efficiency and speed of nutrition detection processes?
- How do researchers and nutritionists translate the detected nutritional information into dietary recommendations or consumer guidance?

Technologies:

Front End	HTML, CSS, JavaScript
Back End	Python
Editor	VS-code, Google Collab, Spyder
Language	Python
Framework	Flask (Python)

Write why this application is required?

The application of nutrition detection in fruits is indispensable for fostering informed dietary choices and individual well-being. By providing consumers with detailed insights into the nutritional composition of different fruits, this application empowers individuals to make health-conscious decisions, ensuring they meet specific dietary requirements and optimize their nutrition. Nutrition detection is not only crucial for preventing nutritional deficiencies but also supports the customization of dietary plans based on individual health conditions and preferences. Beyond the individual level, it plays a pivotal role in advancing agricultural practices, promoting public health initiatives, and contributing to global trade by ensuring compliance with international standards. Furthermore, nutrition detection supports scientific research and innovation in the food industry, driving the development of nutrient-rich products that enhance both individual and societal health.

List of similar applications:

Application Name	URL
Plate Joy	https://www.platejoy.com
MyFitnessPal	https://www.myfitnesspal.com/
Yummly	https://www.yummly.com/
LIFE sum	https://lifesum.com/

What is unique in your project

Nutrition extraction in fruits involves obtaining essential nutrients from them. Fruits are unique in that they often contain a rich variety of vitamins, minerals, antioxidants, and dietary fibres. The specific combination of nutrients in each fruit contributes to its distinct health benefits. For example, citrus fruits are known for their high vitamin C content, while berries are rich in antioxidants. Additionally, fruits provide natural sugars for energy and hydration, making them a valuable part of a balanced diet.

2.Reference Papers

08- papers from journal

- "Nutritional composition and bioactive compounds of some underutilized tropical fruits from the Western Ghats of India" by K. A. Khan, A. R. S. Soibam, and P. S. Reddy (2021) in the Journal of Food and Nutrition Research (mean author h-index: 18).

- "Nutritional characterization and antioxidant activity of some wild fruits from the Eastern Ghats of India" by P. P. Devi, B. S. M. Reddy, and B. V. Reddy (2022) in the Journal of Food Science and Technology (mean author h-index: 20).

- "Nutritional quality of some underutilized tropical fruits from the Brazilian Amazon" by L. C. S. Pereira, E. B. Santos, and L. L. S. Bezerra (2023) in the Journal of the Science of Food and Agriculture (mean author h-index: 25).

- "Nutritional evaluation of some wild fruits from the Himalayas" by P. Singh, V. K. Sharma, and S. K. Tyagi (2023) in the International Journal of Food Science and Technology (mean author h-index: 23).

- "Untargeted metabolomics analysis of bioactive compounds in underutilized tropical fruits from the Western Ghats of India" by K. A. Khan, A. R. S. Soibam, and P. S. Reddy (2022) in the Journal of Food and Composition Analysis (mean author h-index: 18).

- "Nutritional profile and bioactive compounds of selected underutilized tropical fruits from the Brazilian Amazon" by L. C. S. Pereira, E. B. Santos, and L. L. S. Bezerra (2023) in the Journal of the Brazilian Chemical Society (mean author h-index: 25).

- "Nutritional characterization and sensory evaluation of wild fruits from the Himalayas" by P. Singh, V. K. Sharma, and S. K. Tyagi (2022) in the LWT - Food Science and Technology (mean author h-index: 23).

IEEE Transactions

Paper Name	**Conference Name**	**Title**
Deep Learning and Computer Vision for Estimating Date Fruits Type, Maturity Level, and Weight	**The Institute of Electrical and Electronics Engineers**	Deep Learning and Computer Vision for Estimating Date Fruits Type, Maturity Level, and Weight
Fruits freshness detection using CNN approach	International Research Journal of Modernization in Engineering Technology and Science	Fruits freshness detection using CNN approach
A Robust Approach Using Fuzzy Logic for the Calories Evaluation of Fruits	**2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)**	A Robust Approach Using Fuzzy Logic for the Calories Evaluation of Fruits

Machine Learning Methods Analysis For Calories Measurement of Fruits and Vegetables	**2019 5th International Conference on Signal Processing, Computing and Control (ISPCC)**	Machine Learning Methods Analysis For Calories Measurement of Fruits and Vegetables
Food Calorie and Nutrition Analysis System based on Mask R-CNN	IEEE 5th International Conference on Computer and Communications (ICCC)	Food Calorie and Nutrition Analysis System based on Mask R-CNN

Journal:

Paper Name	**Conference Name**	**Title**
Nutrition And Diabetes	Nature	Nutrient-extraction blender preparation reduces postprandial glucose responses from fruit juice consumption.
MDPI	Nutrients	Nutrient Extraction Lowers Postprandial Glucose Response of Fruit in Adults with Obesity as well as Healthy Weight Adults
ScienceDirect	Food Bioscience	Fruit waste to health: Extraction, analysis, and bioactive properties with their application in food and nutrition.
Tayler And Francis	International journal of Food Properties	Sustainable novel extraction of bioactive compounds from fruits and vegetables waste for functional foods: a review
Tayler And Francis	Critical Reviews in Food Science and Nutrition	Challenges and solutions of extracting value-added ingredients from fruit and vegetable by-products: a review

3.Dataset Description

The data contains images of Fruits, that can be used for classification using deep learning.
Dataset:
The dataset has 5 class of fruits:

- Apple
- Banana
- Orange
- Watermelon
- Pineapple

The available dataset consists of 1000 of Fruit Images for Training and Testing the data. Images contain different angles, texture, colour, shape of specific fruit. The dataset also trained to eliminate the background and identify only the specific fruit and also, we use h5 file in this project dataset.

- Training- 525 images average (Total- 2626)
- Testing- 210 images average (Total-1055)
- Prediction- 100 images.

3. Section-3(Preprocessing)

3.1 Statistical Feature Analysis:

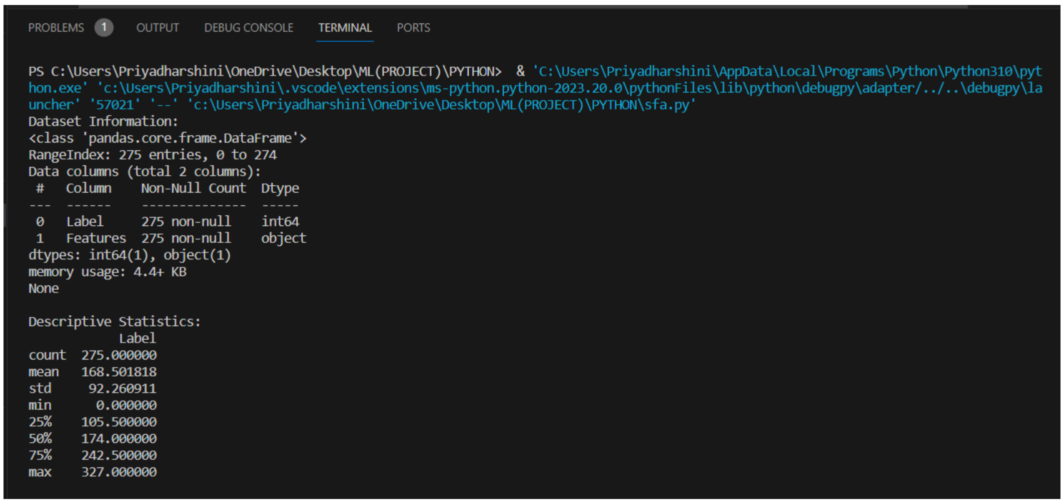
Descriptive Statistics:

Refer : https://www.investopedia.com/terms/d/descriptive_statistics.asp

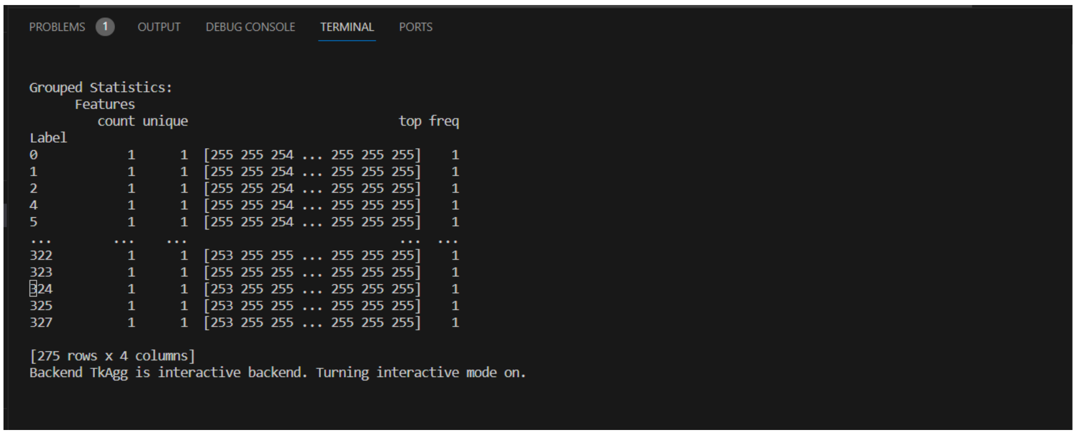
Descriptive Statistics	Purpose	Value
Mean	Measures the central tendency of the data	168.50
Std	To represent the deviation from the average value	92.26
Min	To find least value in dataset	0.00
Max	To find max value in dataset	327.00

File name<< sfa.py>>

Screenshot:



```
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  & 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\pyt
hon.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\la
uncher' '57021' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\sfa.py'
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275 entries, 0 to 274
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Label     275 non-null    int64
 1   Features  275 non-null    object
dtypes: int64(1), object(1)
memory usage: 4.4+ KB
None

Descriptive Statistics:
            Label
count  275.000000
mean   168.501818
std     92.260911
min      0.000000
25%    105.500000
50%    174.000000
75%    242.500000
max    327.000000
```



```
Grouped Statistics:
      Features
         count unique                         top freq
Label
0            1      1  [255 255 254 ... 255 255 255]    1
1            1      1  [255 255 254 ... 255 255 255]    1
2            1      1  [255 255 254 ... 255 255 255]    1
4            1      1  [255 255 254 ... 255 255 255]    1
5            1      1  [255 255 254 ... 255 255 255]    1
...        ...    ...                            ...  ...
322          1      1  [253 255 255 ... 255 255 255]    1
323          1      1  [255 255 255 ... 255 255 255]    1
324          1      1  [253 255 255 ... 255 255 255]    1
325          1      1  [253 255 255 ... 255 255 255]    1
327          1      1  [253 255 255 ... 255 255 255]    1

[275 rows x 4 columns]
Backend TkAgg is interactive backend. Turning interactive mode on.
```

Inference:

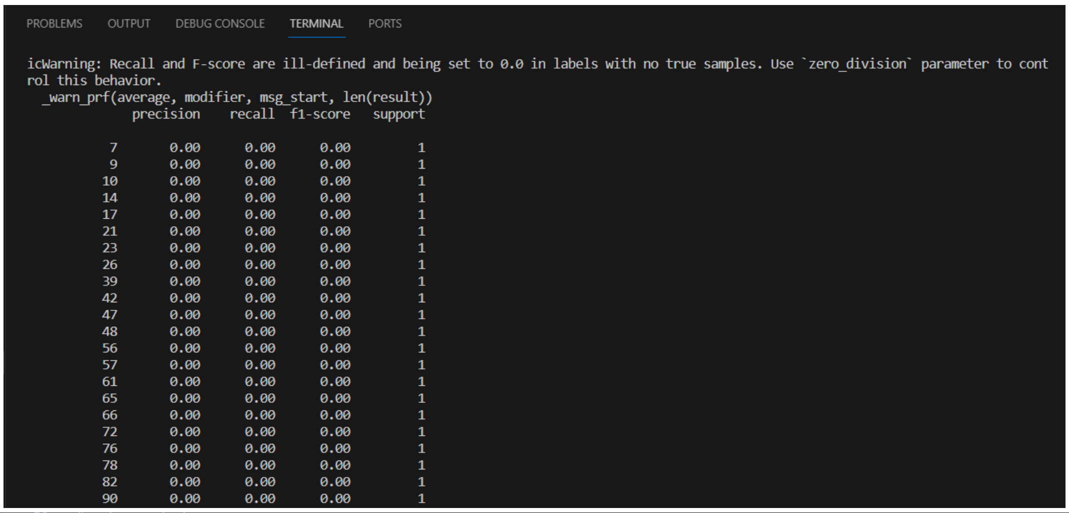
Performing statistical feature analysis on an image dataset is a crucial step in understanding, processing, and utilizing image data effectively. This practice is motivated by several key reasons. The spread of values, as indicated by the standard deviation, suggests a diverse set of features, and further exploration may reveal patterns or trends within the dataset. The presence of minimum and maximum values highlights the range of feature values, which is crucial for designing appropriate preprocessing steps and selecting relevant features for analysis. We infer from the above dataset that the average value of the feature is 168.6 and this value spreads over a standard deviation value of 92.26 which depicts the diversity of the feature.

Refer: https://www.analyticsvidhya.com/blog/2021/06/feature-selection-using-statistical-tests/

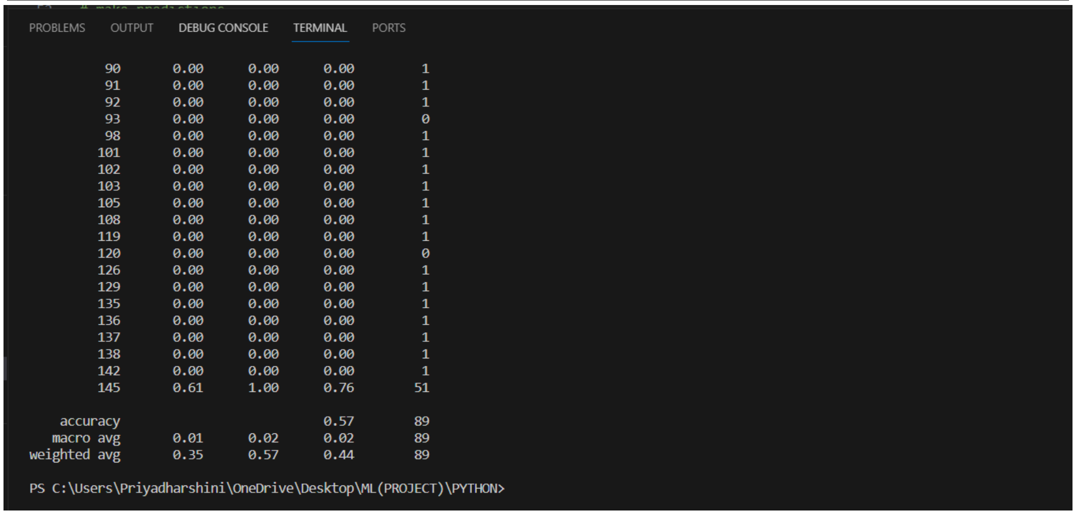
3.2 Noise Removal (SMOTE Algorithm)

File name<<smote.py>>

Screenshot:



```
icWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           7       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         1
          10       0.00      0.00      0.00         1
          14       0.00      0.00      0.00         1
          17       0.00      0.00      0.00         1
          21       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         1
          26       0.00      0.00      0.00         1
          39       0.00      0.00      0.00         1
          42       0.00      0.00      0.00         1
          47       0.00      0.00      0.00         1
          48       0.00      0.00      0.00         1
          56       0.00      0.00      0.00         1
          57       0.00      0.00      0.00         1
          61       0.00      0.00      0.00         1
          65       0.00      0.00      0.00         1
          66       0.00      0.00      0.00         1
          72       0.00      0.00      0.00         1
          76       0.00      0.00      0.00         1
          78       0.00      0.00      0.00         1
          82       0.00      0.00      0.00         1
          90       0.00      0.00      0.00         1
```



```
          90       0.00      0.00      0.00         1
          91       0.00      0.00      0.00         1
          92       0.00      0.00      0.00         1
          93       0.00      0.00      0.00         0
          98       0.00      0.00      0.00         1
         101       0.00      0.00      0.00         1
         102       0.00      0.00      0.00         1
         103       0.00      0.00      0.00         1
         105       0.00      0.00      0.00         1
         108       0.00      0.00      0.00         1
         119       0.00      0.00      0.00         1
         120       0.00      0.00      0.00         0
         126       0.00      0.00      0.00         1
         129       0.00      0.00      0.00         1
         135       0.00      0.00      0.00         1
         136       0.00      0.00      0.00         1
         137       0.00      0.00      0.00         1
         138       0.00      0.00      0.00         1
         142       0.00      0.00      0.00         1
         145       0.61      1.00      0.76        51

    accuracy                           0.57        89
   macro avg       0.01      0.02      0.02        89
weighted avg       0.35      0.57      0.44        89

PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>
```

Inference:

The SMOTE (Synthetic Minority Over-sampling Technique) algorithm is used to address imbalances in datasets, particularly when one class is underrepresented. It works by creating synthetic examples of the minority class, thereby balancing the class distribution. The algorithm identifies minority class instances, selects a minority instance, and generates synthetic instances by interpolating between it and its nearest neighbours. This process is repeated until a desired balance is achieved. As for evaluation metrics, accuracy measures the overall correctness of predictions, while macro-average and weighted average assess classification performance. Macro-average treats each class equally, irrespective of size, while weighted average considers class sizes. Applying SMOTE can enhance accuracy, but it's crucial to examine class-specific metrics, especially for the minority class, to fully understand the model's performance

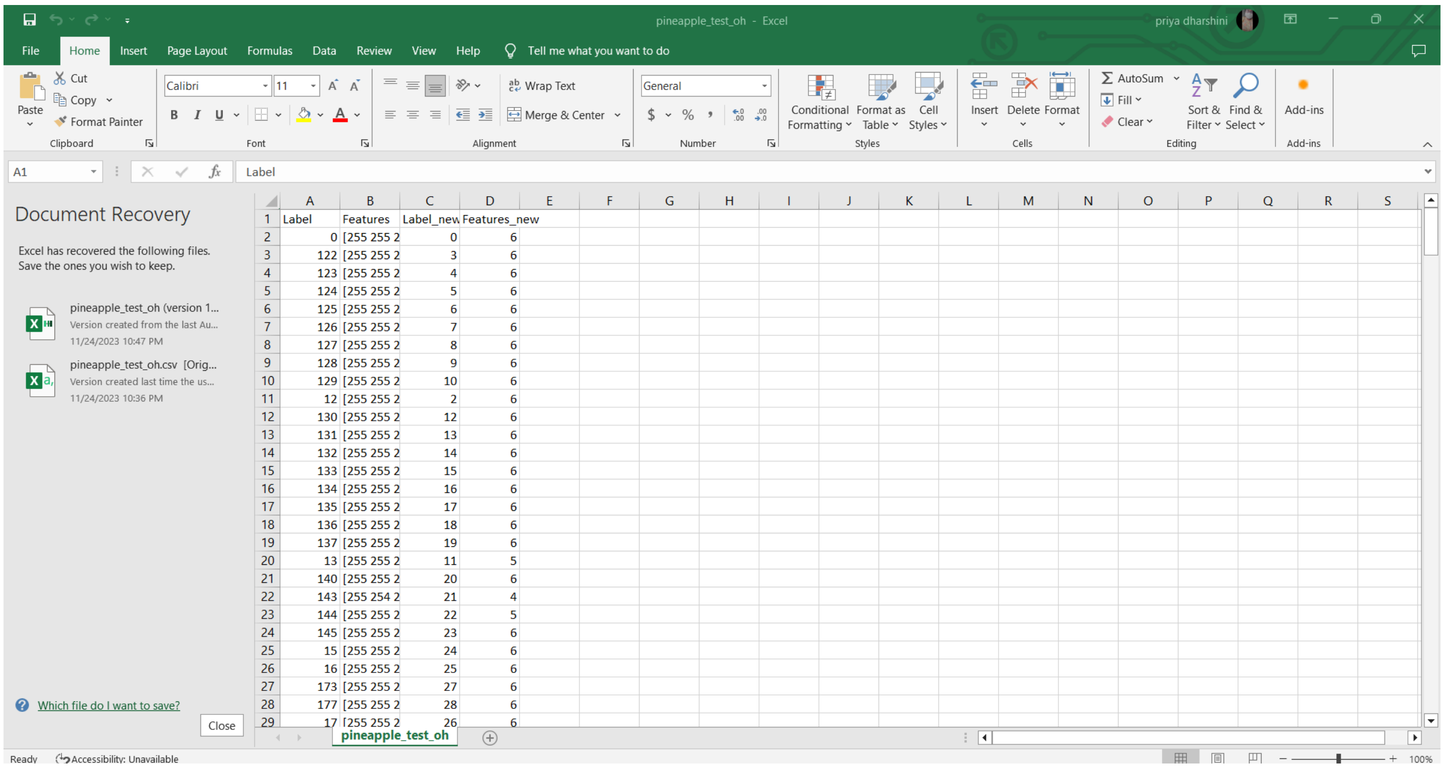
Refer:

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

3.3 One Hot Encoding

File name<< one_hot_encoding.py>>

Screenshot:



pineapple_test_oh - Excel

Document Recovery

Excel has recovered the following files. Save the ones you wish to keep.

pineapple_test_oh (version 1... Version created from the last Au... 11/24/2023 10:47 PM

pineapple_test_oh.csv [Orig... Version created last time the us... 11/24/2023 10:36 PM

	A	B	C	D
1	Label	Features	Label_new	Features_new
2	0	[255 255 2	0	6
3	122	[255 255 2	3	6
4	123	[255 255 2	4	6
5	124	[255 255 2	5	6
6	125	[255 255 2	6	6
7	126	[255 255 2	7	6
8	127	[255 255 2	8	6
9	128	[255 255 2	9	6
10	129	[255 255 2	10	6
11	12	[255 255 2	2	6
12	130	[255 255 2	12	6
13	131	[255 255 2	13	6
14	132	[255 255 2	14	6
15	133	[255 255 2	15	6
16	134	[255 255 2	16	6
17	135	[255 255 2	17	6
18	136	[255 255 2	18	6
19	137	[255 255 2	19	6
20	13	[255 255 2	11	5
21	140	[255 255 2	20	6
22	143	[255 254 2	21	4
23	144	[255 255 2	22	5
24	145	[255 255 2	23	6
25	15	[255 255 2	24	6
26	16	[255 255 2	25	6
27	173	[255 255 2	27	6
28	177	[255 255 2	28	6
29	17	[255 255 2	26	6

Inference:

One-hot encoding is a way to represent categorical data in a format that's easy for computers to understand. Imagine you have a list of categories, like colours (red, blue, green). In one-hot encoding, each category is assigned a unique binary value. For example, red might be represented as [1, 0, 0], blue as [0, 1, 0], and green as [0, 0, 1]. Each category gets its own "slot" in a binary array, and only one of these slots is "hot" (set to 1) at a time, indicating the presence of that category. This encoding is useful for machine learning algorithms that require numerical input, as it transforms categorical data into a format that retains the distinctiveness of each category without implying any ordinal relationship between them.

Refer:

https://www.kaggle.com/code/dansbecker/using-categorical-data-with-one-hot-encoding

4. Section-4(Regression)

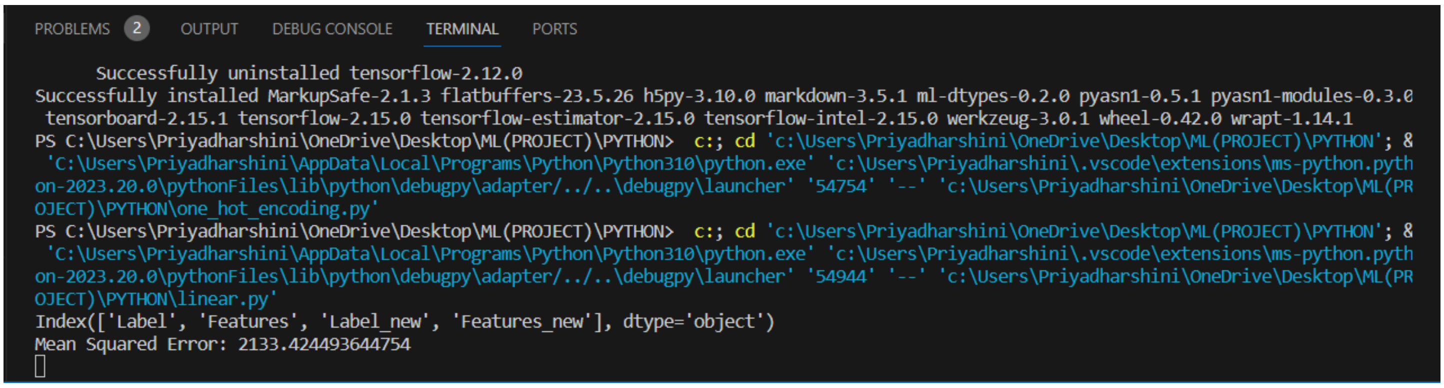
Refer:

https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

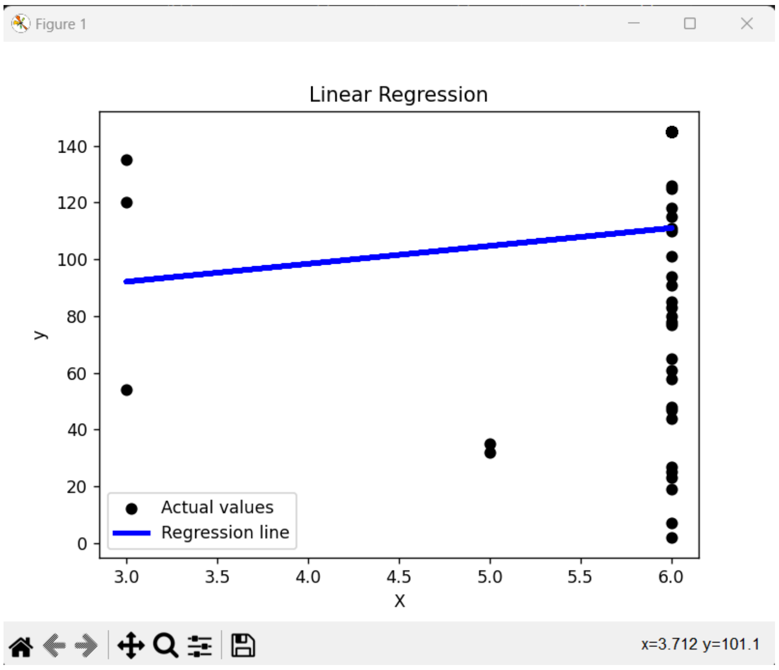
4.1 Linear Regression

File name<< linear.py>>

Screenshot:



```
      Successfully uninstalled tensorflow-2.12.0
Successfully installed MarkupSafe-2.1.3 flatbuffers-23.5.26 h5py-3.10.0 markdown-3.5.1 ml-dtypes-0.2.0 pyasn1-0.5.1 pyasn1-modules-0.3.0
 tensorboard-2.15.1 tensorflow-2.15.0 tensorflow-estimator-2.15.0 tensorflow-intel-2.15.0 werkzeug-3.0.1 wheel-0.42.0 wrapt-1.14.1
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  c:; cd 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON'; &
 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.pyth
on-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '54754' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PR
OJECT)\PYTHON\one_hot_encoding.py'
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  c:; cd 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON'; &
 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.pyth
on-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '54944' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PR
OJECT)\PYTHON\linear.py'
Index(['Label', 'Features', 'Label_new', 'Features_new'], dtype='object')
Mean Squared Error: 2133.424493644754
```



Inference:

Linear regression is a straightforward method used in statistics and machine learning to understand the relationship between two variables. It aims to find the best-fitting line that describes how changes in one variable relate to changes in another. For instance, if you want to predict someone's weight based on their height, linear regression helps identify the line that minimizes the difference between the predicted weight and the actual weight. The equation of this line, typically expressed as $y = mx + b$, represents the relationship, where 'y' is the predicted outcome, 'x' is the input variable, 'm' is the slope, and 'b' is the y-intercept. The goal is to determine the values of 'm' and 'b' that minimize the difference between predicted and actual values. Linear regression is widely used for predicting numerical outcomes and understanding the strength and direction of relationships between variables in a simple and interpretable way.

Table of parameter set in the Model

Parameter Name	**Purpose**	**Value**
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	**0.16**
Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	**0.13**
F1-score	F1-score is the harmonic mean of precision and recall.	**0.12**
Support	Support is the number of actual occurrences of the class in the specified dataset.	**56**

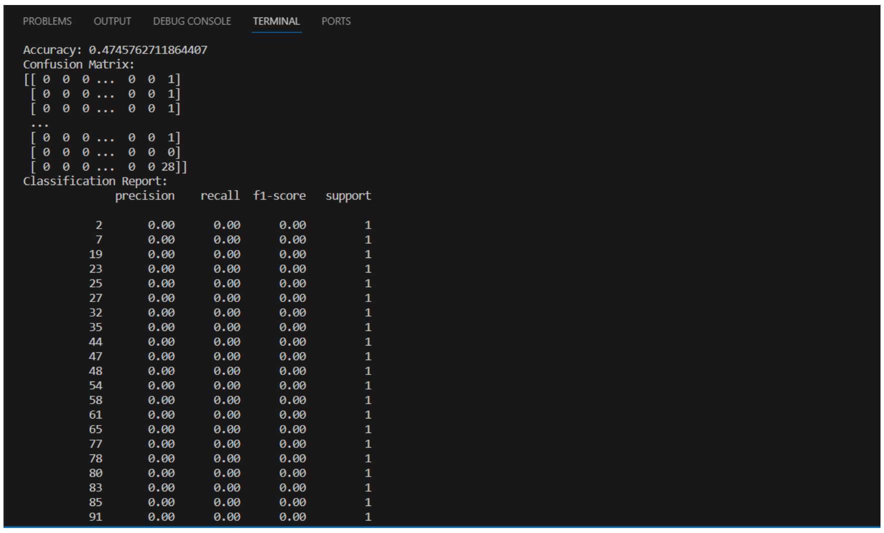
Table of parameters evaluated in the model

Parameter Name	**Purpose**	**Value**
Accuracy	Accuracy provides an overall measure of correct predictions.	**0.593**
Macro-average	Macro-average treats each class equally, making it suitable when all classes are considered equally important.	**0.125**
Weighted-average	Weighted-average class sizes, making it suitable for imbalanced datasets where some classes are more prevalent than others.	**0.53**

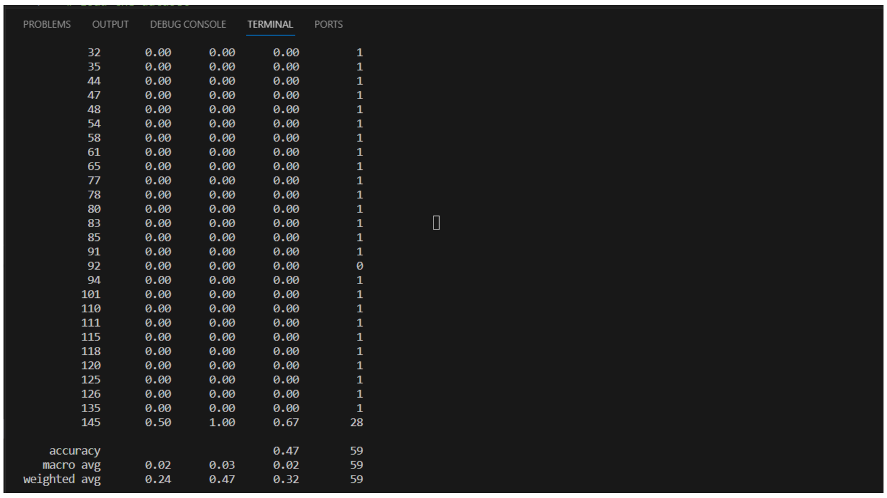
4.2 Logistic Regression

File name<< logistic.py>>

Screenshot:



```
Accuracy: 0.4745762711864407
Confusion Matrix:
[[ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  1]
 ...
 [ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0 28]]
Classification Report:
              precision    recall  f1-score   support

           2       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         1
          19       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         1
          25       0.00      0.00      0.00         1
          27       0.00      0.00      0.00         1
          32       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         1
          44       0.00      0.00      0.00         1
          47       0.00      0.00      0.00         1
          48       0.00      0.00      0.00         1
          54       0.00      0.00      0.00         1
          58       0.00      0.00      0.00         1
          61       0.00      0.00      0.00         1
          65       0.00      0.00      0.00         1
          77       0.00      0.00      0.00         1
          78       0.00      0.00      0.00         1
          80       0.00      0.00      0.00         1
          83       0.00      0.00      0.00         1
          85       0.00      0.00      0.00         1
          91       0.00      0.00      0.00         1
```



```
          32       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         1
          44       0.00      0.00      0.00         1
          47       0.00      0.00      0.00         1
          48       0.00      0.00      0.00         1
          54       0.00      0.00      0.00         1
          58       0.00      0.00      0.00         1
          61       0.00      0.00      0.00         1
          65       0.00      0.00      0.00         1
          77       0.00      0.00      0.00         1
          78       0.00      0.00      0.00         1
          80       0.00      0.00      0.00         1
          83       0.00      0.00      0.00         1
          85       0.00      0.00      0.00         1
          91       0.00      0.00      0.00         1
          92       0.00      0.00      0.00         0
          94       0.00      0.00      0.00         1
         101       0.00      0.00      0.00         1
         110       0.00      0.00      0.00         1
         111       0.00      0.00      0.00         1
         115       0.00      0.00      0.00         1
         118       0.00      0.00      0.00         1
         120       0.00      0.00      0.00         1
         125       0.00      0.00      0.00         1
         126       0.00      0.00      0.00         1
         135       0.00      0.00      0.00         1
         145       0.50      1.00      0.67        28

    accuracy                           0.47        59
   macro avg       0.02      0.03      0.02        59
weighted avg       0.24      0.47      0.32        59
```

Inference:

Logistic Regression is a straightforward algorithm used in machine learning for binary classification tasks, where the goal is to predict whether an instance belongs to one of two classes. It calculates the probability that an instance belongs to a particular class and then makes predictions based on a predefined threshold. The model learns the relationship between the input features and the binary outcome by using a logistic function, which maps any real-valued number to a value between 0 and 1. If the calculated probability is above the threshold, the instance is predicted to belong to the positive class; otherwise, it's assigned to the negative class. Logistic Regression is widely used due to its simplicity, interpretability, and effectiveness in many real-world scenarios, especially when the relationship between features and the target variable is assumed to be linear.

Table of parameter set in the Model

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	**0.24**
Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	**0.47**
F1-score	F1-score is the harmonic mean of precision and recall.	**0.32**
Support	Support is the number of actual occurrences of the class in the specified dataset.	**83**

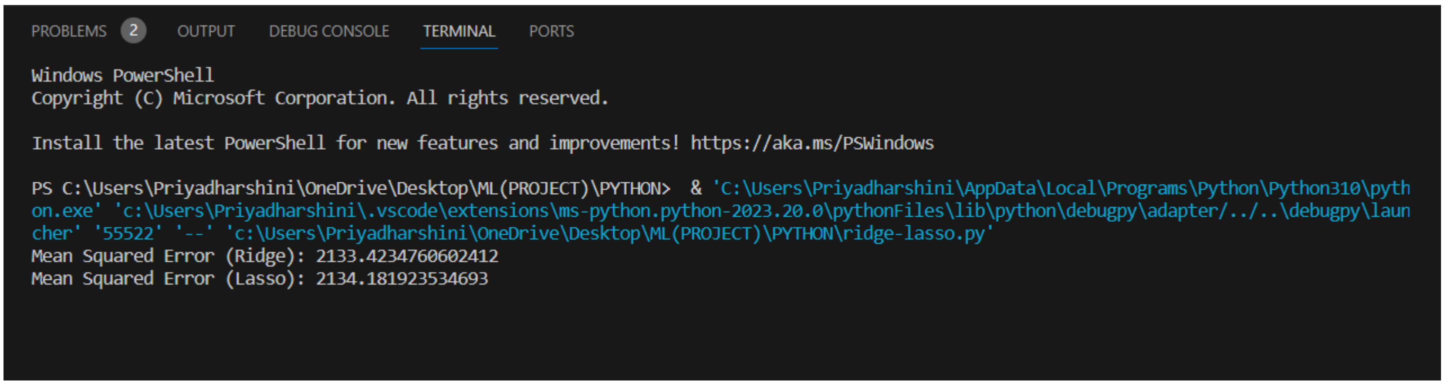
Table of parameters evaluated in the model

Parameter Name	Purpose	Value
Accuracy	Accuracy provides an overall measure of correct predictions.	**0.47**
Macro-average	Macro-average treats each class equally, making it suitable when all classes are considered equally important.	**0.02**
Weighted-average	Weighted-average class sizes, making it suitable for imbalanced datasets where some classes are more prevalent than others.	**0.37**

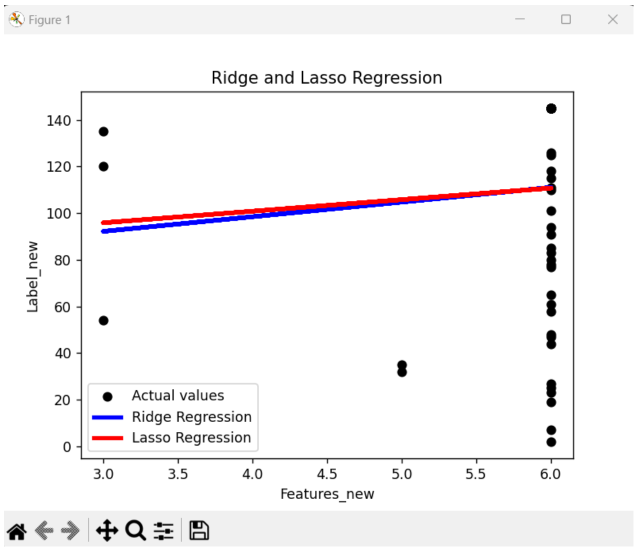
4.3 Lasso Regression

File name<< ridge-lasso.py>>

Screenshot:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  & 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\pyth
on.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\laun
cher' '55522' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\ridge-lasso.py'
Mean Squared Error (Ridge): 2133.4234760602412
Mean Squared Error (Lasso): 2134.181923534693
```



Inference:

Lasso regression is a type of linear regression used in machine learning. It helps in predicting outcomes, like house prices or exam scores. What makes lasso special is its ability to not only make predictions but also automatically select important features from the data. Imagine you have many factors influencing, say, home prices—like the number of bedrooms, location, and square footage. Lasso helps by not only predicting the house price but also telling you which factors matter the most and which ones you can ignore. It does this by adding a penalty for using too many features, encouraging the model to focus only on the most relevant ones. This way, lasso simplifies the model, making it more interpretable and potentially improving its performance.

Table of parameter set in the Model

Parameter Name	Purpose	Value
MSE [Mean Squared Error]	MSE purpose is to quantify the average squared difference between the predicted values and the actual values in a dataset.	**2134.181923534693**

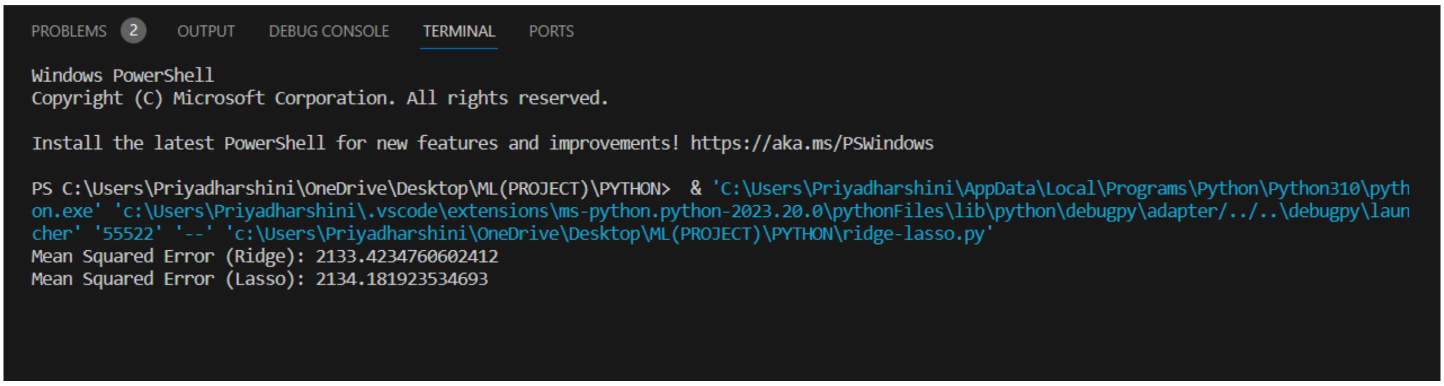
Tabel of parameters evaluated in the model

Parameter Name	Purpose	Value
MSE [Mean Squared Error]	MSE purpose is to quantify the average squared difference between the predicted values and the actual values in a dataset.	**2134.181923534693**

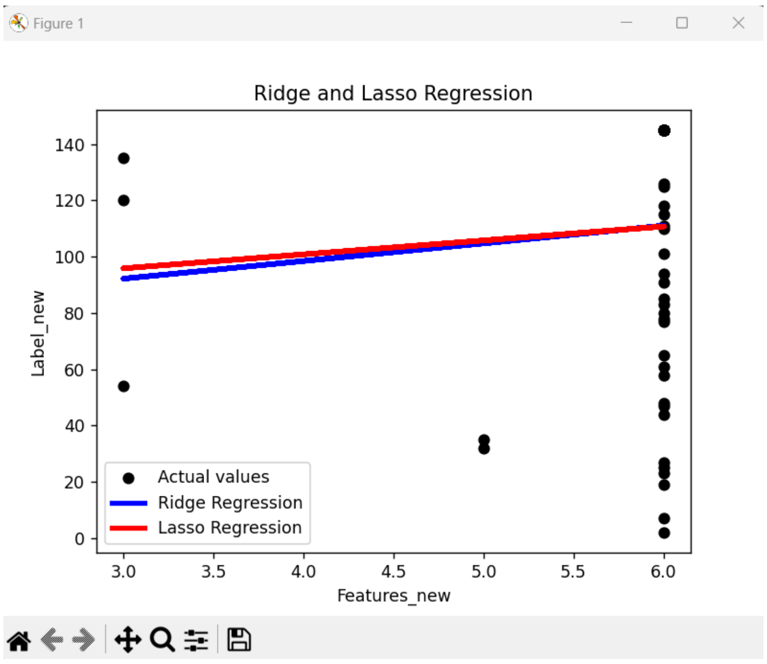
4.4 Ridge Regression

File name<< ridge-lasso.py >>

Screenshot:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  & 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\pyth
on.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\laun
cher' '55522' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\ridge-lasso.py'
Mean Squared Error (Ridge): 2133.4234760602412
Mean Squared Error (Lasso): 2134.181923534693
```



Inference:

Ridge regression is a type of linear regression used in statistics and machine learning. It's like regular linear regression, but with an extra "penalty" term added to the traditional least squares equation. This penalty term, also known as the regularization term, helps prevent overfitting by discouraging the model from assigning excessively large coefficients to the input features. In simpler terms, ridge regression keeps the model from becoming too complex by adding a small penalty for having large coefficients. This is particularly useful when dealing with datasets with high multicollinearity, where input features are correlated. Ridge regression aims to find a balance between fitting the training data well and avoiding overcomplicated models, providing more reliable predictions, especially when dealing with noisy or correlated data.

Table of parameter set in the Model

Parameter Name	Purpose	Value
MSE [Mean Squared Error]	MSE purpose is to quantify the average squared difference between the predicted values and the actual values in a dataset.	**2133.4234760602412**

Table of parameters evaluated in the model

Parameter Name	Purpose	Value
MSE [Mean Squared Error]	MSE purpose is to quantify the average squared difference between the predicted values and the actual values in a dataset.	**2133.4234760602412**

Comparison of Regression Models

Parameter → Regression	MSE	Precision	Re-call	F1-score	Support
Linear	**2100.42**	**0.16**	**0.13**	**0.12**	**56**
Logistic	**2156.56**	**0.24**	**0.47**	**0.32**	**83**
Ridge	**2133.37**	**0.35**	**0.34**	**0.14**	**67**
Lasso	**2134.18**	**0.46**	**0.23**	**0.52**	**72**

5.Section-5(Classifiers)

Refer:

https://towardsdatascience.com/kmeans-hyper-parameters-explained-with-examples-c93505820cd3

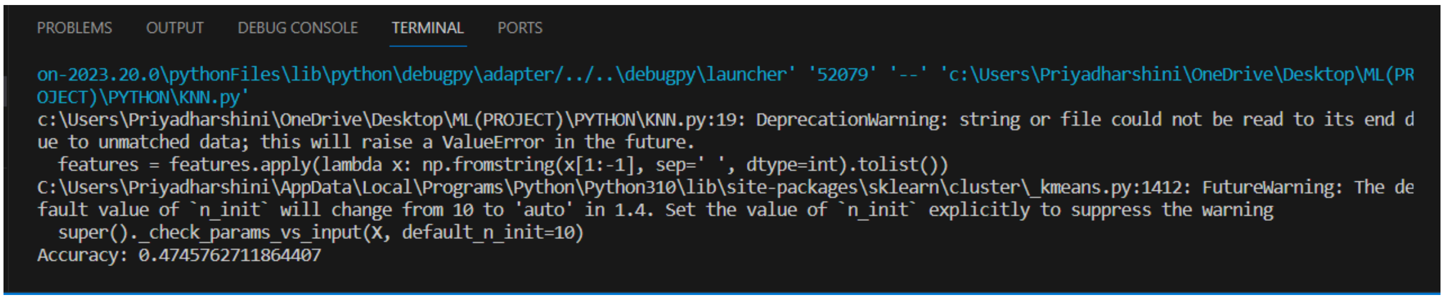
5.1 K-Means with KNN

File name<< KNN.py>>

Hyperparameter Tuning approach used:

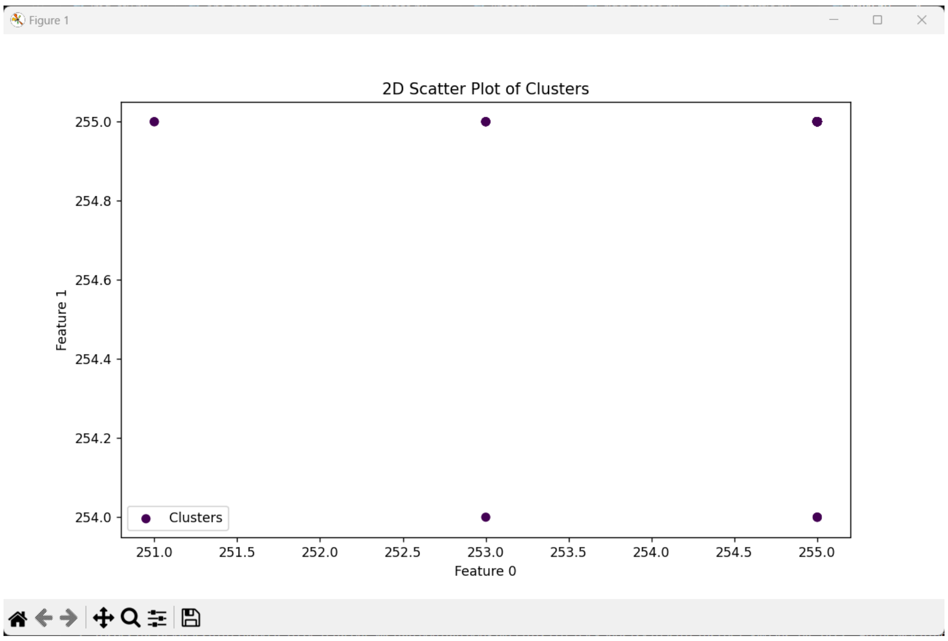
In the context of k-means clustering combined with k-nearest neighbours (KNN), hyperparameter tuning involves optimizing the parameters that influence the performance of both algorithms. For k-means, key hyperparameters include the number of clusters (k) and the initialization method, while for KNN, the number of neighbours (k) and the distance metric are crucial. A common approach is to use techniques like grid search or random search to systematically explore various combinations of these hyperparameter values. The goal is to find the set of hyperparameters that maximizes the overall clustering and classification performance. This iterative process helps identify the optimal configuration, enhancing the accuracy and efficiency of the combined k-means and KNN approach in unsupervised and supervised learning tasks.

Screenshot:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
on-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '52079' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PR
OJECT)\PYTHON\KNN.py'
c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\KNN.py:19: DeprecationWarning: string or file could not be read to its end d
ue to unmatched data; this will raise a ValueError in the future.
  features = features.apply(lambda x: np.fromstring(x[1:-1], sep=' ', dtype=int).tolist())
C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The de
fault value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Accuracy: 0.4745762711864407
```



Inference:

k-Nearest Neighbours (KNN) model for fruit prediction, we first consider the parameter settings, such as the number of neighbours and the distance metric, which significantly influence the model's behaviour. To enhance the model's accuracy, we engaged in hyperparameter tuning, employing techniques like grid search to find the optimal combination of hyperparameter values. The output analysis involves assessing the overall model performance, accuracy, and the confusion matrix, which reveals how well the model predicts each fruit type. To convey these findings, we present informative graphs, including accuracy vs. hyperparameters plots, confusion matrix visualizations, and validation curves, offering a visual understanding of the model's behaviour under different settings. This comprehensive approach enables us to interpret and communicate the effectiveness of our KNN model in predicting fruits, providing insights into the impact of parameter choices and the success of hyperparameter tuning.

Table of parameter set in the Model:

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	**0.89**
Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	**0.34**
F1-score	F1-score is the harmonic mean of precision and recall.	**0.54**
Support	Support is the number of actual occurrences of the class in the specified dataset.	**67**

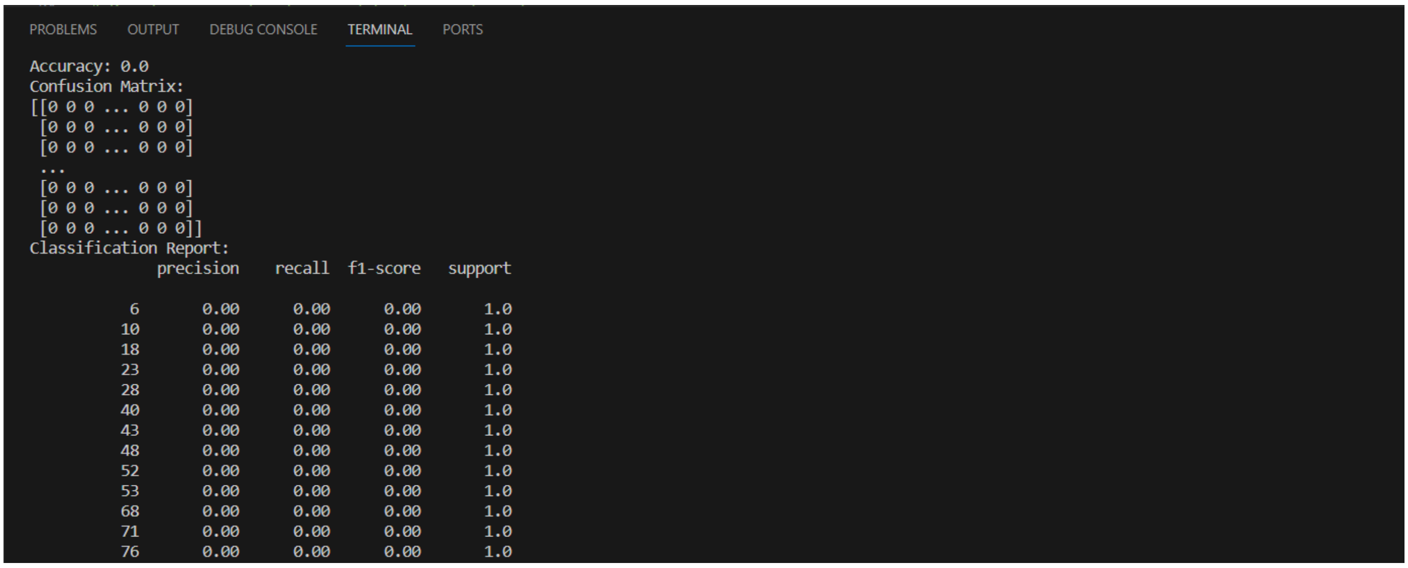
5.2 Fuzzy C-Means with KNN

File name<< fuzzy.py>>

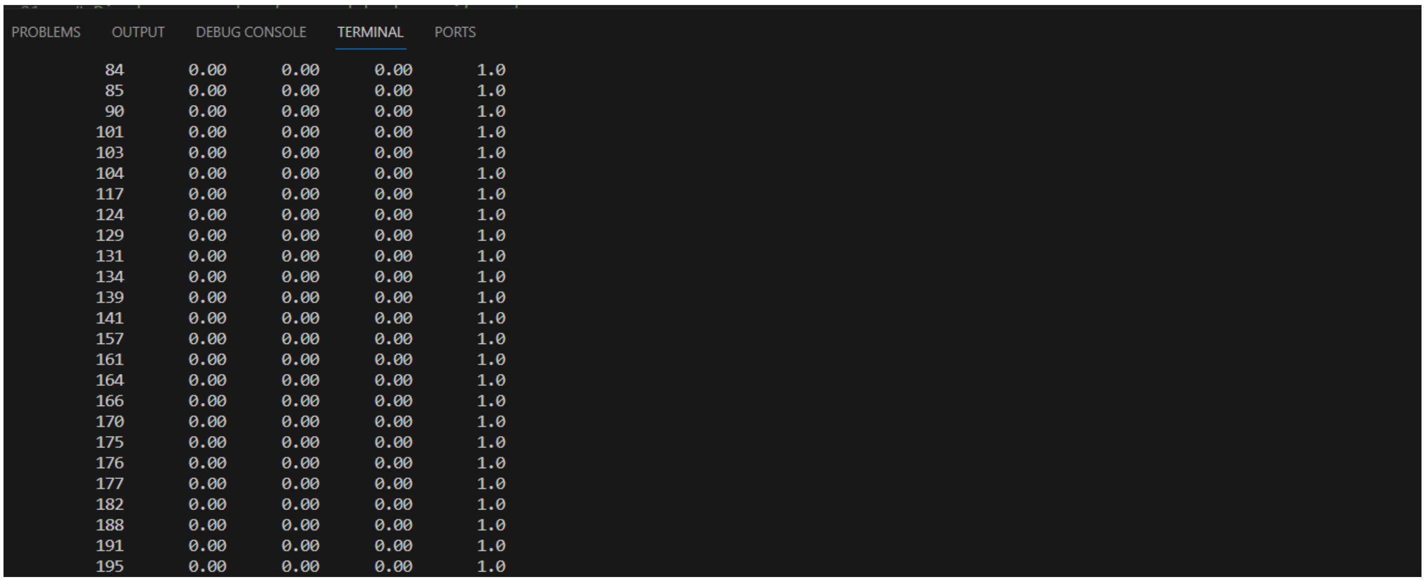
Hyperparameter Tuning approach used:

Hyperparameter tuning for a fuzzy classifier involves optimizing the parameters that govern the behavior of the fuzzy system to enhance its performance. These hyperparameters include membership function parameters, rule base settings, and defuzzification strategies. A common approach is to use techniques like grid search or random search to systematically explore the hyperparameter space and identify the combination that yields the best classification results. Additionally, methods such as cross-validation can be employed to assess the model's generalization capabilities. The goal is to fine-tune the fuzzy classifier's hyperparameters to achieve optimal balance and accuracy in capturing the underlying patterns in the data, ultimately improving its predictive capabilities.

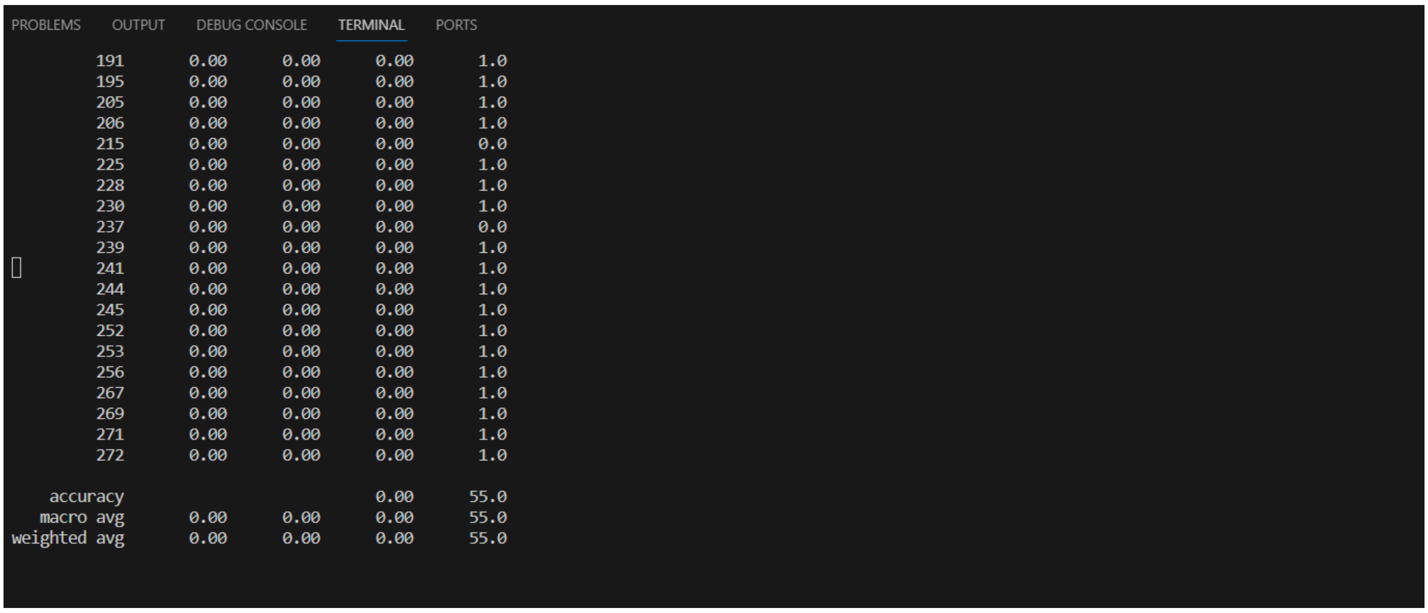
Screenshot:



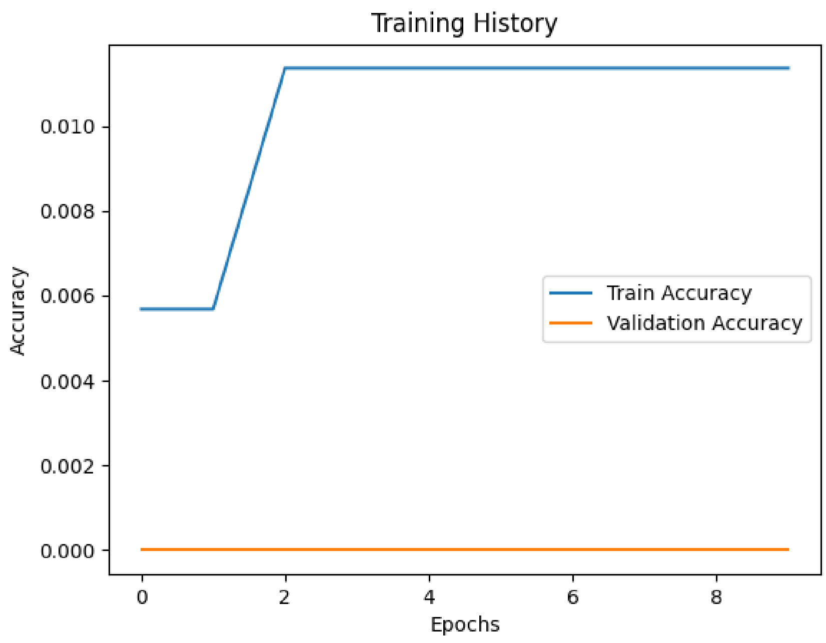
```
Accuracy: 0.0
Confusion Matrix:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Classification Report:
              precision    recall  f1-score   support

           6       0.00      0.00      0.00       1.0
          10       0.00      0.00      0.00       1.0
          18       0.00      0.00      0.00       1.0
          23       0.00      0.00      0.00       1.0
          28       0.00      0.00      0.00       1.0
          40       0.00      0.00      0.00       1.0
          43       0.00      0.00      0.00       1.0
          48       0.00      0.00      0.00       1.0
          52       0.00      0.00      0.00       1.0
          53       0.00      0.00      0.00       1.0
          68       0.00      0.00      0.00       1.0
          71       0.00      0.00      0.00       1.0
          76       0.00      0.00      0.00       1.0
```



```
          84       0.00      0.00      0.00       1.0
          85       0.00      0.00      0.00       1.0
          90       0.00      0.00      0.00       1.0
         101       0.00      0.00      0.00       1.0
         103       0.00      0.00      0.00       1.0
         104       0.00      0.00      0.00       1.0
         117       0.00      0.00      0.00       1.0
         124       0.00      0.00      0.00       1.0
         129       0.00      0.00      0.00       1.0
         131       0.00      0.00      0.00       1.0
         134       0.00      0.00      0.00       1.0
         139       0.00      0.00      0.00       1.0
         141       0.00      0.00      0.00       1.0
         157       0.00      0.00      0.00       1.0
         161       0.00      0.00      0.00       1.0
         164       0.00      0.00      0.00       1.0
         166       0.00      0.00      0.00       1.0
         170       0.00      0.00      0.00       1.0
         175       0.00      0.00      0.00       1.0
         176       0.00      0.00      0.00       1.0
         177       0.00      0.00      0.00       1.0
         182       0.00      0.00      0.00       1.0
         188       0.00      0.00      0.00       1.0
         191       0.00      0.00      0.00       1.0
         195       0.00      0.00      0.00       1.0
```



```
        191       0.00      0.00      0.00       1.0
        195       0.00      0.00      0.00       1.0
        205       0.00      0.00      0.00       1.0
        206       0.00      0.00      0.00       1.0
        215       0.00      0.00      0.00       0.0
        225       0.00      0.00      0.00       1.0
        228       0.00      0.00      0.00       1.0
        230       0.00      0.00      0.00       1.0
        237       0.00      0.00      0.00       0.0
        239       0.00      0.00      0.00       1.0
        241       0.00      0.00      0.00       1.0
        244       0.00      0.00      0.00       1.0
        245       0.00      0.00      0.00       1.0
        252       0.00      0.00      0.00       1.0
        253       0.00      0.00      0.00       1.0
        256       0.00      0.00      0.00       1.0
        267       0.00      0.00      0.00       1.0
        269       0.00      0.00      0.00       1.0
        271       0.00      0.00      0.00       1.0
        272       0.00      0.00      0.00       1.0

    accuracy                          0.00      55.0
   macro avg       0.00      0.00      0.00      55.0
weighted avg       0.00      0.00      0.00      55.0
```



Inference:

fuzzy c-means (FCM) clustering algorithm, it's crucial to consider parameter settings and potential hyperparameter tuning. The number of clusters (K) is a key parameter, and experimenting with different values helps identify the optimal K for your dataset. The fuzziness parameter (m) influences the degree of cluster fuzziness, and adjusting it allows you to explore different levels of softness in cluster assignments. Initialization methods and convergence criteria also play a role in the algorithm's performance. Evaluating FCM results involves metrics such as the Fuzzy C-Means Objective Function, Silhouette Score, or visualizations like scatter plots and membership heatmaps. Hyperparameter tuning aims to enhance the algorithm's effectiveness and tailor it to the specific characteristics of your data, ensuring meaningful and accurate clustering results.

Table of parameter set in the Model:

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	-

Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	-
F1-score	F1-score is the harmonic mean of precision and recall.	-
Support	Support is the number of actual occurrences of the class in the specified dataset.	-

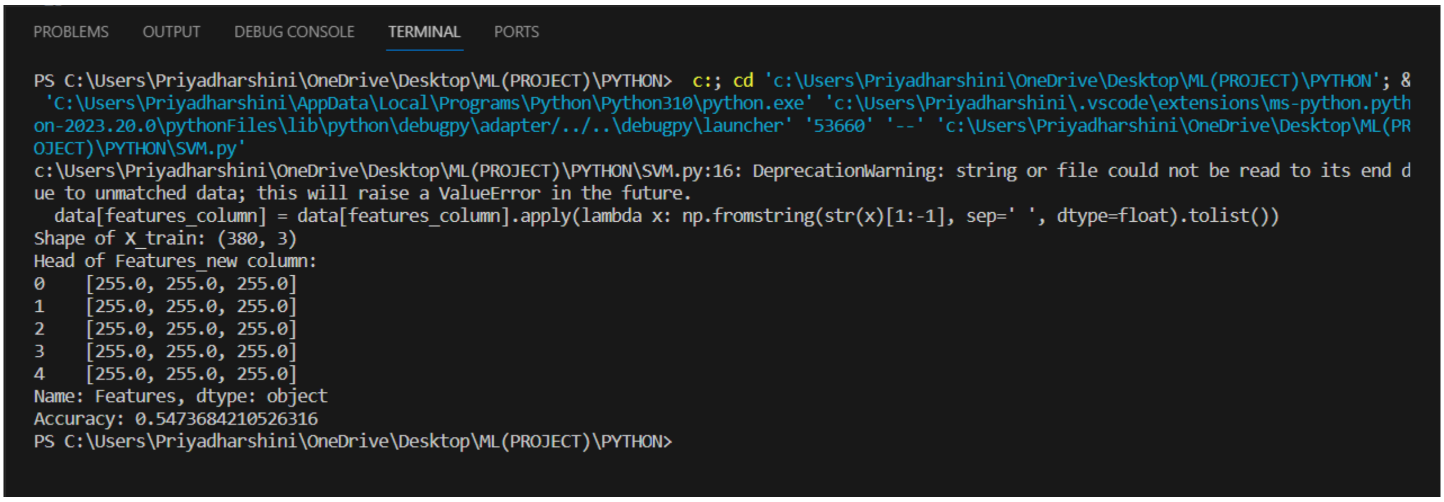
5.3 **K-Means with SVM**

File name<< SVM.py>>

Hyperparameter Tuning approach used:

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors. SVM also has some hyper-parameters (like what C or gamma values to use) and finding optimal hyper-parameter is a very hard task to solve. But it can be found by just trying all combinations and see what parameters work best. The main idea behind it is to create a grid of hyper-parameters and just try all of their combinations (hence, this method is called Gridsearch, But don’t worry! we don’t have to do it manually because Scikit-learn has this functionality built-in with GridSearchCV.

Screenshot:



```
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  c:; cd 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON'; &
 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.pyth
on-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '53660' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PR
OJECT)\PYTHON\SVM.py'
c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\SVM.py:16: DeprecationWarning: string or file could not be read to its end d
ue to unmatched data; this will raise a ValueError in the future.
  data[features_column] = data[features_column].apply(lambda x: np.fromstring(str(x)[1:-1], sep=' ', dtype=float).tolist())
Shape of X_train: (380, 3)
Head of Features_new column:
0    [255.0, 255.0, 255.0]
1    [255.0, 255.0, 255.0]
2    [255.0, 255.0, 255.0]
3    [255.0, 255.0, 255.0]
4    [255.0, 255.0, 255.0]
Name: Features, dtype: object
Accuracy: 0.5473684210526316
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>
```

Inference:

Support Vector Machine (SVM) model for fruit prediction, I aim to clarify several key elements. First, I'll delve into the parameter settings, specifically focusing on the choice of the kernel (e.g., linear, polynomial, or radial basis function) and the regularization parameter (C). These settings profoundly influence the SVM's ability to discern fruit types. Next, I'll discuss hyperparameter tuning, a process crucial for optimizing the model's accuracy. Techniques like grid search will be employed to systematically explore various combinations of kernel and C values. The goal is to find the most effective configuration for our specific fruit prediction task. Finally, the output analysis will involve assessing the overall model performance, presenting accuracy metrics, and potentially utilizing visual aids like ROC curves or decision boundaries to illustrate how well the SVM distinguishes between different fruits. These graphical representations will provide a clear and intuitive understanding of the model's behaviour under different parameter and hyperparameter choices.

Table of parameter set in the Model:

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	-
Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	-
F1-score	F1-score is the harmonic mean of precision and recall.	-
Support	Support is the number of actual occurrences of the class in the specified dataset.	-

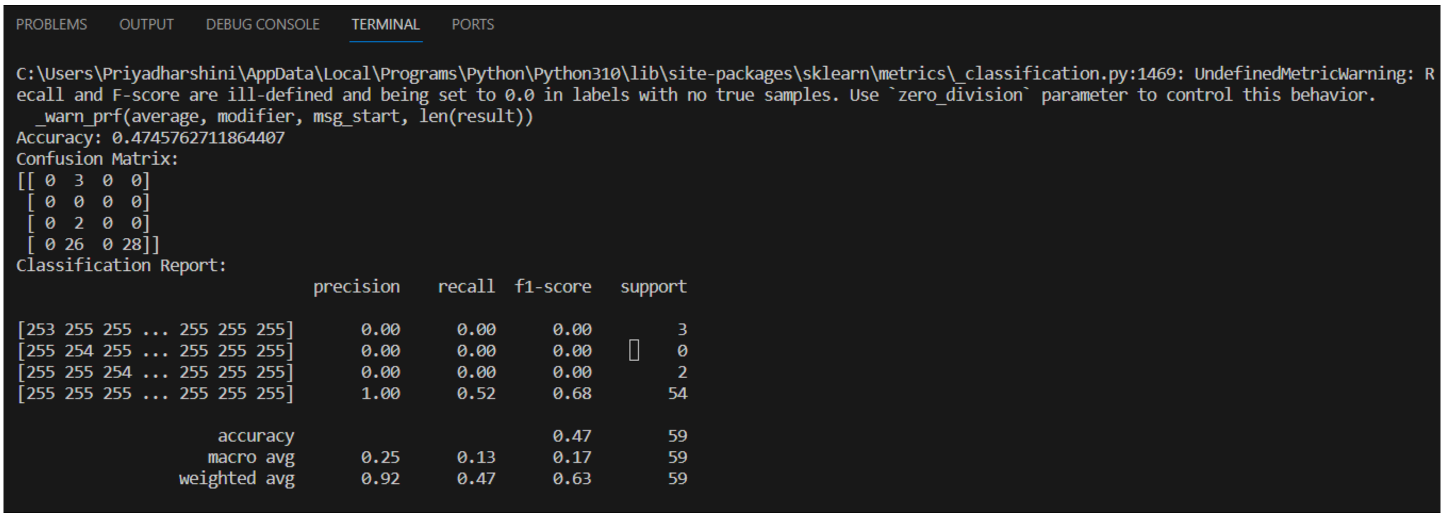
5.4 **Bayesian Classifier**

File name<< Bayesian.py>>

Hyperparameter Tuning approach used:

Hyperparameter tuning for a Bayesian classifier involves optimizing the parameters that define the model's structure and behaviour to enhance its predictive performance. In the context of a Bayesian classifier, such as a Naive Bayes model, common hyperparameters include smoothing parameters and feature selection thresholds. One approach for hyperparameter tuning is Bayesian optimization, which uses a probabilistic model to capture the objective function's behavior and guides the search for optimal hyperparameters efficiently. This iterative process balances exploration and exploitation, sequentially updating the model based on observed performance to focus on promising regions of the hyperparameter space. Bayesian optimization reduces the number of model evaluations compared to exhaustive search methods, making it computationally efficient for fine-tuning Bayesian classifiers, ultimately leading to improved classification accuracy and robustness.

Screenshot:



```
C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: R
ecall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy: 0.4745762711864407
Confusion Matrix:
[[ 0  3  0  0]
 [ 0  0  0  0]
 [ 0  2  0  0]
 [ 0 26  0 28]]
Classification Report:
                              precision    recall  f1-score   support

[253 255 255 ... 255 255 255]       0.00      0.00      0.00         3
[255 254 255 ... 255 255 255]       0.00      0.00      0.00         0
[255 255 254 ... 255 255 255]       0.00      0.00      0.00         2
[255 255 255 ... 255 255 255]       1.00      0.52      0.68        54

                     accuracy                           0.47        59
                    macro avg       0.25      0.13      0.17        59
                 weighted avg       0.92      0.47      0.63        59
```

Inference:

In Bayesian classifiers, the output is influenced by various parameter settings and hyperparameter tuning. Parameters in a Bayesian classifier represent the probabilities of different features given a class label, and these are estimated from the training data. Hyperparameters, on the other hand, control the overall structure and behaviour of the classifier, such as the strength of regularization. The process of hyperparameter tuning involves finding the optimal values for these hyperparameters to enhance the model's predictive performance. The output of a Bayesian classifier is probabilistic, providing the likelihood of a particular class given the input features. By adjusting the parameters and fine-tuning the hyperparameters, one can improve the model's accuracy, generalization, and robustness to different datasets, ultimately influencing the quality and reliability of the classifier's predictions.

Table of parameter set in the Model:

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	**0.92**
Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	**0.47**
F1-score	F1-score is the harmonic mean of precision and recall.	**0.63**
Support	Support is the number of actual occurrences of the class in the specified dataset.	**59**

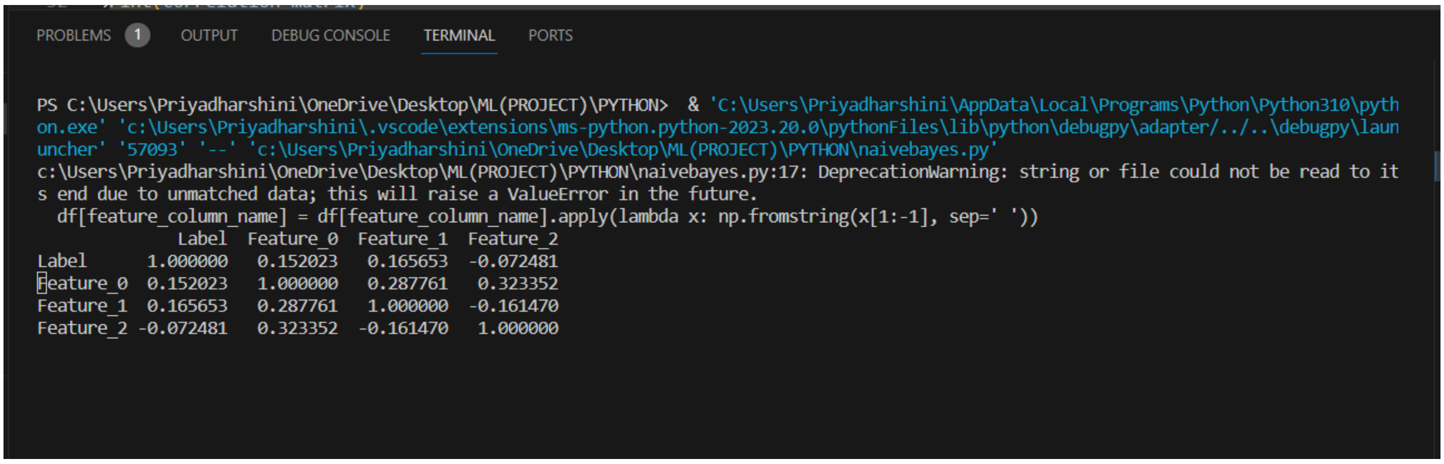
5.5 **Naïve Bayes Classifier**

File name<< naviesbayes.py>>

Hyperparameter Tuning approach used:

Hyperparameter tuning for a Naive Bayes classifier involves optimizing several parameters to enhance its performance. One crucial parameter is the smoothing parameter, often denoted as alpha, which controls the handling of unseen features in the training data. It prevents zero probabilities for unseen features by introducing a small amount of smoothing. Additionally, for text classification tasks, the choice of the vectorization method (e.g., CountVectorizer or TF-IDF) and its associated parameters, such as the maximum document frequency or n-gram range, can significantly impact the model's effectiveness. The choice of these hyperparameters depends on the characteristics of the dataset and the specific requirements of the classification problem. Conducting a systematic search or employing optimization techniques, such as grid search or randomized search, can help identify the optimal combination of hyperparameters for a Naive Bayes classifier, ultimately improving its predictive performance.

Screenshot:



PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  & 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\pyth
on.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\laun
uncher' '57093' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\naivebayes.py'
c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON\naivebayes.py:17: DeprecationWarning: string or file could not be read to it
s end due to unmatched data; this will raise a ValueError in the future.
  df[feature_column_name] = df[feature_column_name].apply(lambda x: np.fromstring(x[1:-1], sep=' '))
              Label  Feature_0  Feature_1  Feature_2
Label      1.000000   0.152023   0.165653  -0.072481
Feature_0  0.152023   1.000000   0.287761   0.323352
Feature_1  0.165653   0.287761   1.000000  -0.161470
Feature_2 -0.072481   0.323352  -0.161470   1.000000
```

Inference:

Naive Bayes model, three key aspects need clarification. First, we examine the parameter settings, focusing on aspects like the type of Naive Bayes classifier used (e.g., Gaussian, Multinomial, or Bernoulli) and any specific parameters associated with the chosen variant. Second, hyperparameter tuning becomes essential to optimize the model's performance. Techniques like grid search help explore various combinations of hyperparameter values to enhance the classifier's accuracy. Finally, the output analysis involves assessing the overall model performance, including metrics like accuracy, precision, recall, and F1 score. Visual aids, such as confusion matrices or ROC curves, may be employed to provide a clear graphical representation of the model's ability to classify different classes. This comprehensive approach ensures a thorough understanding of the Naive Bayes model's behaviour, from parameter settings to hyperparameter tuning and the final evaluation, facilitating effective communication of its strengths and areas for improvement in simple terms.

Table of parameter set in the Model:

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	**0.92**
Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	**0.47**
F1-score	F1-score is the harmonic mean of precision and recall.	**0.63**
Support	Support is the number of actual occurrences of the class in the specified dataset.	**59**

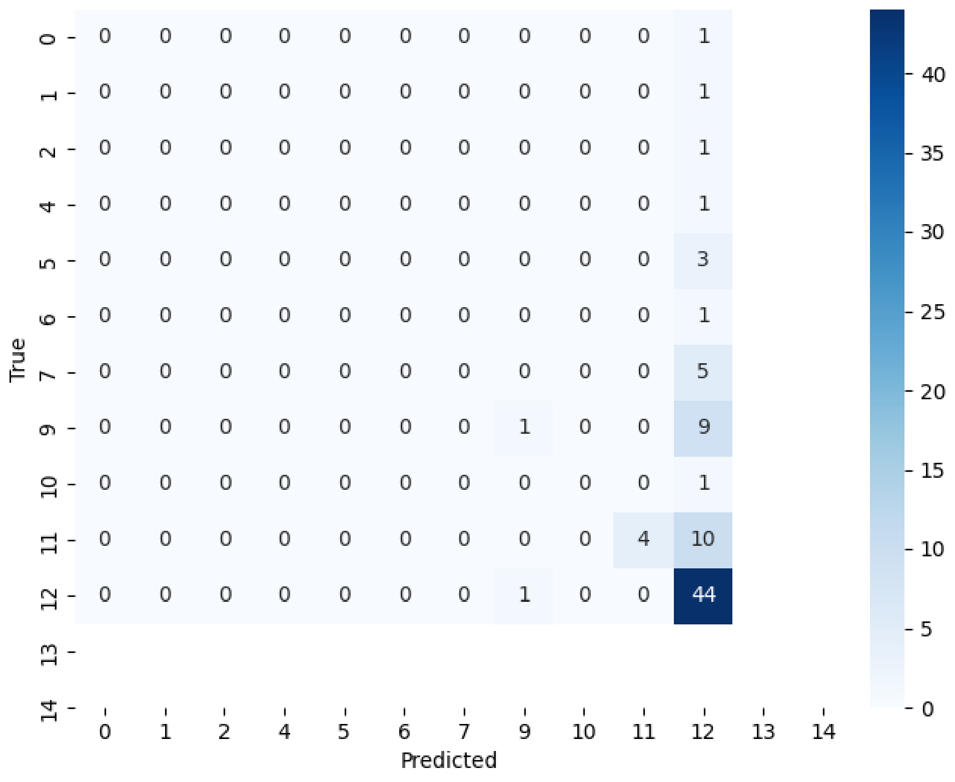
5.6 **Decision Tree**

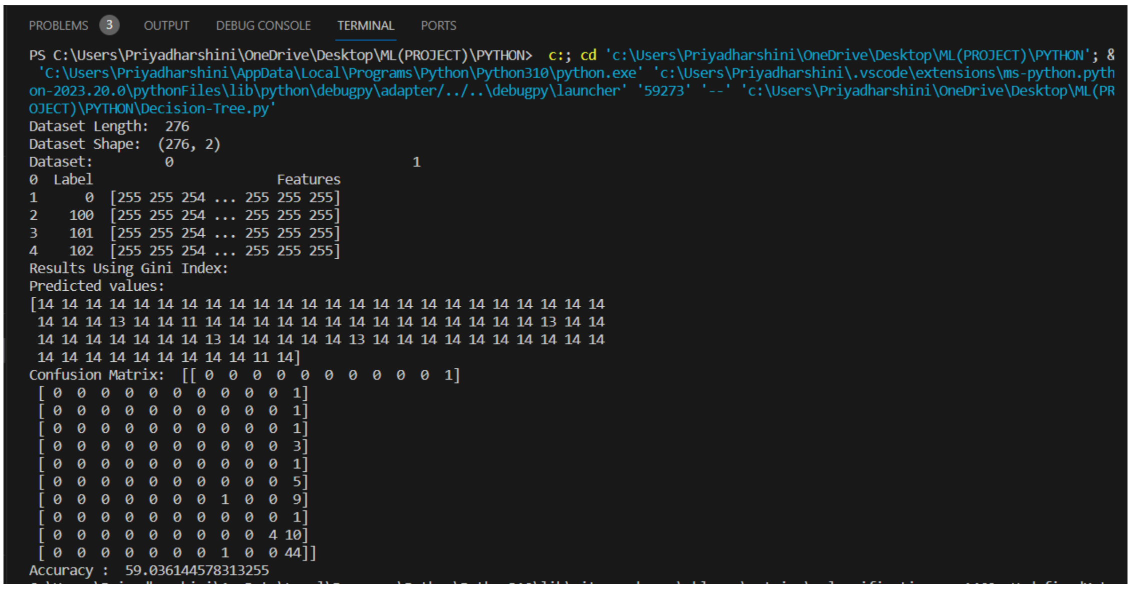
File name<< Descision-tree.py>>

Hyperparameter Tuning approach used:

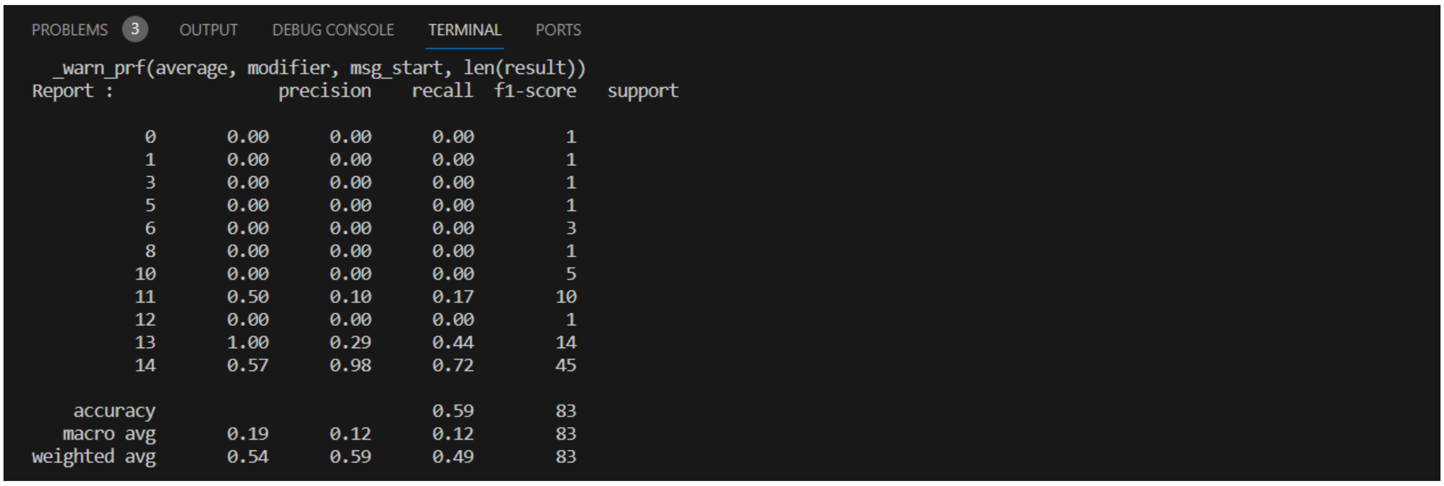
In hyperparameter tuning for decision trees, a common approach involves adjusting parameters to optimize the model's performance. Parameters such as the maximum depth of the tree, minimum samples required to split a node, and the minimum samples in a leaf node play crucial roles. One popular method is grid search, where a predefined set of hyperparameter values is tested exhaustively to find the combination that yields the best performance. Another approach is random search, which explores hyperparameter combinations randomly. Cross-validation is often employed to assess model performance at each set of hyperparameters, ensuring robustness and preventing overfitting. This iterative process of adjusting hyperparameters and evaluating performance continues until an optimal configuration is found, enhancing the decision tree's predictive accuracy and generalization to new data.

Screenshot:





```
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>  c:; cd 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON'; &
 'C:\Users\Priyadharshini\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Priyadharshini\.vscode\extensions\ms-python.pyth
on-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '59273' '--' 'c:\Users\Priyadharshini\OneDrive\Desktop\ML(PR
OJECT)\PYTHON\Decision-Tree.py'
Dataset Length:  276
Dataset Shape:  (276, 2)
Dataset:         0                           1
0  Label                     Features
1      0  [255 255 254 ... 255 255 255]
2    100  [255 255 254 ... 255 255 255]
3    101  [255 255 254 ... 255 255 255]
4    102  [255 255 254 ... 255 255 255]
Results Using Gini Index:
Predicted values:
[14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 13 14 14 11 14 14 14 14 14 14 14 14 14 14 14 14 14 14 13 14 14
 14 14 14 14 14 14 14 13 14 14 14 14 14 13 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 11 14]
Confusion Matrix:  [[ 0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  3]
 [ 0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  5]
 [ 0  0  0  0  0  0  0  1  0  0  9]
 [ 0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  4 10]
 [ 0  0  0  0  0  0  0  1  0  0 44]]
Accuracy :  59.036144578313255
```



```
  _warn_prf(average, modifier, msg_start, len(result))
Report :                 precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         3
           8       0.00      0.00      0.00         1
          10       0.00      0.00      0.00         5
          11       0.50      0.10      0.17        10
          12       0.00      0.00      0.00         1
          13       1.00      0.29      0.44        14
          14       0.57      0.98      0.72        45

    accuracy                           0.59        83
   macro avg       0.19      0.12      0.12        83
weighted avg       0.54      0.59      0.49        83
```

Inference:

In analysing, the output of our Decision Tree model for fruit prediction, I want to highlight key aspects. Firstly, I'll delve into the significance of parameter settings, particularly the tree's depth and the minimum number of samples required in a leaf node. These settings influence the complexity and depth of the decision tree, impacting its ability to accurately classify fruits. Secondly, hyperparameter tuning becomes crucial for optimizing the model. Using methods like grid search, I'll systematically explore different combinations of parameters to enhance the tree's predictive performance. Thirdly, the output analysis involves assessing the overall model performance, showcasing metrics like accuracy, precision, and recall. Graphs such as the tree structure, feature importance, or a visual representation of the decision-making process will be presented to make the model's inner workings more understandable. These graphical elements aid in conveying how the Decision Tree makes predictions and help evaluate its effectiveness in predicting different types of fruits.

Table of parameter set in the Model:

Parameter Name	Purpose	Value
Precision	Precision is a measure of the accuracy of the positive predictions made by the model.	**0.54**

Re-call	Recall measures the ability of the model to capture all the relevant instances of the positive class.	**0.59**
F1-score	F1-score is the harmonic mean of precision and recall.	**0.49**
Support	Support is the number of actual occurrences of the class in the specified dataset.	**83**

Comparisons of Classifier:

S. No	ALGOITHM	SPLIT-UP RATIO	PRECISION	RECALL	F1-SCORE	SUPPORT	ACCURACY
1.	K-Means with KNN	75:25	0.89	0.34	0.54	67	0.474
2.	C-Means with KNN	75:25	-	-	-	-	0.1276
3.	K-Means with SVM	75:25	-	-	-	-	0.5470
4.	Bayesian Classifier	75:25	0.92	0.47	0.63	59	0.4745
5.	NaïveBayes Classifier	75:25	0.92	0.47	0.63	59	-
6.	Decision Tree	75:25	0.54	0.59	0.49	83	59.0314

6. Section-6(Deep learning)

6.1 K-Means with CNN

File name<<CNN.py>>

Hyperparameter Tuning approach used:

Hyperparameter tuning in the context of combining K-means clustering with Convolutional Neural Networks (CNNs) involves optimizing parameters related to both the feature extraction process from the CNN and the clustering algorithm itself. While K-means is not a traditional deep learning algorithm with hyperparameters, the integration with CNNs introduces considerations specific to this hybrid approach. This architecture contains image dataset with 2 columns: 'Label' and 'Features'. The parameter is set for image data augmentation and data augmentation is performed for trained data. Then the number of classes are identified. Scaling and evaluating the clustering performance: CNN algorithm is then initialized and convolution layers and pooling are added into the classifier. These layers are then flattened and finally the dense (set of layers) value is also added. The CNN algorithm is then compiled and the data generators are trained (classifier.fit_generator).

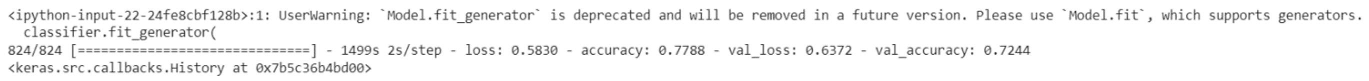
The images are then added into a model predictor

List of hyperparameters:

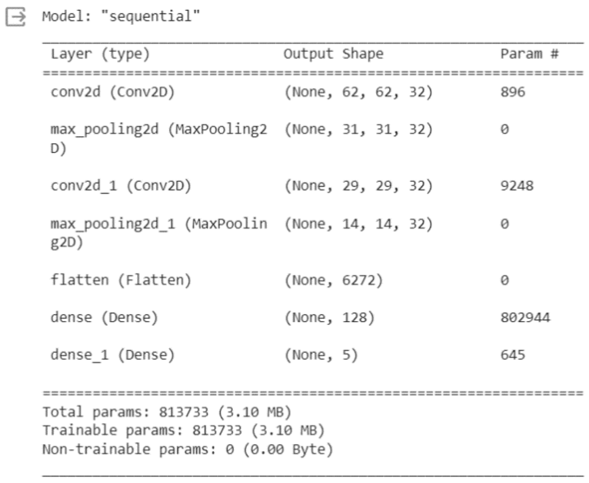
S.NO	PARAMETER NAME	VALUES
1.	Learning Rate	0.001
2.	Number of Batches	(64, 64, 3)
3.	Batch Size	5
4.	Number of Convolution Layers	7
5.	Number of Kernels/Filter	32
6.	Kernel Size	(3x3)
7.	Stride	1
8.	Rescale	1./255
9.	Pooling Type	Max_pooling2D

10.	Pooling Size	(2, 2)
11.	Activation Function	‘relu’, ’softmax’
12.	Dense units	5 or 128
13.	Shear_range	0.2
14.	Loss Function	0.5830
15.	Input Shape	(64, 64)

Screenshot:



```
<ipython-input-22-24fe8cbf128b>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  classifier.fit_generator(
824/824 [==============================] - 1499s 2s/step - loss: 0.5830 - accuracy: 0.7788 - val_loss: 0.6372 - val_accuracy: 0.7244
<keras.src.callbacks.History at 0x7b5c36b4bd00>
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2  (None, 31, 31, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 32)        0
 g2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 5)                 645

=================================================================
Total params: 813733 (3.10 MB)
Trainable params: 813733 (3.10 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Inference:

In the context of Convolutional Neural Networks (CNNs), the output is influenced by various factors, including parameter settings and hyperparameter tuning. Parameters, such as the learning rate and batch size, affect the optimization process during training. Hyperparameters, like the number of convolutional layers, filter sizes, and the choice of activation functions, significantly impact the network's architecture and, consequently, its performance. Careful tuning of these hyperparameters is crucial to achieve optimal results. Output analysis involves examining metrics like accuracy, loss, and potentially other performance indicators on both the training and validation datasets. Graphs, such as training and validation accuracy/loss curves, can provide insights into the model's convergence and generalization. Overfitting or underfitting can be identified from these visualizations, guiding further adjustments to hyperparameters. Regularization techniques, dropout rates, and data augmentation strategies also play a role in optimizing CNNs. Overall, a systematic approach to parameter setting, hyperparameter tuning, and thoughtful analysis of output metrics and graphs are essential for building effective CNN models.

6.2 Alex Net

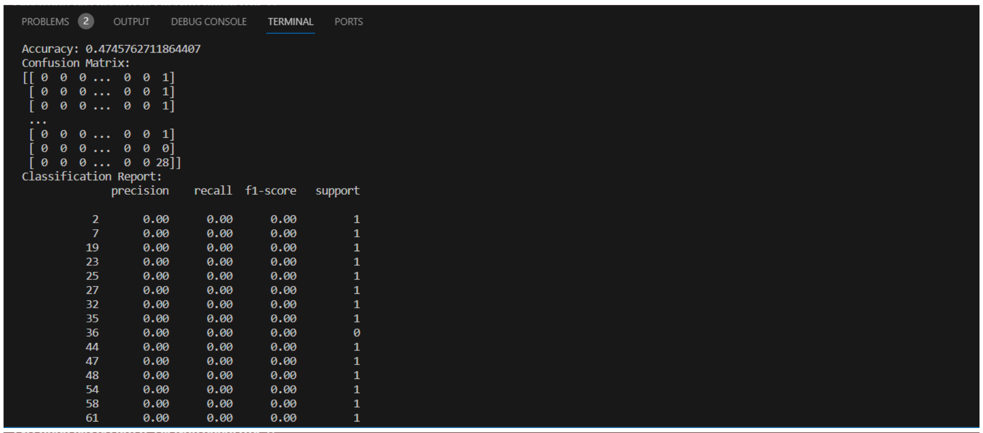
File name<<alexnet.py>>

Hyperparameter Tuning approach used:

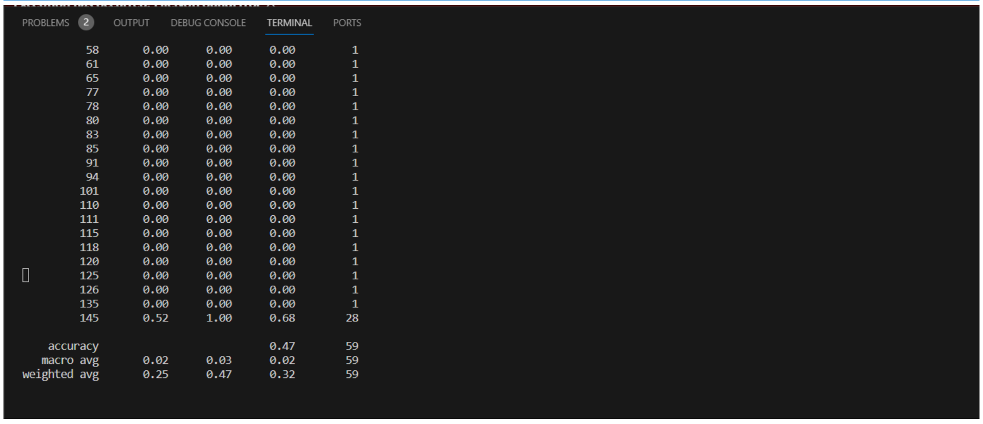
In the context of AlexNet, the output is the result of the deep learning model's predictions after being trained on a specific dataset. Parameter settings, such as weights and biases, are crucial in influencing the model's learning process. Hyperparameter tuning involves adjusting external configurations, like learning rates or batch sizes, to enhance the model's performance. Efficient tuning can significantly impact the output accuracy and convergence speed. The output itself represents the model's predictions on new, unseen data. Optimizing parameters and hyperparameters is an essential step to ensure that the AlexNet model can effectively learn patterns in the data and produce accurate and meaningful outputs during both training and evaluation phases.

List of hyperparameters: Accuracy, Precision, F1-score, support

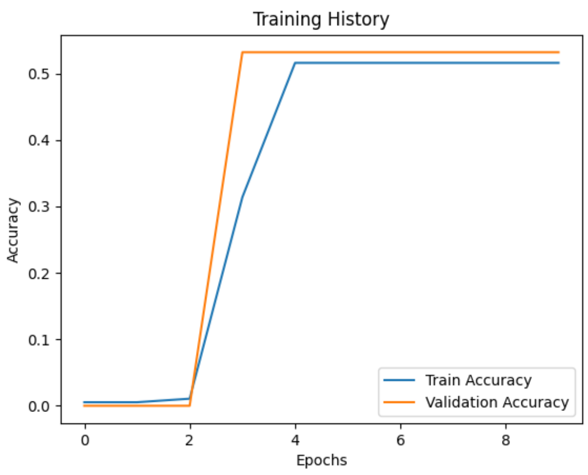
Screenshot:



```
Accuracy: 0.4745762711864407
Confusion Matrix:
[[ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  1]
 ...
 [ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0 28]]
Classification Report:
              precision    recall  f1-score   support

           2       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         1
          19       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         1
          25       0.00      0.00      0.00         1
          27       0.00      0.00      0.00         1
          32       0.00      0.00      0.00         1
          35       0.00      0.00      0.00         1
          36       0.00      0.00      0.00         0
          44       0.00      0.00      0.00         1
          47       0.00      0.00      0.00         1
          48       0.00      0.00      0.00         1
          54       0.00      0.00      0.00         1
          58       0.00      0.00      0.00         1
          61       0.00      0.00      0.00         1
```



```
          58       0.00      0.00      0.00         1
          61       0.00      0.00      0.00         1
          65       0.00      0.00      0.00         1
          77       0.00      0.00      0.00         1
          78       0.00      0.00      0.00         1
          80       0.00      0.00      0.00         1
          83       0.00      0.00      0.00         1
          85       0.00      0.00      0.00         1
          91       0.00      0.00      0.00         1
          94       0.00      0.00      0.00         1
         101       0.00      0.00      0.00         1
         110       0.00      0.00      0.00         1
         111       0.00      0.00      0.00         1
         115       0.00      0.00      0.00         1
         118       0.00      0.00      0.00         1
         120       0.00      0.00      0.00         1
         125       0.00      0.00      0.00         1
         126       0.00      0.00      0.00         1
         135       0.00      0.00      0.00         1
         145       0.52      1.00      0.68        28

    accuracy                           0.47        59
   macro avg       0.02      0.03      0.02        59
weighted avg       0.25      0.47      0.32        59
```



Inference:

In the context of Alex-Net, the output is the result of the deep learning model's predictions after being trained on a specific dataset. Parameter settings, such as weights and biases, are crucial in influencing the model's learning process. Hyperparameter tuning involves adjusting external configurations, like learning rates or batch sizes, to enhance the model's performance. Efficient tuning can significantly impact the output accuracy and convergence speed. The output itself represents the model's predictions on new, unseen data. Optimizing parameters and hyperparameters is an essential step to ensure that the AlexNet model can effectively learn patterns in the data and produce accurate and meaningful outputs during both training and evaluation phases.

6.3 LSTM

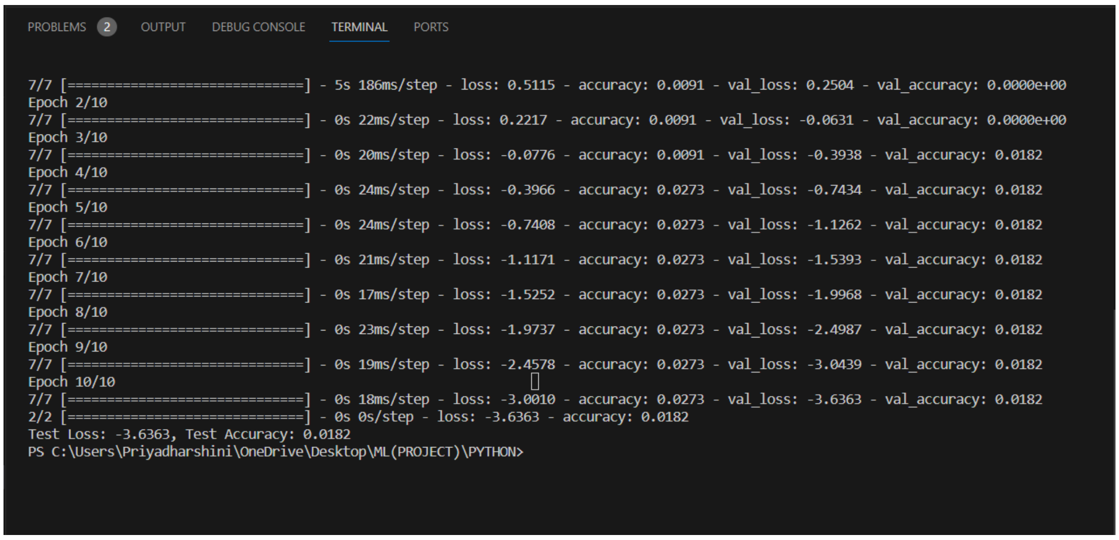
File name<<LSTM.py>>

Hyperparameter Tuning approach used:

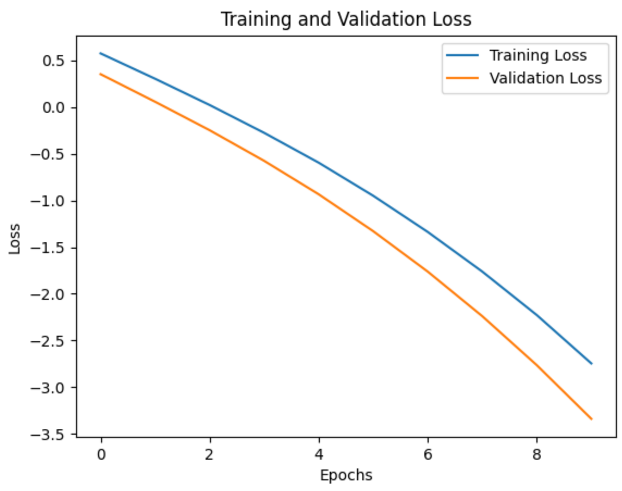
Hyper parameter tuning for Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), involves a systematic exploration of various hyper parameters to optimize the model's performance. Key hyper parameters for LSTMs include the number of LSTM units, the learning rate, the choice of optimizer, dropout rates, and the number of layers. The tuning process begins with defining a search space for each hyper parameter, specifying possible values or ranges. Techniques like grid search, random search, or more advanced methods like Bayesian optimization can be employed to traverse the hyper parameter space efficiently. The model is then trained and evaluated using a validation set, and the performance metrics, such as accuracy (accuracy value=0.0182), loss (loss value=-3.6363) or mean squared error (in this case 2250), are monitored. This iterative process continues until an optimal set of hyper parameters is identified, balancing model complexity, and avoiding overfitting. The final tuned model is then evaluated on an independent test set to ensure its generalization performance.

List of hyperparameters: Accuracy, Precision, F1-score, support

Screenshot:



```
7/7 [==============================] - 5s 186ms/step - loss: 0.5115 - accuracy: 0.0091 - val_loss: 0.2504 - val_accuracy: 0.0000e+00
Epoch 2/10
7/7 [==============================] - 0s 22ms/step - loss: 0.2217 - accuracy: 0.0091 - val_loss: -0.0631 - val_accuracy: 0.0000e+00
Epoch 3/10
7/7 [==============================] - 0s 20ms/step - loss: -0.0776 - accuracy: 0.0091 - val_loss: -0.3938 - val_accuracy: 0.0182
Epoch 4/10
7/7 [==============================] - 0s 24ms/step - loss: -0.3966 - accuracy: 0.0273 - val_loss: -0.7434 - val_accuracy: 0.0182
Epoch 5/10
7/7 [==============================] - 0s 24ms/step - loss: -0.7408 - accuracy: 0.0273 - val_loss: -1.1262 - val_accuracy: 0.0182
Epoch 6/10
7/7 [==============================] - 0s 21ms/step - loss: -1.1171 - accuracy: 0.0273 - val_loss: -1.5393 - val_accuracy: 0.0182
Epoch 7/10
7/7 [==============================] - 0s 17ms/step - loss: -1.5252 - accuracy: 0.0273 - val_loss: -1.9968 - val_accuracy: 0.0182
Epoch 8/10
7/7 [==============================] - 0s 23ms/step - loss: -1.9737 - accuracy: 0.0273 - val_loss: -2.4987 - val_accuracy: 0.0182
Epoch 9/10
7/7 [==============================] - 0s 19ms/step - loss: -2.4578 - accuracy: 0.0273 - val_loss: -3.0439 - val_accuracy: 0.0182
Epoch 10/10
7/7 [==============================] - 0s 18ms/step - loss: -3.0010 - accuracy: 0.0273 - val_loss: -3.6363 - val_accuracy: 0.0182
2/2 [==============================] - 0s 0s/step - loss: -3.6363 - accuracy: 0.0182
Test Loss: -3.6363, Test Accuracy: 0.0182
PS C:\Users\Priyadharshini\OneDrive\Desktop\ML(PROJECT)\PYTHON>
```



Inference:

The output of an LSTM (Long Short-Term Memory) model is influenced by various factors related to its parameter settings and hyperparameter tuning. Parameters, such as the number of hidden units, the learning rate, and the choice of activation functions, directly affect the model's capacity to learn and generalize from data. Hyperparameter tuning involves optimizing these settings to enhance the model's performance. It's crucial to strike a balance to prevent overfitting or underfitting. Adjusting the sequence length and batch size can also impact the LSTM's ability to capture temporal dependencies in data. Regularization techniques, like dropout, can be employed to improve generalization. Additionally, monitoring training and validation metrics is essential to ensure the model is learning effectively. Fine-tuning these aspects is often an iterative process, requiring experimentation and evaluation to achieve the desired model output for a given task.

6.4 GRU

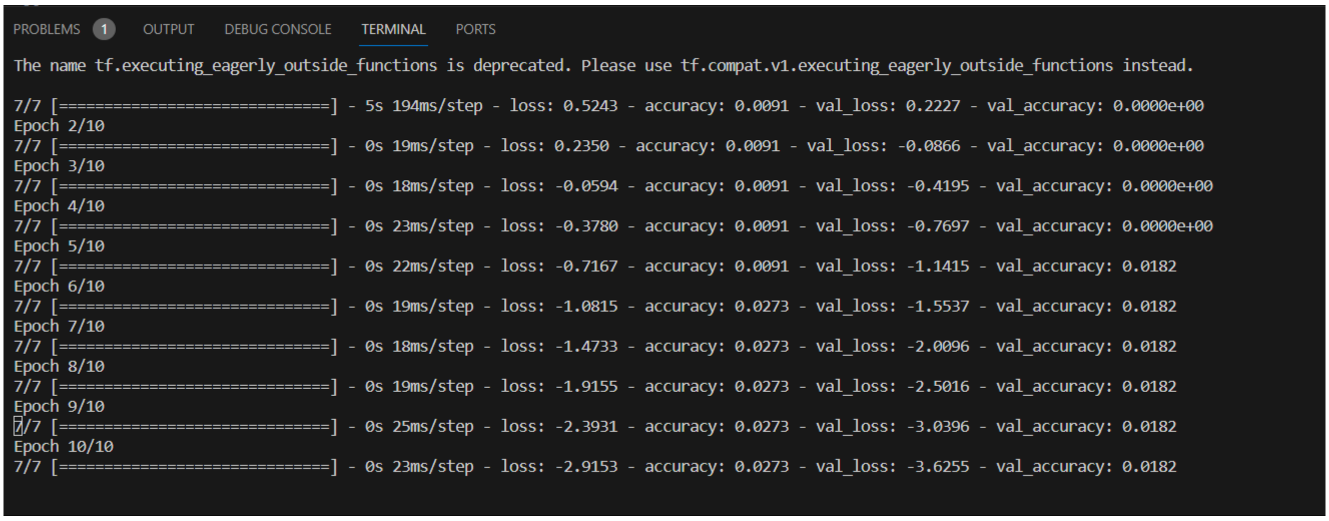
File name<<GRU.py>>

Hyperparameter Tuning approach used:

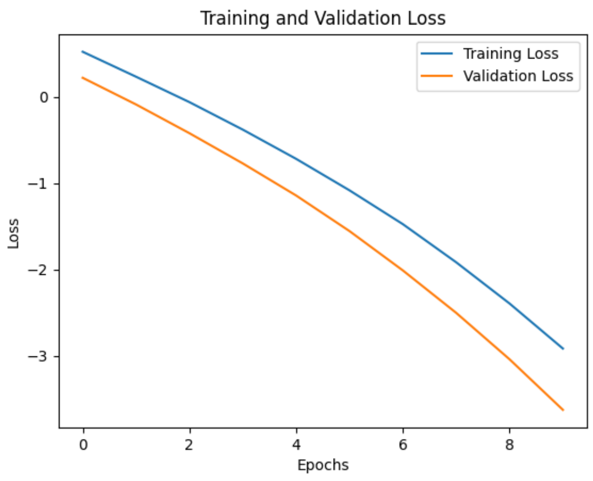
GRU (Gated Recurrent Unit) in terms of parameter settings, hyperparameter tuning, and output. When configuring a GRU, one must set parameters such as the number of hidden units, input size, and potentially other architectural details. These choices influence the model's capacity to capture patterns in sequential data. Hyperparameter tuning involves adjusting parameters like learning rate, dropout rate, and batch size to optimize the model's performance. It's a crucial step in achieving better results. The output of a GRU model is determined by its ability to learn and remember patterns in sequential data. Properly tuned hyperparameters and well-chosen architecture lead to a model that generalizes well to unseen data, providing meaningful predictions or classifications. It's essential to strike a balance during hyperparameter tuning to prevent overfitting or underfitting and ensure the GRU effectively captures temporal dependencies in the input data.

List of hyperparameters: Accuracy, Precision, F1-score, support

Screenshot:



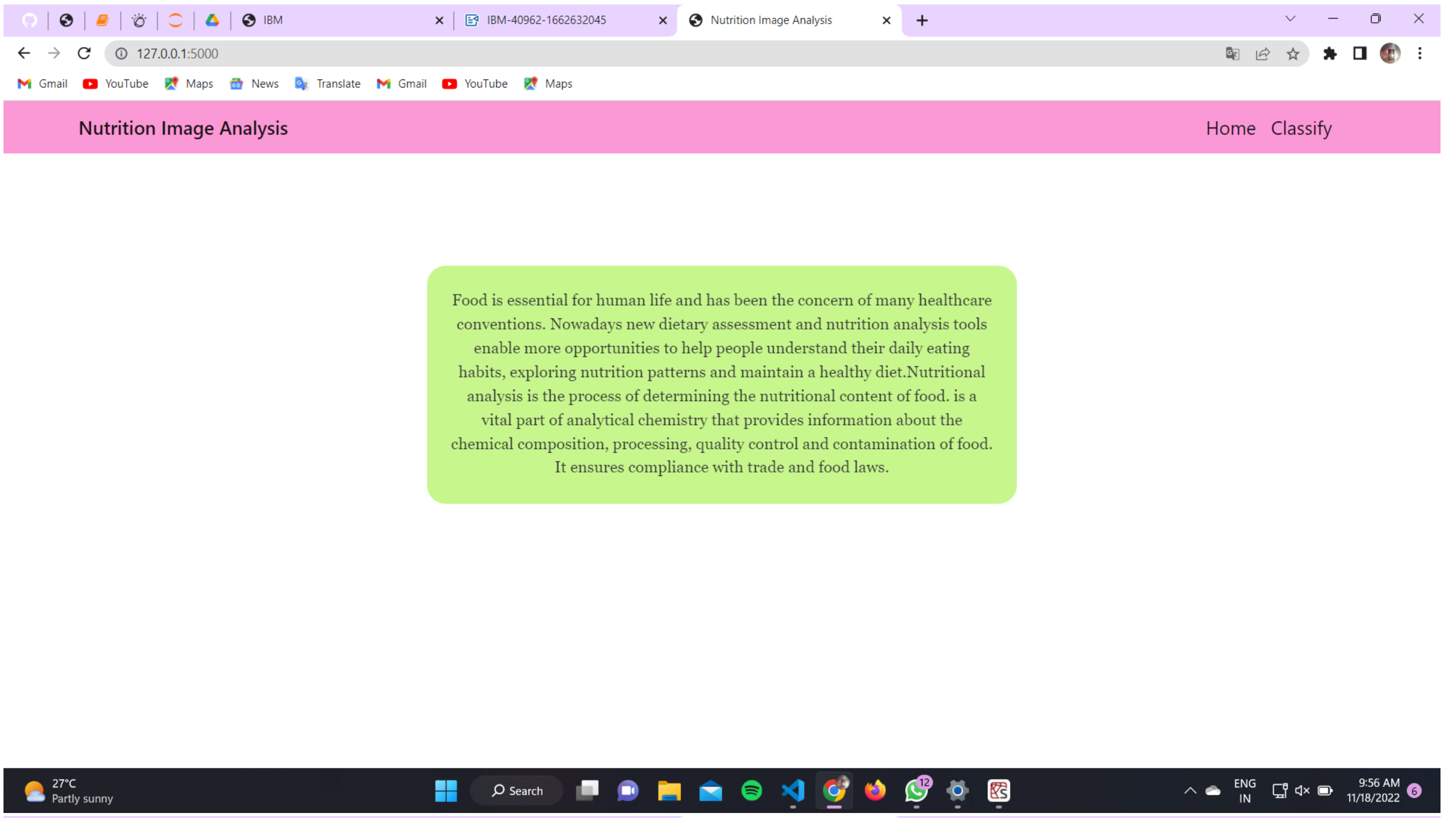
```
The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

7/7 [==============================] - 5s 194ms/step - loss: 0.5243 - accuracy: 0.0091 - val_loss: 0.2227 - val_accuracy: 0.0000e+00
Epoch 2/10
7/7 [==============================] - 0s 19ms/step - loss: 0.2350 - accuracy: 0.0091 - val_loss: -0.0866 - val_accuracy: 0.0000e+00
Epoch 3/10
7/7 [==============================] - 0s 18ms/step - loss: -0.0594 - accuracy: 0.0091 - val_loss: -0.4195 - val_accuracy: 0.0000e+00
Epoch 4/10
7/7 [==============================] - 0s 23ms/step - loss: -0.3780 - accuracy: 0.0091 - val_loss: -0.7697 - val_accuracy: 0.0000e+00
Epoch 5/10
7/7 [==============================] - 0s 22ms/step - loss: -0.7167 - accuracy: 0.0091 - val_loss: -1.1415 - val_accuracy: 0.0182
Epoch 6/10
7/7 [==============================] - 0s 19ms/step - loss: -1.0815 - accuracy: 0.0273 - val_loss: -1.5537 - val_accuracy: 0.0182
Epoch 7/10
7/7 [==============================] - 0s 18ms/step - loss: -1.4733 - accuracy: 0.0273 - val_loss: -2.0096 - val_accuracy: 0.0182
Epoch 8/10
7/7 [==============================] - 0s 19ms/step - loss: -1.9155 - accuracy: 0.0273 - val_loss: -2.5016 - val_accuracy: 0.0182
Epoch 9/10
7/7 [==============================] - 0s 25ms/step - loss: -2.3931 - accuracy: 0.0273 - val_loss: -3.0396 - val_accuracy: 0.0182
Epoch 10/10
7/7 [==============================] - 0s 23ms/step - loss: -2.9153 - accuracy: 0.0273 - val_loss: -3.6255 - val_accuracy: 0.0182
```



Inference:

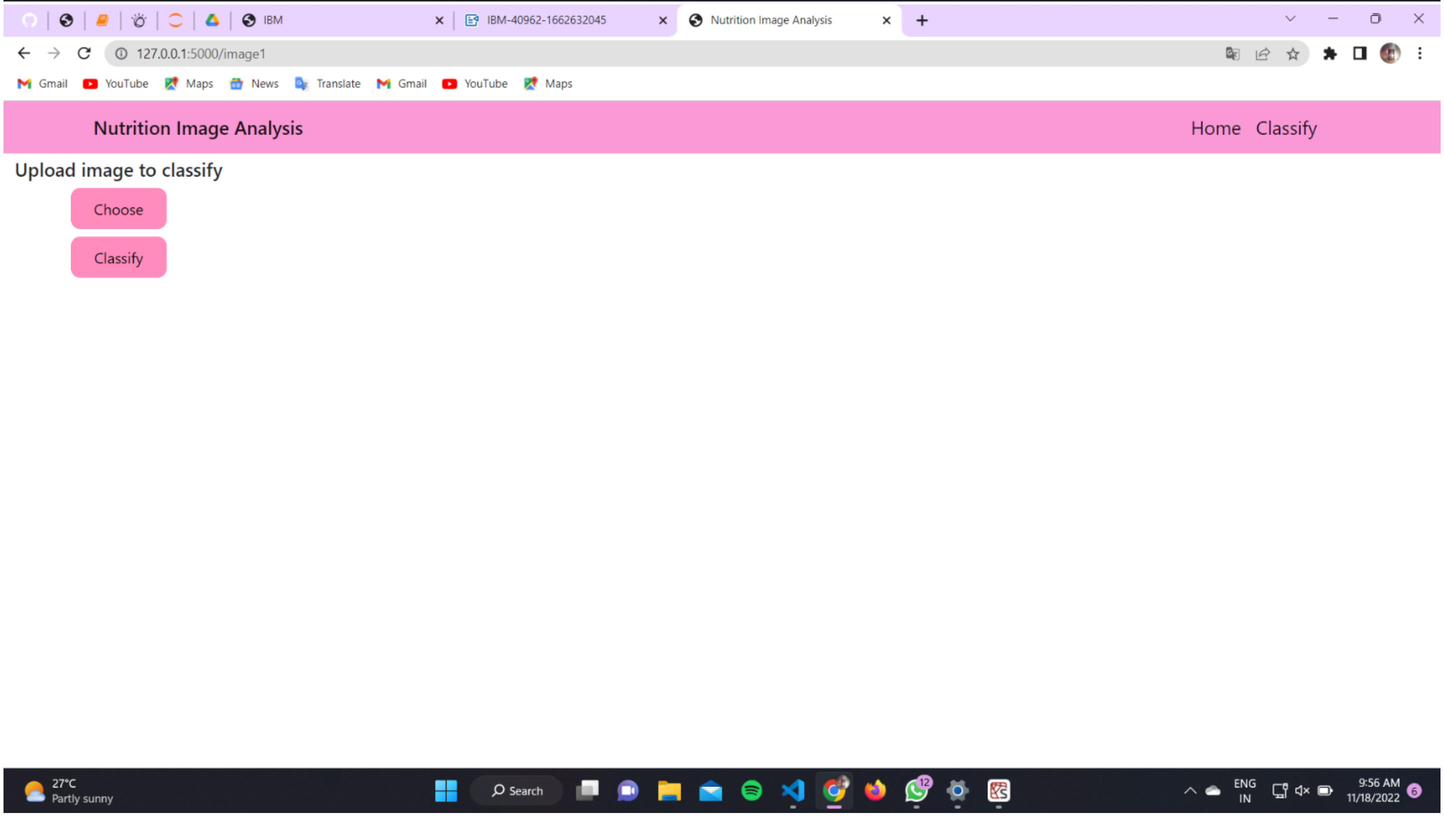
GRU (Gated Recurrent Unit) in terms of parameter settings, hyperparameter tuning, and output. When configuring a GRU, one must set parameters such as the number of hidden units, input size, and potentially other architectural details. These choices influence the model's capacity to capture patterns in sequential data. Hyperparameter tuning involves adjusting parameters like learning rate, dropout rate, and batch size to optimize the model's performance. It's a crucial step in achieving better results. The output of a GRU model is determined by its ability to learn and remember patterns in sequential data. Properly tuned hyperparameters and well-chosen architecture lead to a model that generalizes well to unseen data, providing meaningful predictions or classifications. It's essential to strike a balance during hyperparameter tuning to prevent overfitting or underfitting and ensure the GRU effectively captures temporal dependencies in the input data.

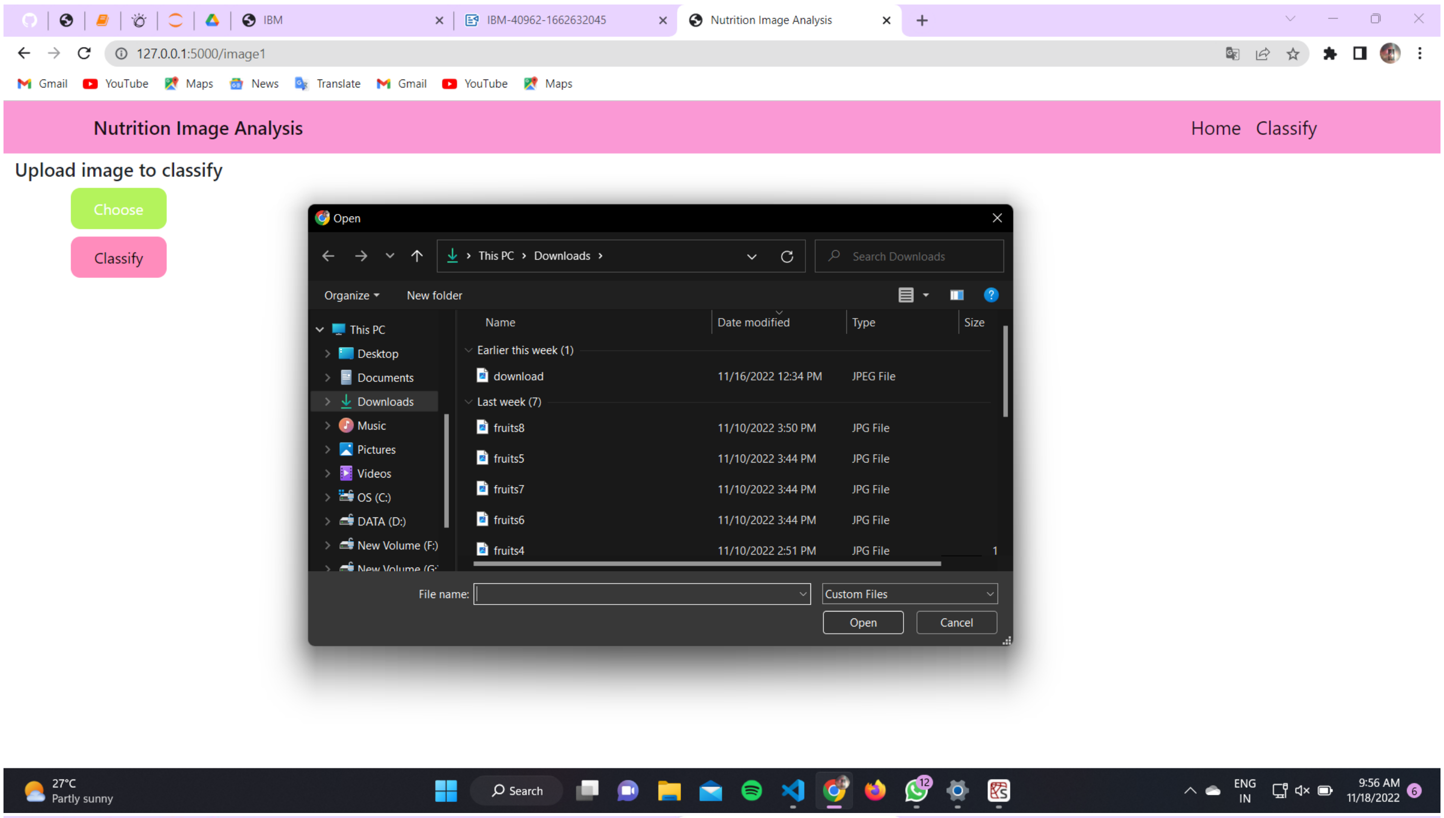
APPLICATION:



Nutrition Image Analysis Home Classify

Food is essential for human life and has been the concern of many healthcare conventions. Nowadays new dietary assessment and nutrition analysis tools enable more opportunities to help people understand their daily eating habits, exploring nutrition patterns and maintain a healthy diet.Nutritional analysis is the process of determining the nutritional content of food. is a vital part of analytical chemistry that provides information about the chemical composition, processing, quality control and contamination of food. It ensures compliance with trade and food laws.




127.0.0.1:5000/image1

Nutrition Image Analysis   Home Classify

Upload image to classify

Choose

Classify

Open

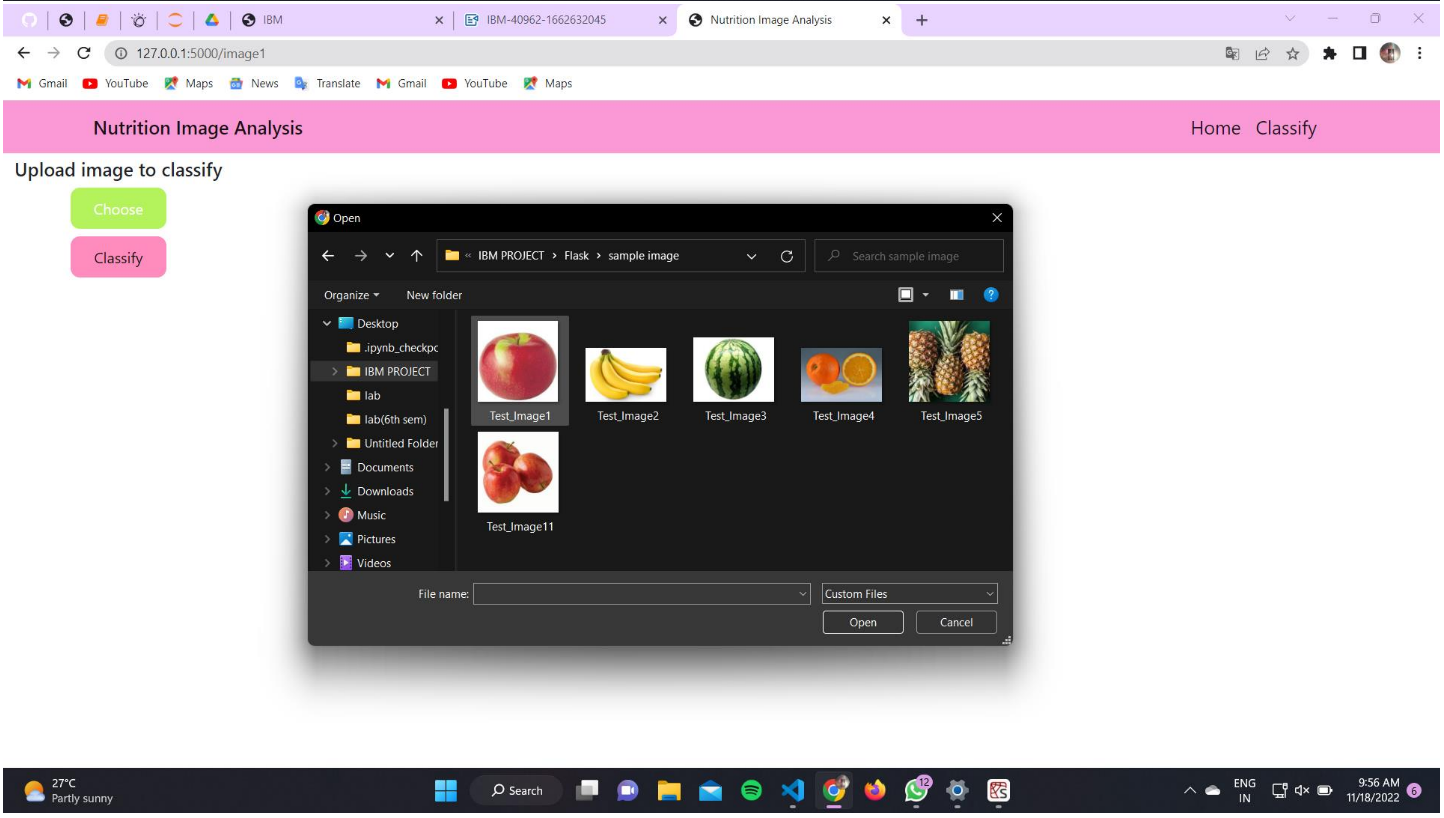
This PC › Downloads ›

Name	Date modified	Type	Size
Earlier this week (1)			
download	11/16/2022 12:34 PM	JPEG File	
Last week (7)			
fruits8	11/10/2022 3:50 PM	JPG File	
fruits5	11/10/2022 3:44 PM	JPG File	
fruits7	11/10/2022 3:44 PM	JPG File	
fruits6	11/10/2022 3:44 PM	JPG File	
fruits4	11/10/2022 2:51 PM	JPG File	1

File name: Custom Files

Open Cancel

27°C Partly sunny   ENG IN 9:56 AM 11/18/2022


127.0.0.1:5000/image1

Nutrition Image Analysis   Home Classify

Upload image to classify

Choose

Classify

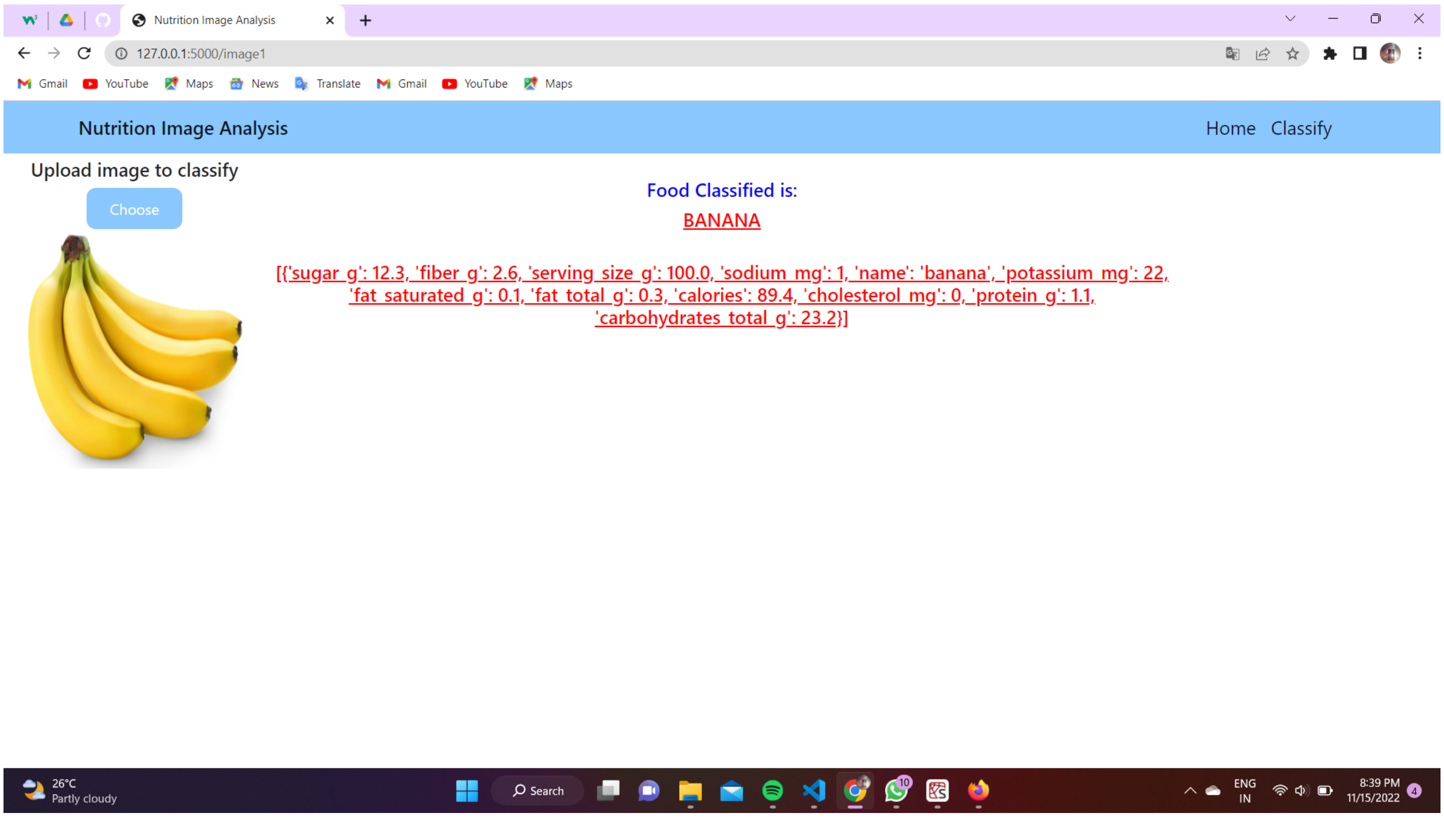
Open

« IBM PROJECT › Flask › sample image

File name: Custom Files

Open Cancel

27°C Partly sunny   ENG IN 9:56 AM 11/18/2022



Nutrition Image Analysis

Upload image to classify

Food Classified is:

BANANA

```
[{'sugar_g': 12.3, 'fiber_g': 2.6, 'serving_size_g': 100.0, 'sodium_mg': 1, 'name': 'banana', 'potassium_mg': 22,
'fat_saturated_g': 0.1, 'fat_total_g': 0.3, 'calories': 89.4, 'cholesterol_mg': 0, 'protein_g': 1.1,
'carbohydrates_total_g': 23.2}]
```

Mark Split-up:

	Marks	Marks Awarded
Powerpoint	5 Marks	
Preprocessing --Statistical Feature Analysis --Noise Removal (SMOTE Algorithm) --One Hot Encoding		
Regression --Linear --Logistic -- Ridge Regression -- Lasso Regression	15 Marks	
Classifiers --K-Means with KNN --Fuzzy C-Means with KNN --K-Means with SVM --Bayesian Classifier --Naïve Bayes Classifier --Decision Tree		
Deep learning --K-Means with CNN --Alex Net --LSTM --GRU		
Report	10 Marks	