PERIYAR MANIAMMAI INSTITUTE OF SCIENCE & TECHNOLOGY

PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited
think • innovate • transform

RECORD NOTE BOOK

COURSE NAME	
COURSE CODE	
STUDENT NAME	
REGISTER NUMBER	
BRANCH	
YEAR	
SEMESTER	

PERIYAR MANIAMMAI INSTITUTE OF SCIENCE & TECHNOLOGY

PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited
think • innovate • transform

CERTIFICATE

Register No :

Certified that this Bona Records of the work done by

_____ In the **APPLIED ARTIFICIAL INTELLIGENCE**

Laboratory Of the Department of _____

During the Academic Year **20__ – 20__**

Programme : Year : Semester :

Laboratory/Course in-Charge
(With date)

Head of the Department
(with date and seal)

Name:

Sumbitted for the Practical Examination held on_____

Internal Examiner
(with Date)
Name:

External Examiner
(with Date)
Name:

INDEX

Exp – 1 Crash Course on Python

Aim:

To practice fundamental operations and syntax of Python by covering key concepts such as variables, data types, control flow, functions, and basic data manipulation.

Code:

Data Types :

Integers (int) - Whole numbers without a fractional component.
x = 5

Floating-point numbers (float) - Numbers with a decimal point or in exponential form. y = 3.14

Strings (str) - Ordered sequence of characters enclosed in single or double quotes. name = "Data"

Booleans (bool)- Represents either True or False.
is_valid = True

Lists (list) - Ordered, mutable sequence of elements.
numbers = [1, 2, 3, 4]

Tuples (tuple)- Ordered, immutable sequence of elements.
coordinates = (3, 5)

Dictionaries (dict) - Unordered collection of key-value pairs.
person = {"name": "Alice", "age": 30}

Sets (set) - Unordered collection of unique elements.
unique_numbers = {1, 2, 3, 4}

Assigning values to variables

```
age = 22
name = "Chandra"
is_student = True
height = 5.65
```

Using the print function to display the information

```
print(age) print(name)
```

Using the type function to determine variable types

```
print("Type of 'is_student':", type(is_student)) print("Type of 'height':", type(height))
```

Basic Operations

Addition

```
add = a + b
print("Addition:", add)
```

Subtraction

```
sub = a - b
print("Subtraction:", sub)

Multiplication
mul = a * b
print("Multiplication:", mul)

Modulo (remainder after division)
 mod = a % b
print("Modulo:", mod)

Exponentiation (a to the power of b)
 power = a ** b
print("Power:", power)
```

Bool

```
x = 10
y = 10
print(bool(x == y)
```

Getting input from the user

```
num1 = input("Enter the first number: ")
num2 = input("Enter the second number: ") print("Entered
numbers:", num1, num
```

Tokens

- Tokens are the smallest units in a program. They are the building blocks of a Python program.
- Examples include keywords, identifiers, literals, operators, and punctuation.

Reserved Words (Keywords)

- Reserved words are predefined words that have special meanings in Python and cannot be used as identifiers (variable names).
- Examples: `if`, `else`, `for`, `while`, `True`, `False`, `def`, `class`, etc.

Identifiers

- Identifiers are names given to entities in a Python program. They can be variable names, function names, class names, etc.
- Rules for identifiers: They must start with a letter (a-z, A-Z) or an underscore (_), followed by letters, numbers, or underscores.

Literals:

- Literals are constant values used in Python. They are raw data given in a variable or constant.
- Examples: Numeric literals (e.g., `42`, `3.14`), string literals (e.g., `"hello"`), boolean literals (`True` or `False`), and special literals (`None`).

Operators

Arithmetic

Operators

```
a = 10
b = 3

Addition addition_result = a + b
print("Addition:", addition_result) # Output: 13

Subtraction subtraction_result = a - b
print("Subtraction:", subtraction_result) # Output: 7

Multiplication
multiplication_result = a * b
print("Multiplication:", multiplication_result) # Output: 30

Division division_result = a / b
print("Division:", division_result) # Output: 3.3333333333333335

Modulo (Remainder)
modulo_result = a % b
print("Modulo:", modulo_result) # Output: 1

Exponentiation
exponentiation_result = a ** b
print("Exponentiation:", exponentiation_result) # Output: 1000
```

Relational Operators

```
x = 5
y = 8

Equal to
equal_result = x == y
print("Equal to:", equal_result) # Output: False

Not Equal to
not_equal_result = x != y
print("Not Equal to:", not_equal_result) # Output: True

Greater than
greater_than_result = x > y
print("Greater than:", greater_than_result) # Output: False

Less than
less_than_result = x < y
print("Less than:", less_than_result) # Output: True

Greater than or Equal to
greater_than_equal_result = x >= y
print("Greater than or Equal to:", greater_than_equal_result) # Output: False

Less than or Equal to
```

```
less_than_equal_result = x <= y
print("Less than or Equal to:", less_than_equal_result) # Output: True
```

Assignment Operators

```
z = 15

Addition
Assignment z +=
5
print("Addition Assignment:", z) # Output: 20

Subtraction
Assignment z -= 3
print("Subtraction Assignment:", z) #Output: 17

 Multiplication
 Assignment z *= 2
 print("Multiplication Assignment:", z) # Output: 34

 Division
 Assignment z /=
 4
 print("Division Assignment:", z) # Output: 8.5

 Modulo
 Assignment z %= 7
 print("Modulo Assignment:", z) # Output: 1.5
```

Logical Operators

```
 p = True
 q = False

 Logical AND
 logical_and_result = p and q
 print("Logical AND:", logical_and_result) # Output: False

 Logical OR
 logical_or_result = p or q
 print("Logical OR:", logical_or_result) # Output: True

 Logical NOT
 logical_not_result = not p
 print("Logical NOT:", logical_not_result) # Output: False

Bitwise XOR
bitwise_xor_result = m ^ n
 print("Bitwise XOR:", bitwise_xor_result) # Output: 6


 Bitwise NOT
 bitwise_not_result = ~m
```

```
print("Bitwise NOT:", bitwise_not_result) # Output: -6

Left Shift
left_shift_result = m << 1
print("Left Shift:", left_shift_result) # Output: 10

Right Shift
right_shift_result = m >> 1
print("Right Shift:", right_shift_result) # Output: 2
```

Conditional Statements

Simple 'if' statement

```
x = 10
if x > 5:
    print("x is greater than 5.")
    # Output: x is greater than 5.
```

if-else' statement

```
y = 3
if y % 2 == 0: print("y is even.")
else:
    print("y is odd.")
    # Output: y is odd.
```

'if-elif-else' statement

```
z = 0
if  z > 0:
    print("z is positive.")
elif  z < 0:
    print("z is negative.")
else:
    print("z is zero.")
    # Output: z is zero.
```

Nested 'if' statements

```
a = 12
if a > 10:
    print("a is greater than 10.")
    if a % 2 == 0:
        print("a is even.")
        # Output: a is even.
         else:
        print("a is odd.")
```

Relation Between Values And Comparing

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
```

```
if num1 > num2:
    print(f"{num1} is greater than {num2}.")
    elif num1 < num2:
    print(f"{num1} is less than {num2}.")
    else:
    print(f"{num1} and {num2} are equal.")
```

Output:
Enter the first number: 1
Enter the second number: 4
1.0 is less than 4.0.

Checking the divisibility by 2

```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print(f"{num} is divisible by 2.")
    else:
    print(f"{num} is not divisible by 2.")
```

Output:
Enter a number: 56
56 is divisible by 2.

For Loop

```
a = {1, 2, 3,}
for i in a: print("HELLO!")
```
Output: HELLO! HELLO! HELLO!

Iterates range of numbers using range()

```
for i in range(10):
print("Data")
```
Output:
Data Data Data

Using for , if, and else combination

```
for number in range(1, 6):
 if number % 2 == 0:
        print(number, "is an even number.")
        else:
        print(number, "is an odd number.")
```

Output:
1 is an odd number.
2 is an even number.
3 is an odd number.

4 is an even number.
5 is an odd number.

Break statement:

```
for i in range(10):
if i == 5:
        break
        print(i)
        Output:
0
1
2
3
4
```

Pass statement:

```
for i in
    range(3):
    pass
# No output since pass does nothing
```

Continue statement:

```
for letter in 'Python':
if letter == 'h':
        continue
        print(letter)
Output: Py t o n
```

While loop

```
count = 0
while True:
    print(count)
    count+=1
    if count>=5:
     break

Output :
0
1
2
3
4
```

Calculates the sum of numbers from 0 to 'n'

```
n = input("Enter the number = ")
val = 0
i = 0
```

```
while i <= int(n): val += i
   i += 1
    print(f"The sum is {val}")
```

Output:
Enter the number = 5
The sum is 0
The sum is 1
The sum is 3
The sum is 6
The sum is 10
The sum is 15
The sum is 21

Functions

```
def intro():
    print("Hello World !")
    intro()
```

Output:
Hello World !

Built-in Functions

```
len():
    my_list = [1, 2, 3, 4, 5]
    length = len(my_list)
    print(length)
  # Output: 5

max():
    numbers = [5, 2, 8, 1, 6]
    maximum = max(numbers)
    print(maximum)
  # Output: 8

min():   numbers = [5, 2, 8, 1, 6]
minimum = min(numbers)
print(minimum)
# Output: 1

sum():
    numbers = [1, 2, 3, 4, 5]
    total = sum(numbers)
    print(total)
# Output: 15

abs():
absolute_value = abs(-7)
Print(absolute_value)
# Output: 7
```

```
sorted():
    numbers = [5, 2, 8, 1, 6]
    sorted_numbers = sorted(numbers)
    print(sorted_numbers)
# Output: [1, 2, 5, 6, 8]
```

List

Creating a list

```
fruits = ['apple', 'orange', 'banana', 'grape']
```

Accessing elements

```
first_fruit = fruits[0]

print(first_fruit)

# Output: 'apple'
```

Slicing

```
subset_fruits = fruits[1:3]
print(subset_fruits)
# Output: ['orange', 'banana']
```

Modifying elements

```
fruits[1] = 'pear' print(fruits)

# Output: ['apple', 'pear', 'banana', 'grape']
```

Adding elements

```
fruits.append('kiwi')

print(fruits)

# Output: ['apple', 'pear', 'banana', 'grape', 'kiwi']
```

Removing elements

```
removed_fruit = fruits.pop(2)

print(removed_fruit)

# Output: 'banana' print(fruits)
# Output: ['apple', 'pear', 'grape', 'kiwi']
```

Concatenation

```
more_fruits = ['pineapple', 'mango']
all_fruits = fruits + more_fruits print(all_fruits)
# Output: ['apple', 'pear', 'grape', 'kiwi', 'pineapple', 'mango']
```

Length of the list

```
num_fruits = len(all_fruits)
print(num_fruits)
# Output: 6
```

Check if an item is in the list

```
is_mango_present = 'mango' in all_fruits
print(is_mango_present)
# Output: True
```

Tuple

Accessing Elements:

```
my_tuple = (1, 2, 3, 4, 5)
first_element = my_tuple[0]
print(first_element)
# Output: 1
```

Slicing:

```
my_tuple = (1, 2, 3, 4, 5)
subset_tuple = my_tuple[1:4]
print(subset_tuple)
# Output: (2, 3, 4)
```

Concatenation:

```
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
concatenated_tuple = tuple1 + tuple2
print(concatenated_tuple)
# Output: (1, 2, 3, 4, 5, 6)
```

Length of the Tuple:

```
my_tuple = (1, 2, 3, 4, 5)
tuple_length = len(my_tuple)
print(tuple_length)
# Output: 5
```

Checking for Membership:

```
my_tuple = (1, 2, 3, 4, 5)
is_present = 3
in my_tuple print(is_present)
```

```
# Output: True
```

Tuple Unpacking:

```
coordinates = (3, 4)
x, y = coordinates
  print(f"x = {x}, y = {y}")
  # Output: x = 3, y = 4
```

Count occurrences of an element:

```
my_tuple = (1, 2, 2, 3, 4, 2)
count_of_2 = my_tuple.count(2)
print(count_of_2)
# Output: 3
```

Find the index of an element:

```
  my_tuple = (1, 2, 3, 4, 5)
  index_of_3 = my_tuple.index(3)
  print(index_of_3)
   # Output: 2
```

Minimum and Maximum:

```
numbers = (5, 2, 8, 1, 6)
min_value = min(numbers)
max_value = max(numbers)
print(min_value, max_value)
# Output: 1 8
```

Set

Creating sets

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
```

Adding elements

```
set1.add(6)
print(f"Set 1 after adding 6: {set1}")
# Output: Set 1 after adding 6: {1, 2, 3, 4, 5, 6}
```

Removing elements

```
set1.remove(3)
print(f"Set 1 after removing 3: {set1}")
# Output: Set 1 after removing 3: {1, 2, 4, 5, 6}
```

Union

```
union_set = set1.union(set2)
```

```
print(f"Union Set: {union_set}")
# Output: Union Set: {1, 2, 4, 5, 6, 7, 8}
```

Intersection

```
intersection_set = set1.intersection(set2)
print(f"Intersection Set:
{intersection_set}") # Output: Intersection
Set: {4, 5, 6}
```

Difference

```
difference_set = set1.difference(set2)
 print(f"Difference Set (set1 - set2): {difference_set}")
# Output: Difference Set (set1 - set2): {1, 2}
```

Dictionary

Creating a dictionary

```
my_dict = {'name': 'Nataraj', 'age': 25, 'city': 'Tnj'}
```

Accessing elements

```
 name_value = my_dict['name']
 print(f"Name: {name_value}")
 # Output: Name: Nataraj
```

Modifying elements

```
my_dict['age'] = 26
print(f"Updated Age: {my_dict['age']}")
 # Output: Updated Age: 26
```

Adding a new key-value pair

```
my_dict['gender'] = 'Male'
print(f"Dictionary after adding 'gender': {my_dict}")
# Output: Dictionary after adding 'gender': {'name': 'Nataraj', 'age': 26, 'city': 'Tnj', 'gender': 'Male'}
```

Removing a key-value pair

```
removed_age = my_dict.pop('age')

print(f"Removed Age: {removed_age}")

print(f"Dictionary after removing 'age': {my_dict}")
# Output: Removed Age: 26
Dictionary after removing 'age': {'name': 'Nataraj', 'city': 'Tnj', 'gender': 'Male'}
```

Checking if a key is present

```
is_city_present = 'city' in my_dict
print(f"Is 'city' present: {is_city_present}")
# Output: Is 'city' present: True
```

Getting all keys and values

```
all_keys = my_dict.keys()

all_values = my_dict.values()

print(f"All Keys: {list(all_keys)}")

print(f"All Values: {list(all_values)}")
#Output:
All Keys: ['name', 'city', 'gender']
All Values: ['Nataraj', 'Tnj', 'Male']
```

Iterating through keys and values

```
print("Iterating through Dictionary:")

for key, value in my_dict.items():

    print(f"{key}: {value}")
    # Output:
Iterating through Dictionary:
name: Nataraj
city: Tnj
 gender: Male
```

Copying a dictionary

```
copied_dict = my_dict.copy()
print(f"Copied Dictionary: {copied_dict}")
# Output: Copied Dictionary: {'name': 'Nataraj', 'city': 'Tnj', 'gender': 'Male'}
```

Clearing all elements

```
my_dict.clear()
print(f"Dictionary after clearing: {my_dict}")
 # Output: Dictionary after clearing: {}
```

Result :

Thus, we have performed the basic operations in python .

Exp- 2 **Working with NumPy & Pandas**

Aim: To practice fundamental operations with NumPy and Pandas in Python, utilizing NumPy for array manipulation and mathematical operations, and Pandas for efficient data handling.

Code :

NumPy Library – Numerical computing Install the NumPy library using pip

```
pip install numpy
```

Import the Pandas library

```
import numpy as np
```

Creating a one-dimensional array

```
my_array = np.array([1, 2, 3, 4, 5])
```

Creating a two-dimensional array

```
arr2 = np.array([(1, 2, 3), (4, 5, 6)])
```

Creating a NumPy array of zeros with shape

```
zeros_array = np.zeros((5, 2))
```

Creating a NumPy array of ones with shape

```
ones_array=np.ones((2,3))
```

Create a NumPy array using linspace, generating 100 evenly spaced values between 0 and 5 (inclusive)

```
array = np.linspace(0, 5, 100)
```

Transpose the array

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
transposed_arr = np.transpose(arr)
```

Flatten the array into a 1D array

```
arr = np.array([[1, 2, 3], [4, 5, 6]])

flattened_arr = arr.flatten()
```

Using ravel() to flatten the array

```
 my_matrix = np.array([[1, 2, 3], [4, 5, 6]])

flattened_array = my_matrix.ravel()
```

Reshaping an array

```
arr = np.array([1, 2, 3, 4, 5, 6])
reshaped_arr = arr.reshape(3, 2)
```

Pandas Library – Data Manipulation and Analysis

Install the Pandas library using pip

```
pip install pandas
```

Import the Pandas library

```
 import pandas as pd
```

Creating series

```
a = [1, 2, 3]
myset = pd.Series(a)
print(myset)
```

Creating a Series with data and custom index

```
alphabet = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
print(alphabet)
```

Creating a dictionary with keys and their corresponding values

```
data = {'Subject': ['Physics', 'Chemistry', 'Biology', 'Maths'], 'CGPA': [8, 9, 9, 8]}
```

Creating a Data Frame

```
dataframe = pd.DataFrame(data, columns=['Subject', 'CGPA'], index=[1, 2, 3, 4])
```

Creating a random data frame

```
import numpy random=pd.DataFrame(np.random.randint(0,300,size=(20,4)),columns=list('ABCD))
```

Exporting the DataFrame to a CSV file

```
random.to_csv('random.csv')
```

Reading the CSV file

```
data = pd.read_csv('random.csv')
```

Result :

Thus, we have performed basic operations with NumPy and Pandas library.

Exp-3 Building a KNN model in Scikit Learn

Aim: The primary aim in building the KNN model using Scikit-Learn is to achieve accurate predictions and understand the model's performance.

Code:

```
import warnings
warnings.simplefilter('ignore')
```

In [2]:

```
pip install numpy
```

In [4]:

```
import numpy as np
import pandas as pd
import data visualization library

pip install matplotlib

pip install seaborn
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

importing dataset

pip install openpyxl
ds=pd.read_excel('student_dataset.xlsx')
ds
```

	S.NO	ATTN	THEORY	Daily assn	TOTAL	certificate
0	1	20	38	25	83	DISTINCTION
1	2	15	25	0	40	COMPLETION
2	3	20	31	25	76	FIRST CLASS
3	4	20	25	25	70	FIRST CLASS
4	5	20	17	15	52	FIRST CLASS
...	...	...	...	...	...	...
362	363	15	0	10	25	PARTICIPATION
363	364	20	9	25	54	COMPLETION
364	365	15	20	25	60	FIRST CLASS

365	366	15	0	10	25	PARTICIPATION
366	367	20	23	10	53	COMPLETION

367 rows × 6 columns

ds.shape

(367, 6)

ds.head()

	S.NO	ATTN	THEORY	Daily assn	TOTAL	certificate
0	1	20	38	25	83	DISTINCTION
1	2	15	25	0	40	COMPLETION
2	3	20	31	25	76	FIRST CLASS
3	4	20	25	25	70	FIRST CLASS
4	5	20	17	15	52	FIRST CLASS

In [13]:

ds.tail()

Out[13]:

	S.NO	ATTN	THEORY	Daily assn	TOTAL	certificate
362	363	15	0	10	25	PARTICIPATION
363	364	20	9	25	54	COMPLETION
364	365	15	20	25	60	FIRST CLASS
365	366	15	0	10	25	PARTICIPATION
366	367	20	23	10	53	COMPLETION

slice the dataset

In [14]:

data=ds.drop(['S.NO','TOTAL'],axis=1)

In [15]:

data

Out[15]:

	ATTN	THEORY	Daily assn	certificate
0	20	38	25	DISTINCTION
1	15	25	0	COMPLETION
2	20	31	25	FIRST CLASS

3	20	25	25	FIRST CLASS
4	20	17	15	FIRST CLASS
...	...	...	...	...
362	15	0	10	PARTICIPATION
363	20	9	25	COMPLETION
364	15	20	25	FIRST CLASS
365	15	0	10	PARTICIPATION
366	20	23	10	COMPLETION

367 rows × 4 columns

unique value in dataset

In [16]:

```
data['ATTN'].unique()
```

Out[16]:

```
array([20, 15,  1, 25, 10], dtype=int64)
```

grouping the dataset

In [17]:

```
data.groupby('certificate').size()
```

Out[17]:

```
certificate
COMPLETION       117
DISTINCTION       32
FIRST CLASS       66
PARTICIPATION    152
dtype: int64
```

Encoding The Data

```
from sklearn.preprocessing import LabelEncoder
```

implementing label encoder

In [20]:

```
data.iloc[:,3]=LabelEncoder().fit_transform(data.iloc[:,3])
```

In [21]:

```
data
```

Out[21]:

	ATTN	THEORY	Daily assn	certificate

0	20	38	25	1
1	15	25	0	0
2	20	31	25	2
3	20	25	25	2
4	20	17	15	2
...	...	...	...	...
362	15	0	10	3
363	20	9	25	0
364	15	20	25	2
365	15	0	10	3
366	20	23	10	0

367 rows × 4 columns

0-compltion

1-distinction

2-first class

3-participation

reshape dataset to dataframe

In [22]:

```
x=data.iloc[:,:-1].values
```

In [23]:

```
x
```

Out[23]:

```
array([[20, 38, 25],
       [15, 25,  0],
       [20, 31, 25],
       ...,
       [15, 20, 25],
       [15,  0, 10],
       [20, 23, 10]], dtype=int64)
```

In [24]:

```
y=data.iloc[:,3]
```

In [25]:

```
y
```

Out[25]:

```
0     1
1     0
2     2
3     2
4     2
     ..
362   3
363   0
364   2
365   3
366   0
Name: certificate, Length: 367, dtype: int32
```

Converting To Dataframe

In [26]:

```
x_frame=pd.DataFrame(x)
```

In [27]:

```
x_frame
```

Out[27]:

	0	1	2
0	20	38	25
1	15	25	0
2	20	31	25
3	20	25	25
4	20	17	15
...	...	...	...
362	15	0	10
363	20	9	25
364	15	20	25
365	15	0	10
366	20	23	10

367 rows × 3 columns

In [28]:

```
y_frame=pd.DataFrame(y)
```

In [29]:

```
y_frame
```

Out[29]:

	certificate
0	1
1	0
2	2
3	2
4	2
...	...
362	3
363	0
364	2
365	3
366	0

367 rows × 1 columns

4.divide the dataset into training set

In [30]:

```
from sklearn.model_selection import train_test_split
```

In [31]:

```
x_train,x_test,y_train,y_test=train_test_split(x_frame,y_frame,test_size=0.2,random_state=0)
```

KNN

In [32]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [33]:

```
knn=KNeighborsClassifier(n_neighbors=5)
```

In [34]:

```
knn.fit(x_train,y_train)
```

Out[34]:

```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

6.prediction

```
y_predict=knn.predict(x_test)
```

In [36]:

```
y_predict
```

Out[36]:

```
array([3, 2, 0, 3, 3, 3, 3, 3, 0, 0, 3, 0, 0, 2, 0, 1, 0, 2, 1, 3, 3, 0,
       1, 0, 3, 3, 3, 0, 0, 2, 3, 2, 3, 3, 0, 0, 2, 3, 3, 0, 0, 3, 0, 3,
       0, 0, 0, 0, 3, 3, 3, 2, 0, 1, 3, 0, 0, 3, 2, 0, 2, 0, 1, 0, 3, 0,
       0, 3, 0, 3, 0, 3, 3, 0])
```

model evaluation amd performance metrics

In [37]:

```
knn.score(x_train,y_train)
```

Out[37]:

```
0.7849829351535836
```

In [38]:

```
knn.score(x_test,y_test)
```

Out[38]:

```
0.6891891891891891
```

importing necessary metric

```
from sklearn.metrics import confusion_matrix
```

confusion matrix

```
 con_matrix=confusion_matrix(y_test,y_predict)
con_matrix
array([[15,  0,  2,  1],
       [ 0,  3,  2,  0],
       [ 7,  2,  5,  0],
       [ 9,  0,  0, 28]], dtype=int64)
```

Result:

Thus Building a KNN model in Scikit Learn is executed successfully.

Exp-4 Image recognition model using SVM and PCA

Aim:

The primary objective is to build an effective Image recognition model using support vector machines(svm)and principle component analysis(pca)

Code:

```
#import system libraries
import os
import warnings
warnings.simplefilter('ignore')
#import the data handling libraries
import numpy as np
import pandas as pd
#import data datavisualization library
import matplotlib.pyplot as plt
%matplotlib inline
from skimage.io import imread, imshow
from skimage.transform import resize
from skimage.color import rgb2gray
#setting working directories
dq=os.listdir("C://Users//aedpu//OneDrive//Documents//harshini//dq")
nani=os.listdir("C://Users//aedpu//OneDrive//Documents//harshini//nani")
vijay=os.listdir("C://Users//aedpu//OneDrive//Documents//harshini//vijay")
#reading the image as a matrix of numbers
limit=20
dq_images=[None]*limit
j=0
for i in dq:
    if(j<limit):
        dq_images[j]=imread("C://Users//aedpu//OneDrive//Documents//harshini//dq/"+i)
        j+=1
    else:
        break
limit=20
nani_images=[None]*limit
j=0
for i in nani:
    if(j<limit):
        nani_images[j]=imread("C://Users//aedpu//OneDrive//Documents//harshini//nani/"+i)
        j+=1
    else:
        break
vijay_images=[None]*limit
j=0
for i in vijay:
    if(j<limit):
        vijay_images[j]=imread("C://Users//aedpu//OneDrive//Documents//harshini//vijay/"+i)
        j+=1
    else:
        break
# view images
imshow(vijay_images[10])
<matplotlib.image.AxesImage at 0x2029de2e160>
imshow(nani_images[16])
```

```
<matplotlib.image.AxesImage at 0x2029de730a0>
imshow(dq_images[3])
<matplotlib.image.AxesImage at 0x2029deea370>

#gray scale

dq_gray=[None]*limit
j=0
for i in dq:
    if(j<limit):
        dq_gray[j]=rgb2gray(dq_images[j])
        j+=1
    else:
        break
nani_gray=[None]*limit
j=0
for i in nani:
    if(j<limit):
        nani_gray[j]=rgb2gray(nani_images[j])
        j+=1
    else:
        break
vijay_gray=[None]*limit
j=0
for i in vijay:
    if(j<limit):
        vijay_gray[j]=rgb2gray(vijay_images[j])
        j+=1
    else:
        break
#view the gray
imshow(dq_gray[3])
imshow(nani_gray[15])
imshow(vijay_gray[3])
#reshape
dq_gray[1].shape
nani_gray[2].shape
vijay_gray[3].shape
# resizing
for j in range (20):
    dq=dq_gray[j]
    dq_gray[j]=resize(dq,(512,512))
for j in range (20):
    nani=nani_gray[j]
    nani_gray[j]=resize(nani,(512,512))
for j in range (20) :
    vijay=vijay_gray[j]
    vijay_gray[j]=resize(vijay,(512,512))
#find out the number of gray scale images
len_of_images_dq=len(dq_gray)
len_of_images_dq
len_of_images_nani=len(nani_gray)
len_of_images_nani
len_of_images_vijay=len(vijay_gray)
len_of_images_vijay
#create a variable image size
image_size_dq=dq_gray[1].shape
image_size_dq
image_size_nani=nani_gray[1].shape
```

```
image_size_nani

image_size_vijay=vijay_gray[1].shape
image_size_vijay
#create a variable flatten image size which contains the product of(512,512)
flatten_size_dq=image_size_dq[0]*image_size_dq[1]
flatten_size_dq
flatten_size_nani=image_size_nani[0]*image_size_nani[1]
flatten_size_nani
flatten_size_vijay=image_size_vijay[0]*image_size_vijay[1]
flatten_size_vijay
#now flatten the image from(512,512)matrix to 262144,1 vector
for i in range(len_of_images_dq):
    dq_gray[i]=np.ndarray.flatten(dq_gray[i]).reshape(flatten_size_dq,1)
for i in range(len_of_images_nani):
 nani_gray[i]=np.ndarray.flatten(nani_gray[i]).reshape(flatten_size_nani,1)
for i in range(len_of_images_vijay):
    vijay_gray[i]=np.ndarray.flatten(vijay_gray[i]).reshape(flatten_size_vijay,1)
#now stack the individual image array elements into one array
dq_gray=np.stack(dq_gray)
dq__gray=np.rollaxis(dq_gray,axis=2,start=0)
dq_gray=dq_gray.reshape(len_of_images_dq,flatten_size_dq)
dq_gray.shape
#creating the dataframe of the image
dq_data=pd.DataFrame(dq_gray)
dq_data
#labelling the rows
dq_data["label"]="dq"
dq_data
#now stack the individual image array elements into one array nani_gray=np.stack(nani_gray)
nani__gray=np.rollaxis(nani_gray,axis=2,start=0)
nani_gray=nani_gray.reshape(len_of_images_nani,flatten_size_nani)
nani_gray.shape

#creating the dataframe of the image
nani_data=pd.DataFrame(nani_gray)
nani_data
#labelling the rows
nani_data["label"]="nani"
nani_data
#now stack the individual image array elements into one array
vijay_gray=np.stack(vijay_gray)
vijay_gray=np.rollaxis(vijay_gray,axis=2,start=0)
vijay_gray=vijay_gray.reshape(len_of_images_vijay,flatten_size_vijay)
vijay_gray.shape
#creating the dataframe of the image
vijay_data=pd.DataFrame(vijay_gray)
vijay_data
#labelling the rows
vijay_data["label"]="vijay"
vijay_data
#combining three dataframe to one dataframe
actor_1=pd.concat([dq_data,vijay_data])
actor=pd.concat([actor_1,nani_data])
actor
#shuffling
from sklearn.utils import shuffle
kollywood_indexed=shuffle(actor).reset_index()
```

```
kollywood_indexed

#drop
kollywood_actors=kollywood_indexed.drop(['index'],axis=1)
kollywood_actors

#save as csv file
kollywood_actors.to_csv("actors.csv")
#assigning dependent and independent variables
x=kollywood_actors.values[:,:-1]
y=kollywood_actors.values[:,-1]
x y
#assigning training and testing dataset
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
x_train.shape
x_test.shape
#principle compound analysis(pca) algorithm
from sklearn import decomposition
#componenet asssignment for pca
pca=decomposition.PCA(n_components=20, whiten=True,random_state=1)
#fititng the trainig set to generate principle components
pca.fit(x_train)
PCA(n_components=20, random_state=1, whiten=True)
#transforming principle components
x_train_pca=pca.transform(x_train)
x_test_pca=pca.transform(x_test)
x_train_pca.shape
x_test_pca.shape
#viewing the principle components or eigen faces
#reshape the image back to the original matrix
eigen=(np.reshape(x[10],(512,512)).astype(np.float64))
eigen
#plotting images one by one as subplots
fig=plt.figure(figsize=(30,30))
for i in range(10):
    ax=fig.add_subplot(2,5,i+1,xticks=[],yticks=[])
    ax.imshow(pca.components_[i].reshape(eigen.shape),cmap=plt.cm.bone)

#support vector machines algorithm

from sklearn import svm
clf=svm.SVC(C=2,gamma=0.006,kernel='rbf')
clf.fit(x_train_pca,y_train)
#image prediction
y_pred=clf.predict(x_test_pca)
y_pred
#prediction visualization
for i in (np.random.randint(0,6,6)):
    predicted_image=(np.reshape(x_test[i],(512,512)).astype(np.float64))
    plt.title('predicted label: {0}'.format(y_pred[i]))
    plt.imshow(predicted_image,interpolation='nearest',cmap='gray')
    plt.show()
#prediction accuracy
from sklearn import metrics
accuracy=metrics.accuracy_score(y_test,y_pred)
accuracy
```

Output:

```
[313]: from sklearn import metrics
       from sklearn.metrics import classification_report
       from sklearn.metrics import confusion_matrix

[314]: accuracy=metrics.accuracy_score(y_test,y_pred)

[315]: print(classification_report(y_test,y_pred))
       accuracy

                    precision    recall  f1-score   support

           arshath       1.00      1.00      1.00         2
           dhanush       1.00      1.00      1.00         2
         veeramani       1.00      1.00      1.00         2

          accuracy                           1.00         6
         macro avg       1.00      1.00      1.00         6
      weighted avg       1.00      1.00      1.00         6

[315]: 1.0

[316]: confusion_matrix(y_test,y_pred)

[316]: array([[2, 0, 0],
              [0, 2, 0],
              [0, 0, 2]])
```

Result:

Thus,we have performed image recognition using support vector machine and principle component analysis.

Exp-5 Emoji Classification using Random Classification Method

Aim: To Classify emojis using Random Classification Method for better accuracy andefficiency.

Code :

```
#Import System Libraries

import  os import warnings
warnings.simplefilter('ignore')

#Import the Datahandling Libraries

import numpy as npimport
pandas as pd

#Import DataVisualization Library

import matplotlib.pyplot as plt
%matplotlib inline
pip install scikit-image
from skimage.io import imread,imshowfrom
skimage.transform import resize from skimage.color
import rgb2gray


#Setting Working Directory

relaxed=os.listdir("C:/Users/aedpu/OneDrive/Documents/Butto/Fifth Sem/Emoji/Relaxed")
sassy=os.listdir("C:/Users/aedpu/OneDrive/Documents/Butto/Fifth Sem/Emoji/Sassy")
eyeroll=os.listdir("C:/Users/aedpu/OneDrive/Documents/Butto/Fifth Sem/Emoji/Eyeroll")


#Reading Image as a Matrix of Numbers

limit= 10 relaxed_images=[None]*limitj=0
for i in relaxed:if(j<limit):
        relaxed_images[j]=imread("C:/Users/aedpu/OneDrive/Documents/Butto/Fifth Sem/Emoji/Relaxed/"+i)
        j+=1
    else:
        break


limit=10 sassy_images=[None]*limitj=0
for i in sassy:if(j<limit):
        sassy_images[j]=imread("C:/Users/aedpu/OneDrive/Documents/Butto/Fifth Sem/Emoji/Sassy/"+i)
        j+=1
    else:
        break


limit=10 eyeroll_images=[None]*limitj=0
for i in eyeroll:if(j<limit):
        eyeroll_images[j]=imread("C:/Users/aedpu/OneDrive/Documents/Butto/Fifth Sem/Emoji/Eyeroll/"+i)
```

```
        j+=1
    else:
        break
#View the Images
imshow(sassy_images[2])
imshow(relaxed_images[0])
imshow(eyeroll_images[0])
#Gray Scale
relaxed_gray=[None]*limitj=0
for i in relaxed:if(j<limit):
        relaxed_gray[j]=rgb2gray(relaxed_images[j])j+=1
    else:
        break
sassy_gray=[None]*limitj=0
for i in sassy:if(j<limit):
        sassy_gray[j]=rgb2gray(sassy_images[j])j+=1
    else:
        break
eyeroll_gray=[None]*limitj=0
for i in eyeroll:if(j<limit):
        eyeroll_gray[j]=rgb2gray(eyeroll_images[j])j+=1
    else:
        break

#View the Gray imshow(sassy_gray[0])
imshow(relaxed_gray[0])
imshow(eyeroll_gray[9])
#Reshape
relaxed_gray[2].shape
sassy_gray[2].shape
eyeroll_gray[2].shape
#Resize

for j in range (10): sas =
    sassy_gray[j]
    sassy_gray[j]=resize(sas,(512,512))
imshow(sassy_gray[7])

for j in range (10): relax=relaxed_gray[j]
    relaxed_gray[j]=resize(relax,(512,512))imshow(relaxed_gray[7])
 for j in range (10):
    eye=eyeroll_gray[j] eyeroll_gray[j]=resize(eye,(512,512))
imshow(eyeroll_gray[7])

#Find out the number of Gray_scale Images

len_of_images_relaxed=len(relaxed_gray)len_of_images_relaxed

len_of_images_sassy=len(sassy_gray)
len_of_images_sassy

len_of_images_eyeroll=len(eyeroll_gray)len_of_images_eyeroll
```

#Create a Variable Image Size

```
image_size_relaxed=relaxed_gray[1].shapeimage_size_relaxed
image_size_sassy=sassy_gray[1].shape
image_size_sassy
image_size_eyeroll=eyeroll_gray[1].shape
image_size_eyeroll
```

#Create a variable flatten image size which contains the product of(512,512)

```
flatten_size_relaxed=image_size_relaxed[0]*image_size_relaxed[1] flatten_size_relaxed
flatten_size_sassy=image_size_sassy[0]*image_size_sassy[1]flatten_size_sassy
flatten_size_eyeroll=image_size_eyeroll[0]*image_size_eyeroll[1]
flatten_size_eyeroll
```

#Now flatten the image from(512,512) matrix to 262144,1 vector

```
for i in range(len_of_images_relaxed):
    relaxed_gray[i]=np.ndarray.flatten(relaxed_gray[i]).reshape(flatten_size_relaxed,1)
for i in range(len_of_images_sassy): sassy_gray[i]=np.ndarray.flatten(sassy_gray[i]).reshape(flatten_size_sassy,1)
for i in range(len_of_images_eyeroll):
    eyeroll_gray[i]=np.ndarray.flatten(eyeroll_gray[i]).reshape(flatten_size_eyeroll,1)
```

#Now Stack the individual image array elements into one array

```
relaxed_gray=np.stack(relaxed_gray) relaxed_gray=np.rollaxis(relaxed_gray,axis=2,start=0)
relaxed_gray=relaxed_gray.reshape(len_of_images_relaxed,flatten_size_relaxed) relaxed_gray.shape
```

#Creating a Dataframe of the Image Vector

```
relaxed_data=pd.DataFrame(relaxed_gray)relaxed_data
```

#Labelling the rows of the database

```
relaxed_data["label"]="relaxed"relaxed_data
```

#Now Stack the individual image array elements into one array

```
sassy_gray=np.stack(sassy_gray) sassy_gray=np.rollaxis(sassy_gray,axis=2,start=0)
sassy_gray=sassy_gray.reshape(len_of_images_sassy,flatten_size_sassy)sassy_gray.shape
```

#Creating a Dataframe of the Image Vector

```
sassy_data=pd.DataFrame(sassy_gray)sassy_data
```

#Labelling the rows of the database

```
sassy_data["label"]="sassy"sassy_data
```

#Now Stack the individual image array elements into one array

```
eyeroll_gray=np.stack(eyeroll_gray) eyeroll_gray=np.rollaxis(eyeroll_gray,axis=2,start=0)
eyeroll_gray=eyeroll_gray.reshape(len_of_images_eyeroll,flatten_size_eyeroll) eyeroll_gray.shape
```

#Creating a Dataframe of the Image Vector

```
eyeroll_data = pd.DataFrame(eyeroll_gray)eyeroll_data
```

```
#Labelling the rows of the database

eyeroll_data["label"]="eyeroll"eyeroll_data

#Three dataframes into one dataframes

emoji_1=pd.concat([eyeroll_data,relaxed_data])

emoji=pd.concat([emoji_1,sassy_data])

emoji

#Shuffling

from sklearn.utils import shuffle

collection_indexed=shuffle(emoji).reset_index()collection_indexed

#Drop collection_emoji=collection_indexed.drop(['index'],axis=1)

collection_emoji

#Save as CSV File

collection_emoji.to_csv("emojis.csv")
#Assigning Dependent and Independent values

x=collection_emoji.values[:,:-1]
y=hollywood_emoji.values[:,-1]x
y

#Assigning Training and Test dataset

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0) x_train.shape
x_test.shape

#Principle Component Analysis(PCA) Algorithm

from sklearn import decomposition

#Component Assignment for PCA

pca=decomposition.PCA(n_components=20,whiten=True,random_state=1)

#Fitting the training set to generate principle components

pca.fit(x_train)

#Transforming Principle Components

x_train_pca=pca.transform(x_train)
x_test_pca=pca.transform(x_test) x_train_pca.shape
x_test_pca.shape

#Viewing the Principle components or eigen faces #Reshape the image

back to the original matrix size

eigen=(np.reshape(x[10],(512,512)).astype(np.float64))eigen
```

```
#Plotting images one by one as subplot

fig=plt.figure(figsize=(30,30))
for i in range(10): ax=fig.add_subplot(2,5,i+1,xticks=[],yticks=[])
    ax.imshow(pca.components_[i].reshape(eigen.shape),cmap=plt.cm.bone)


#Support Vector Machines Algorithm

from sklearn import svm




clf=svm.SVC(C=2,gamma=0.006,kernel='rbf')
clf.fit(x_train_pca,y_train)

#IMAGE PREDICTION

y_pred=clf.predict(x_test_pca)y_pred

#Prediction Visualization

for i in(np.random.randint(0,6,6)):
    predicted_images=(np.reshape(x_test[i],(512,512)).astype(np.float64)) plt.title('predicted
    label;{0}'.format(y_pred[i])) plt.imshow(predicted_images,interpolation='nearest',cmap='gray')
    plt.show()


#Prediction Accuracy

from sklearn import metrics
accuracy=metrics.accuracy_score(y_test,y_pred)accuracy
```

Output:

Result :

Thus, we have performed Emoji Classification using Random Classification Method Successfully.

Exp-6 Building Apriori Model for Customer Analysis in Scikit learn

Aim:

To Building apriori model for customer analysis in scikit learn.

Code:

Libraries

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

%matplotlib in line


From mlxtend.frequent_patterns import apriori,association_rules
```

Importing dataset

```
data=pd.read_csv('retail_dataset.csv')

data

items= (data['0'].unique())

items

itemset=set(items)

encoded_values= []

for index, row indata.iterrows():

rowset=set(row)

labels= {}

uncommons=list(itemset-rowset)

commons=list(itemset.intersection(rowset))

forucinuncommons:

labels[uc] =0

for com in commons:

labels[com] =1

encoded_values.append(labels)

encoded_values[0]


binary_data=pd.DataFrame(encoded_values)

binary_data
```

Applying Apriori

```
frequent_items=apriori(binary_data, min_support=0.2, use_colnames=True, verbose=1)
```

```
warnings.warn(frequent_items.head(10)
```

Mining Association

```
rules=association_rules(frequent_items, metric='confidence', min_threshold=0.6)
```

Rules

Rules Output

```
foriinrange(14):
print("Rule: ",rules.antecedents[i],"-->",rules.consequents[i])
print("Support: ",rules.support[i])
print("Confidence: ",rules.confidence[i])
print("************************************************************")
```

Support Vs Confidence

```
plt.scatter(rules['support'], rules['confidence'], alpha=0.5)
plt.xlabel('support')
plt.ylabel('confidence')
plt.title('Support vs Confidence')
plt.show()
```

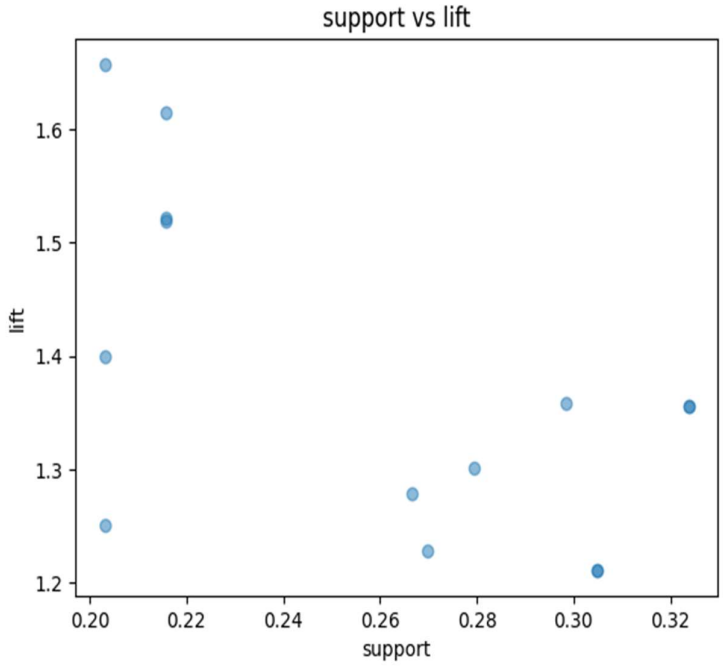
Support Vs Lift

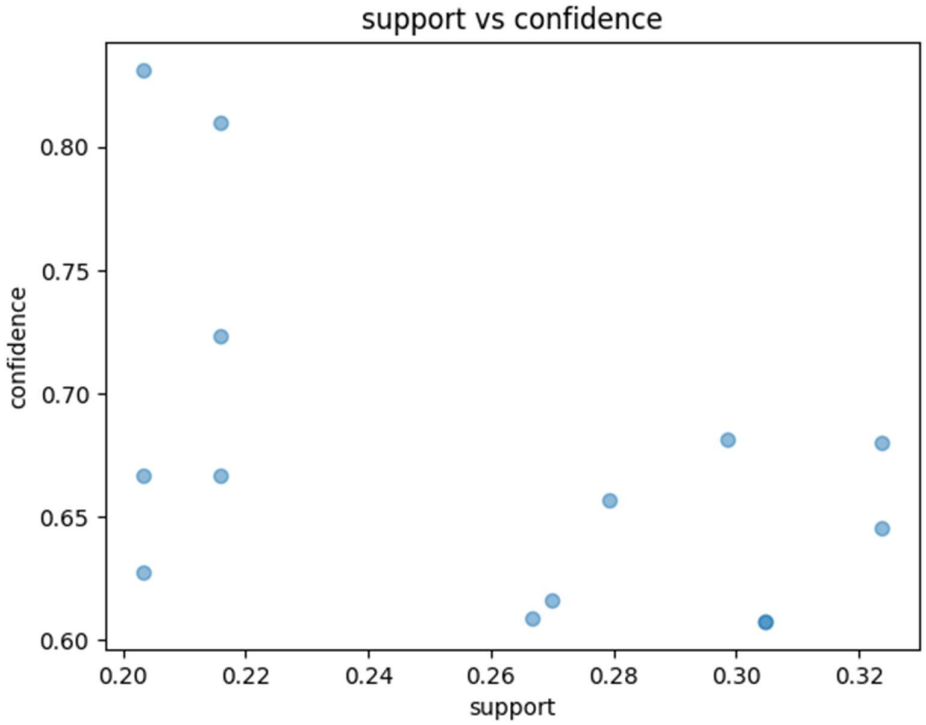
```
plt.scatter(rules['support'], rules['lift'], alpha=0.5)
plt.xlabel('support')
plt.ylabel('lift')
plt.title('Support vs Lift')
plt.show()
```

Lift Vs Confidence

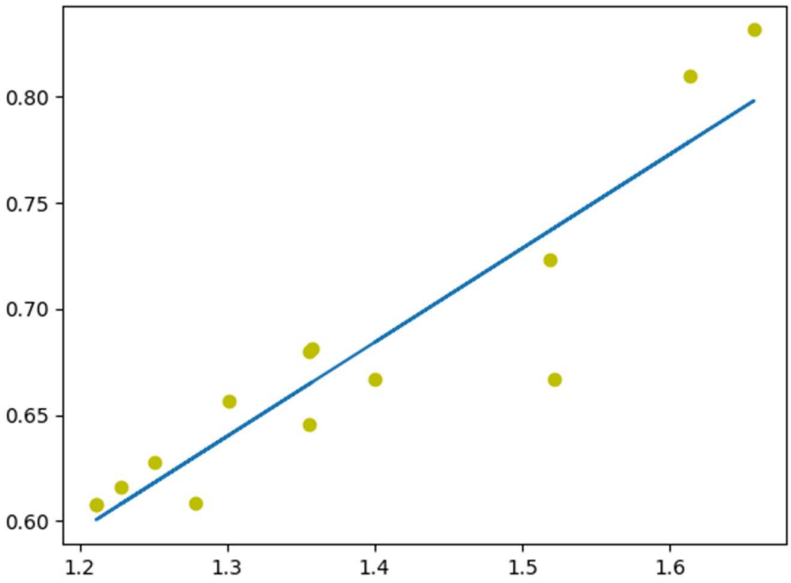
```
it=np.polyfit(rules['lift'], rules['confidence'], 1)
fit_fn=np.poly1d(fit)
plt.plot(rules['lift'], rules['confidence'], 'yo', rules['lift'],
fit_fn(rules['lift']))
```

Output:





Lift Vs Confidence



Result :

Apriori model in Scikit-learn for customer analysis, training insights via lift, confidence, and support metrics was successfully executed.

Exp-8 Implementation of Q – Learning

Aim:

To Implement Q-learning, a reinforcement learning algorithm, to train an agent in an environment by iteratively updating Q-values for state-action pairs.

Code:

Importing Libraries

```
pip install gym
```

```
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym) (1.23.5)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym) (0.0.8)
```

```
import numpy as np
import gym
```

Epsilon Greedy Policy

Choose Random Action

Choose Action of a greedy policy

```
def eps_greedy(Q, s, eps=0.1):
    if np.random.uniform(0,1) < eps:
        return np.random.randint(Q.shape[1])
    else:
        return greedy(Q, s)
```

Greedy Policy

Returning to Maximum Action-State Value

```
def greedy(Q, s):
    return np.argmax(Q[s])
```

Policy Testing

```
def run_episodes(env, Q, num_episodes=100, to_print=False):
    tot_rew = []
    state = env.reset()
```

```
    for _ in range(num_episodes):
        done = False
        game_rew = 0
        while not done:
            next_state, rew, done, _ = env.step(greedy(Q, state))
            state = next_state
            game_rew += rew
            if done:
                state = env.reset()
                tot_rew.append(game_rew)
    if to_print:
        print('Mean score: %.3f of %i games!'%(np.mean(tot_rew), num_episodes))
    return np.mean(tot_rew)
```

Q-Learning

- **Initialize Q Matrix**
- **Decay the epsilon until it reaches the threshold**
- **Select Action following Epsilon-Greedy Policy**
- **Q-Learning updates State-Action value**

Testing the Policy

```
def Q_learning(env, lr=0.01, num_episodes=10000, eps=0.3, gamma=0.95, eps_decay=0.00005):
        nA = env.action_space.n
        nS = env.observation_space.n
 Q = np.zeros((nS, nA))
 games_reward = []
 test_rewards = []
for ep in range(num_episodes):
      state = env.reset()
       done = False
       tot_rew = 0
       if eps > 0.01:
          eps -= eps_decay
       while not done:
```

```
            action = eps_greedy(Q, state, eps)
            next_state, rew, done, _ = env.step(action)
            Q[state][action] = Q[state][action] + lr*(rew + gamma*np.max(Q[next_state]) - Q[state][action])
            state = next_state
            tot_rew += rew
            if done:
                games_reward.append(tot_rew)
        if (ep % 300) == 0:
            test_rew = run_episodes(env, Q, 1000)
            print("Episode:{:5d}  Eps:{:2.4f}  Rew:{:2.4f}".format(ep, eps, test_rew))
            test_rewards.append(test_rew)
    return Q
```

Q-Learning - Taxi v3 Data¶

```
if __name__ == '__main__':
    env = gym.make('Taxi-v3')
    print("Q-Learning")
    Q_learning = Q_learning(env, lr=.1, num_episodes=5000, eps=0.4, gamma=0.95, eps_decay=0.001)
```

Output :

```
[20]
    return Q

if __name__ == '__main__':
  env = gym.make('Taxi-v3')
  print("Q-Learning")
  print()
  Q_learning = Q_learning(env, lr=.1, num_episodes=5000, eps=0.4, gamma=0.95, eps_decay=0.001)

Q-Learning

Episode:    0 eps:0.3990 rew:-262.7840
Episode:  300 eps:0.0990 rew:-221.8180
Episode:  600 eps:0.0100 rew:-199.2200
Episode:  900 eps:0.0100 rew:-143.8630
Episode: 1200 eps:0.0100 rew:-116.5560
Episode: 1500 eps:0.0100 rew:-91.8610
Episode: 1800 eps:0.0100 rew:-38.4550
Episode: 2100 eps:0.0100 rew:-19.2940
Episode: 2400 eps:0.0100 rew:-1.6330
Episode: 2700 eps:0.0100 rew:1.1230
Episode: 3000 eps:0.0100 rew:2.6980
Episode: 3300 eps:0.0100 rew:6.5490
Episode: 3600 eps:0.0100 rew:7.9410
Episode: 3900 eps:0.0100 rew:8.0210
Episode: 4200 eps:0.0100 rew:7.9130
Episode: 4500 eps:0.0100 rew:7.9310
Episode: 4800 eps:0.0100 rew:7.9810
```

Result:

The Q-learning algorithm successfully trained the agent, updating Q-values to maximize cumulative rewards over multiple episodes.

Exp-9

Implementation of SARSA Learning

Aim:

To Implement SARSA learning,a reinforcement learning algorithm, to train an agent in an environment by literatively updating SARSA values for state-action pairs.

Code:

Importing Libraries

```
pip install gym
import numpy as np
import gym
```

Epsilon Greedy Policy

Choose Random Action

Choose Action of a greedy policy

```
def eps_greedy(Q, s, eps=0.1):
    if np.random.uniform(0,1) < eps:
        return np.random.randint(Q.shape[1])
    else:
        return greedy(Q, s)
```

Greedy Policy

Returning to Maximum Action-State Value

```
def greedy(Q, s):
    return np.argmax(Q[s])
```

Policy Testing

```
def run_episodes(env, Q, num_episodes=100, to_print=False):
    tot_rew = []
    state = env.reset()
    for _ in range(num_episodes):
        done = False
        game_rew = 0
```

```
        while not done:
            next_state, rew, done, _ = env.step(greedy(Q, state))
            state = next_state
            game_rew += rew
            if done:
                state = env.reset()
                tot_rew.append(game_rew)
    if to_print:
        print('Mean score: %.3f of %i games!'%(np.mean(tot_rew), num_episodes))
    return np.mean(tot_rew)
```

SARSA

Initialize Q Matrix
Decay the epsilon until it reaches the threshold
Choose next Action
SARSA update
Testing the Policy

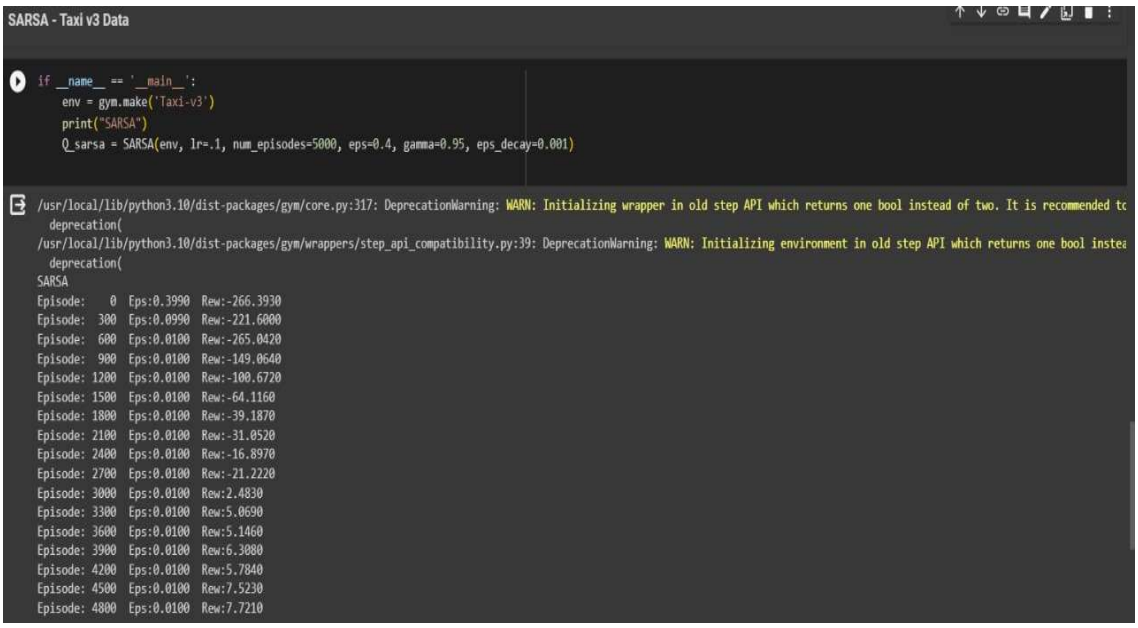
```
def SARSA(env, lr=0.01, num_episodes=10000, eps=0.3, gamma=0.95, eps_decay=0.00005):
    nA = env.action_space.n
    nS = env.observation_space.n
    Q = np.zeros((nS, nA))
    games_reward = []
    test_rewards = []
    for ep in range(num_episodes):
        state = env.reset()
        done = False
        tot_rew = 0
        if eps > 0.01:
            eps -= eps_decay
        action = eps_greedy(Q, state, eps)
        while not done:
            next_state, rew, done, _ = env.step(action)
            next_action = eps_greedy(Q, next_state, eps)
```

```
            Q[state][action] = Q[state][action] + lr*(rew + gamma*Q[next_state][next_action] - Q[state][action])
            state = next_state
            action = next_action
            tot_rew += rew
            if done:
                games_reward.append(tot_rew)
        if (ep % 300) == 0:
            test_rew = run_episodes(env, Q, 1000)
            print("Episode:{:5d}  Eps:{:2.4f}  Rew:{:2.4f}".format(ep, eps, test_rew))
            test_rewards.append(test_rew)
    return Q
```

SARSA - Taxi v3 Data

```
if __name__ == '__main__':
    env = gym.make('Taxi-v3')
    print("SARSA")
    Q_sarsa = SARSA(env, lr=.1, num_episodes=5000, eps=0.4, gamma=0.95, eps_decay=0.001)
```

Output:



Result:

The SARSA learning algorithm successfully trained the agent, updating SARSA values to maximize cumulative rewards over multiple episodes.

EXP – 10 – Data Acquisition from various Sensors in Tiny ML Kit

(Arduino Nano 33 BLE Sense)

AIM:

To acquire data form various Sensors in Tiny ML Kit

SOFTWARE REQUIRED:

1) Edge impulse

HARDWARE REQUIRED:

1. Arduino Nano 33 BLE Sense

STPES TO FOLLOW:

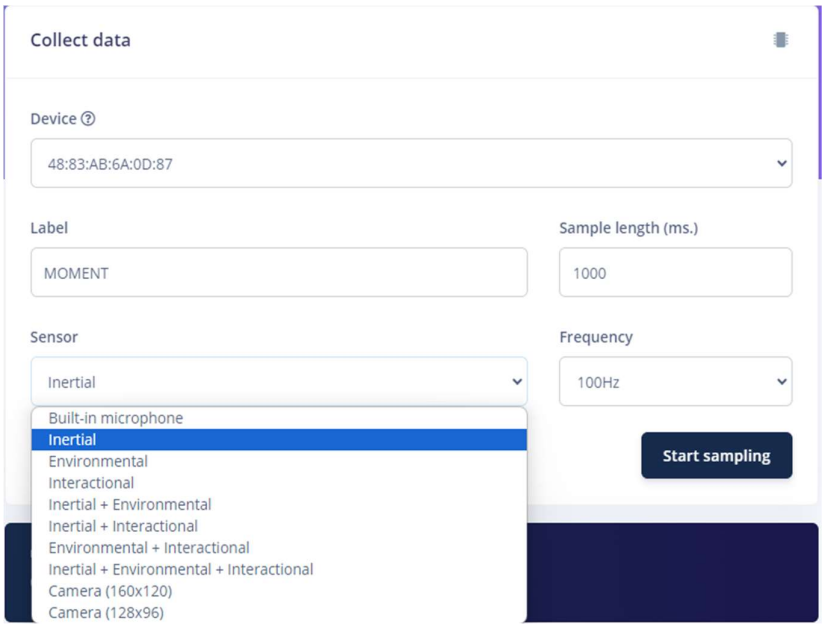
Step 1 – Download required packages and dependencies for your specified kit (NANO-33 BLE)

Step 2 - Connect your tiny ML (NANO-33 BLE) kit to edge impulse via USB cable for data Acquisition

Step 3 – After connecting your kit choose your sensor, set frequency, set sample length and start to sample

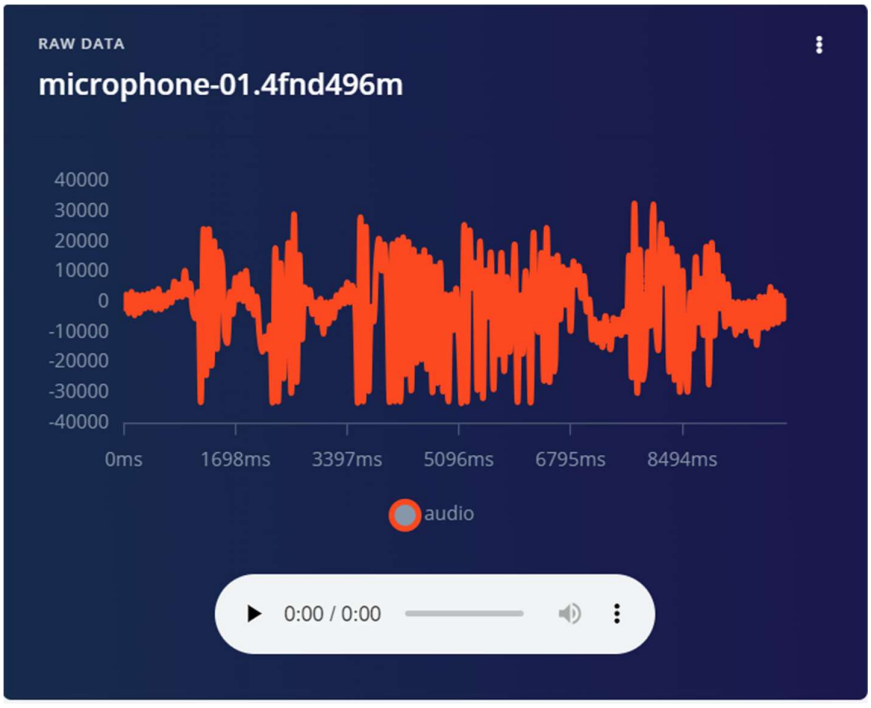
Step 4 - Your sample is collected and being stored to training set or testing set

CHOOSE THE SENSOR FOR DATA AQUSITION:



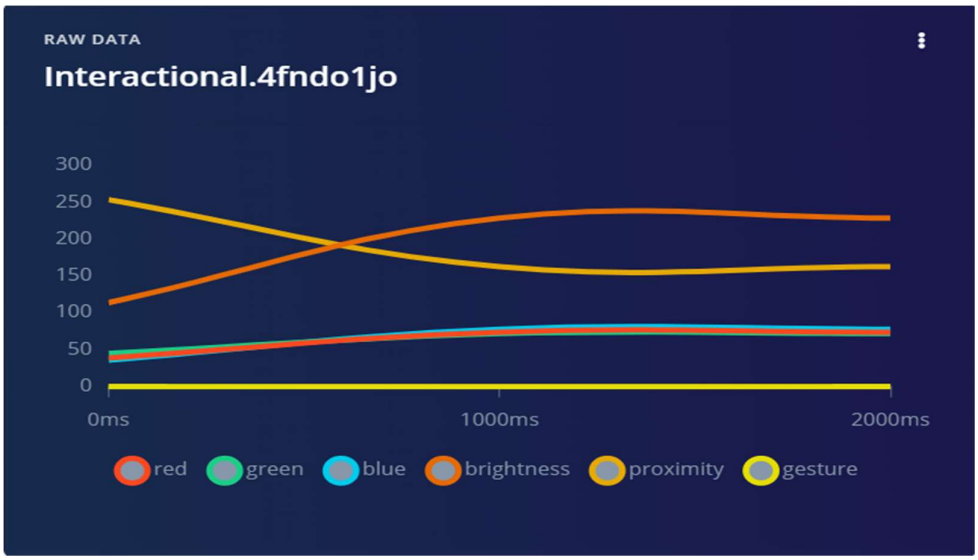
OUTPUT:

MICRO-PHONE:

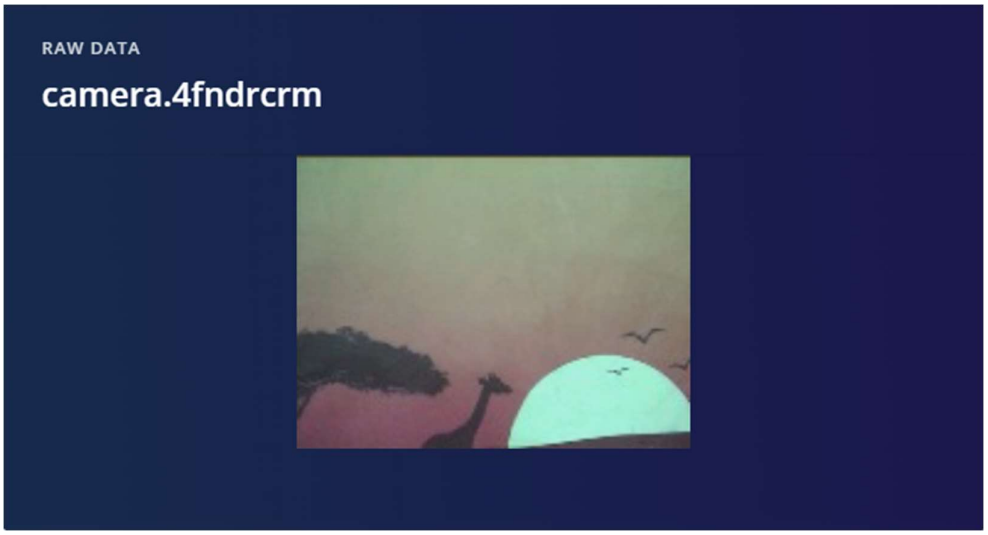


INERTIAL:

INTERACTIONAL:



CAMERA:



OUTPUT:

Data Acquisition has been done through edge impulse using tiny ml kit.

Exp-10 BUILD AN AUDIO CLASSIFICATION MODEL USING TINYML KIT

AIM:

The aim of this project is to develop an audio classification model using the TinyML Kit and deploy it on an Arduino board for real-time classification of audio samples into predefined categories.

SOFTWARE REQUIREMENT:

- Edge Impulse

HARDWARE REQUIREMENT:

- Arduino Nano 33 BLE Sense

STEPS/PROCEDURE:

Step 1 – Download required packages and dependencies for your specified kit (NANO-33 BLE)

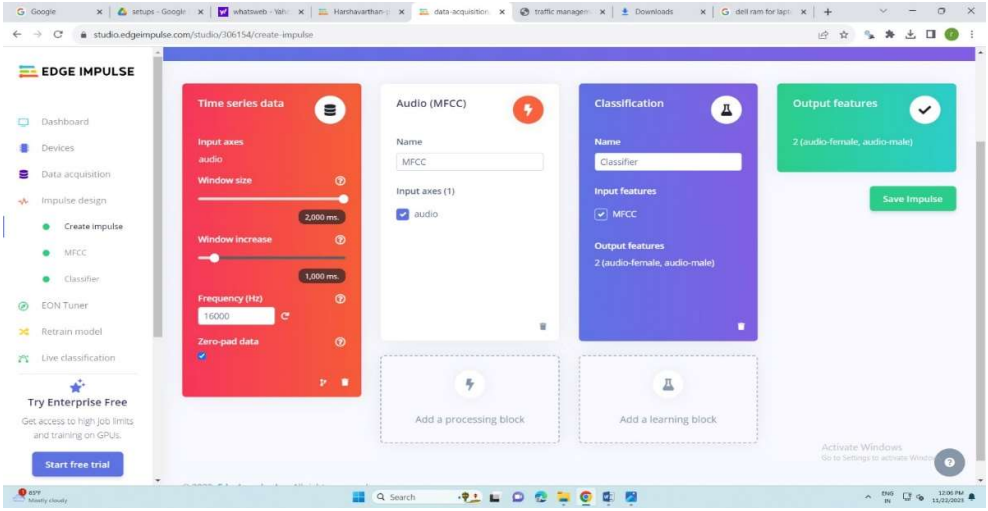
Step 2 - Connect your tiny ML (NANO-33 BLE) kit to edge impulse via USB cable for data Acquisition

Step 3 – After connecting your kit choose your sensor, set frequency, set sample length and start to sample

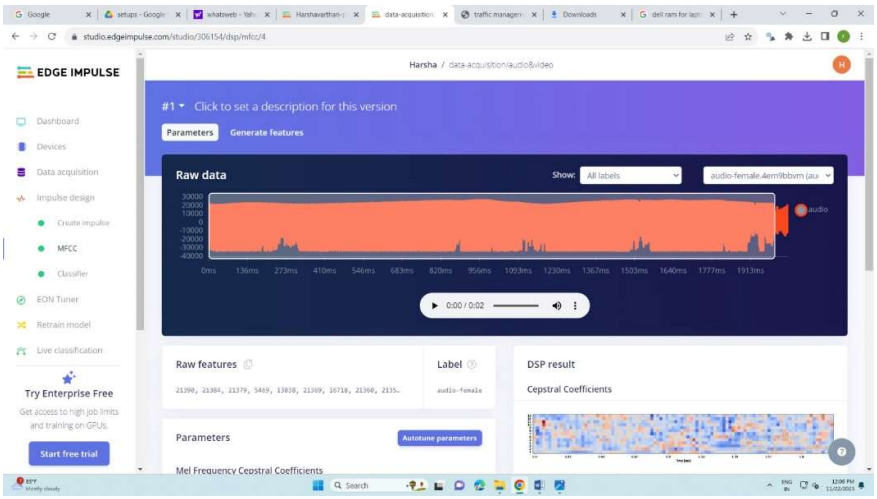
Step 4 - Your sample is collected and being stored to training set or testing set

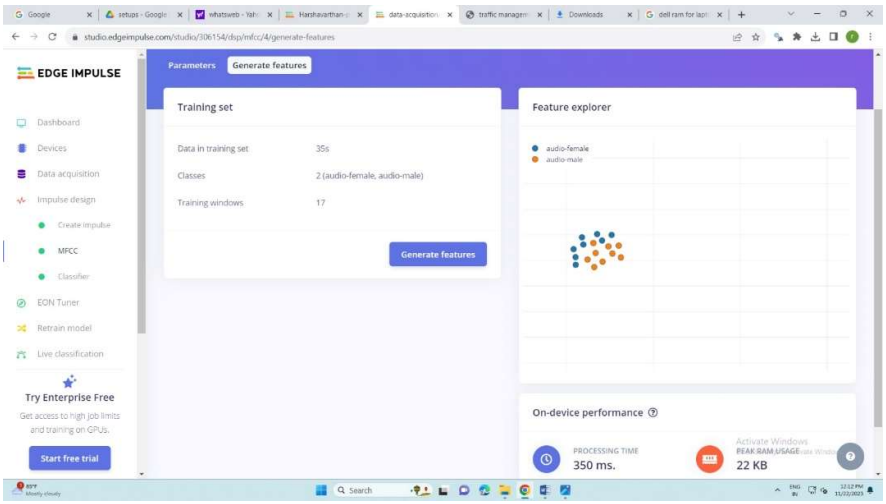
Step 5 – Create Impulse Design :

- Impulse Design

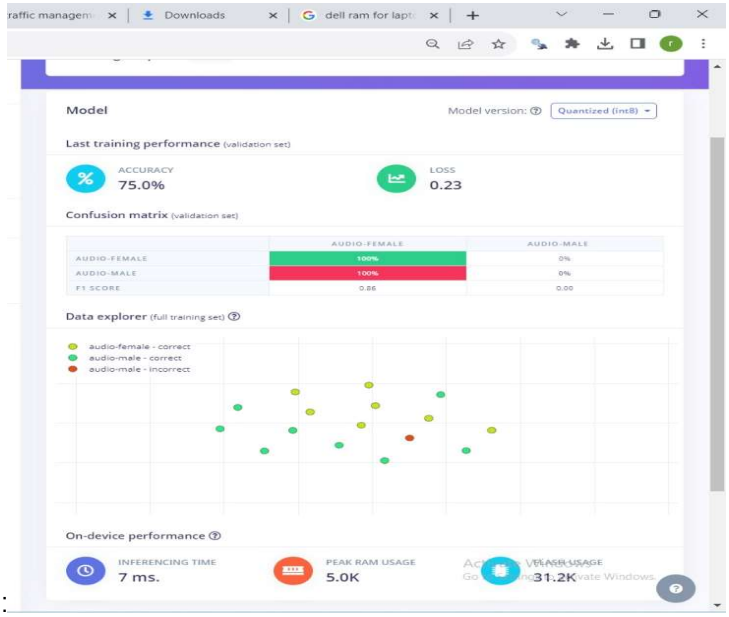


- MFCC



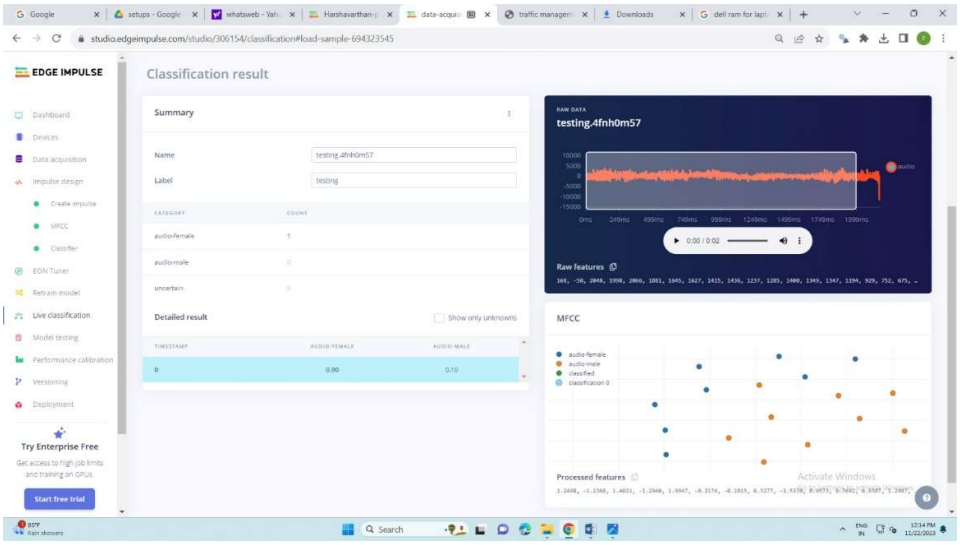


Classifier



OUTPUT:

LIVE CLASSIFICATION



DEPLOYMENT

Set up the Arduino code to load and run the TensorFlow Lite model using the TensorFlow Lite for Microcontrollers (TensorFlow Lite Micro) library.

Optimize if Necessary:

If the model's performance is not satisfactory, consider optimizing the model architecture, quantizing weights, or adjusting hyperparameters to meet the constraints of the Arduino board.

RESULTS:

Analyze the results of model testing on the Arduino, considering factors like classification accuracy, inference speed, and resource usage.