**PART A**

(PART A: TO BE REFFERED BY STUDENTS)

**EXPERIMENT NO. 6**

**A.1 AIM: -** To Implement Multi-Category Classification Using Binary Linear Classifiers

**A.2 Prerequisite**

* Different programming language (Python or Java), Understanding of Machine Learning Algorithms, Machine Learning Algorithms

**A.3 Outcome**

After successful completion of this experiment students will be able to implement Multi-Category Classification Using Binary Linear Classifiers

**A.4 Theory**

**Definition of Classification**

In machine learning, Classification, as the name suggests, classifies data into different parts/classes/groups. It is used to predict from which dataset the input data belongs to.

For example, if we are taking a dataset of scores of a cricketer in the past few matches, along with average, strike rate, not outs etc, we can classify him as “in form” or “out of form”.

Classification is the process of assigning new input variables (X) to the class they most likely belong to, based on a classification model, as constructed from previously labeled training data.

Data with labels is used to train a classifier such that it can perform well on data without labels (not yet labeled). This process of continuous classification, of previously known classes, trains a machine. If the classes are discrete, it can be difficult to perform classification tasks

**Types of Classification**

There are two types of classifications;

1. Binary classification
2. Multi-class classification

* **Binary Classification**

It is a process or task of classification, in which a given data is being classified into two classes. It’s basically a kind of prediction about which of two groups the thing belongs to.

Let us suppose, two emails are sent to you, one is sent by an insurance company that keeps sending their ads, and the other is from your bank regarding your credit card bill. The email service provider will classify the two emails, the first one will be sent to the spam folder and the second one will be kept in the primary one.

This process is known as binary classification, as there are two discrete classes, one is spam and the other is primary. So, this is a problem of binary classification.

Binary classification uses some algorithms to do the task, some of the most common algorithms used by binary classification are .

1. Logistic Regression
2. k-Nearest Neighbors
3. Decision Trees
4. Support Vector Machine
5. Naive Bayes

* **Multiclass Classification**

Multi-class classification is the task of classifying elements into different classes. Unlike binary, it doesn’t restrict itself to any number of classes.

Examples of multi-class classification are

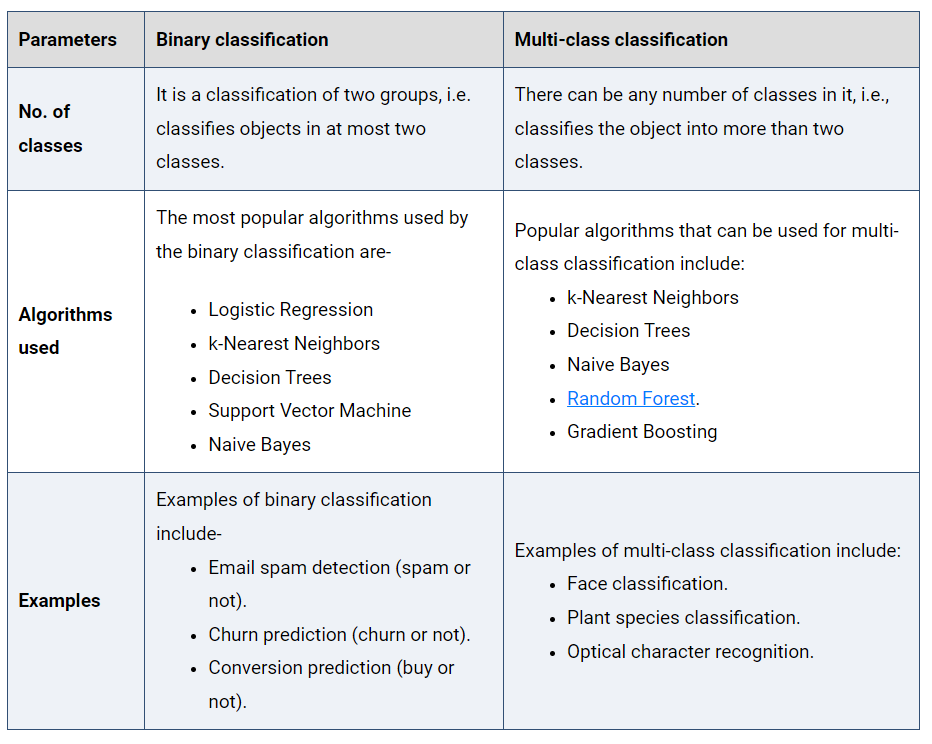
1. classification of news in different categories,
2. classifying books according to the subject,
3. classifying students according to their streams etc.

In these, there are different classes for the response variable to be classified in and thus according to the name, it is a Multi-class classification

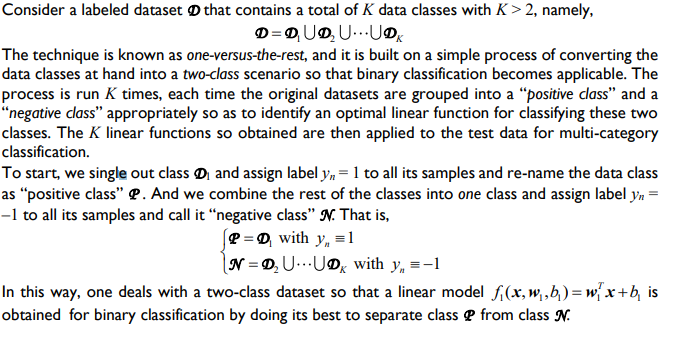
Can a classification possess both binary or multi-class?

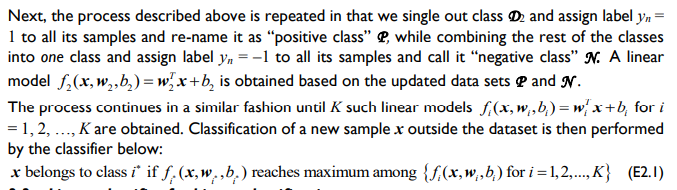
Let us suppose we have to do sentiment analysis of a person, if the classes are just “positive” and “negative”, then it will be a problem of binary class. But if the classes are “sadness”, happiness”, “disgusting”, “depressed”, then it will be called a problem of Multi-class classification.

* **Binary vs Multiclass Classification**

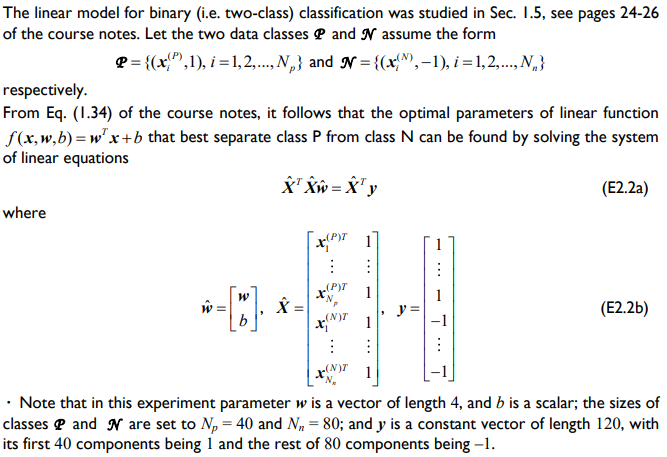


* **The idea of multi-category classification using linear binary classifiers**





* **Linear classifier for binary classification**



**A5. Task**

**Dataset: Use** Fisher’s dataset

In this experiment, we have to investigate a technique for multi-category classification based on binary classifications. The technique is then applied to Fisher’s 3-class datasets of Iris plants to

demonstrate its effectiveness. The dataset of Iris plants to be used in this experiment was created

and published in 1936 by R. A. Fisher [R1]. Fisher's paper is a classic in the field and is referenced

frequently to this day, as a matter of fact the dataset is arguably the best-known in the pattern

recognition literature [R2]. The dataset includes features of 150 Iris plants of 3 species known as

Setosa, Versicolor, and Virginica, where each sample Iris is represented by a 4-dimensional vector

in terms of lengths and widths of the sepal and petal of the flower.

In this experiment, the above data set was divided into two sets, one for training and the other

for testing. The training data set includes 120 samples where there are 40 samples for each

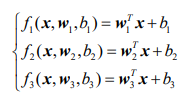
species. The test data set includes 30 samples and there are 10 samples for each species.

Procedure:

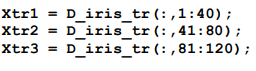
1. Download the dataset.
2. Perform necessary EDA on the dataset

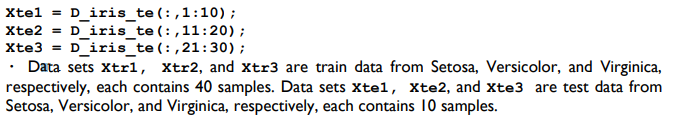
Steps to be considered:

1. this lab experiment the number of classes is K = 3.
2. Consequently, 3 binary classifications are required to produce linear models

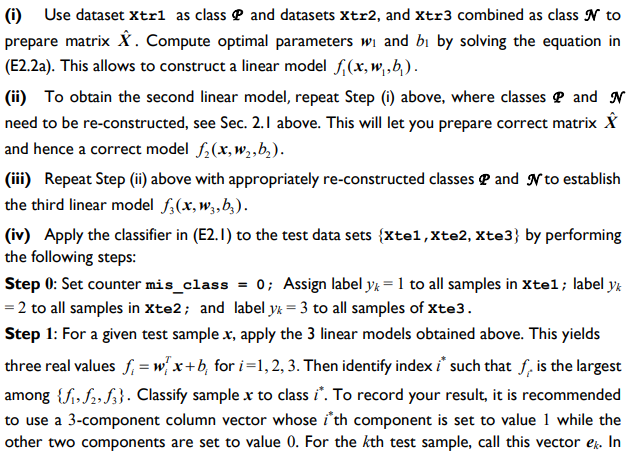


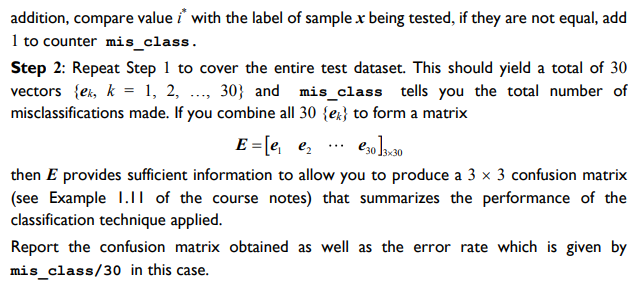
1. Prepare datasets as:





1. Perform the following steps





PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.: C027 | Name: Vishesh Giyanani |
| Class: B | Batch: EB1 |
| Date of Experiment: | Date of Submission |
| Grade : |  |

**B.1 Documentation written by student:**

import pandas as pd

import numpy as np

df = pd.read\_csv('/content/drive/MyDrive/Datasets/Iris.csv')

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

df.head()

def normalize(df):

    for column in df.columns:

        if column != "Species":

            df[column] = (df[column] - df[column].min()) / (df[column].max() - df[column].min())

    return df

normalized\_df = normalize(df.copy())

normalized\_df.drop(columns=['Id'],inplace=True)

normalized\_df['Species'].unique()

df\_setosa = normalized\_df.copy()

df\_setosa['Species'] = normalized\_df['Species'].apply(lambda x: 1 if x == 'Iris-setosa' else 0)

df\_setosa.head()

df\_setosa['Species'].value\_counts()

df\_versicolor = normalized\_df.copy()

df\_versicolor['Species'] = normalized\_df['Species'].apply(lambda x: 1 if x == 'Iris-versicolor' else 0)

df\_versicolor.head()

df\_versicolor['Species'].value\_counts()

df\_virginica = normalized\_df.copy()

df\_virginica['Species'] = normalized\_df['Species'].apply(lambda x: 1 if x == 'Iris-virginica' else 0)

df\_virginica.head()

df\_virginica['Species'].value\_counts()

lr\_setosa = LogisticRegression()

lr\_versicolor = LogisticRegression()

lr\_virginica = LogisticRegression()

X\_setosa\_train, X\_setosa\_test, Y\_setosa\_train, Y\_setosa\_test = train\_test\_split(df\_setosa.drop(columns=['Species']),df\_setosa['Species'],test\_size=0.2,shuffle=True)

X\_versicolor\_train, X\_versicolor\_test, Y\_versicolor\_train, Y\_versicolor\_test = train\_test\_split(df\_versicolor.drop(columns=['Species']),df\_versicolor['Species'],test\_size=0.2,shuffle=True)

X\_virginica\_train, X\_virginica\_test, Y\_virginica\_train, Y\_virginica\_test = train\_test\_split(df\_virginica.drop(columns=['Species']),df\_virginica['Species'],test\_size=0.2,shuffle=True)

lr\_setosa.fit(

    X\_setosa\_train,

    Y\_setosa\_train)

lr\_versicolor.fit(

    X\_versicolor\_train,

    Y\_versicolor\_train

)

lr\_virginica.fit(

    X\_virginica\_train,

    Y\_virginica\_train

)

y\_pred\_setosa = lr\_setosa.predict(X\_setosa\_test)

print("Accuracy:", metrics.accuracy\_score(y\_pred\_setosa, Y\_setosa\_test))

print(metrics.classification\_report(Y\_setosa\_test, y\_pred\_setosa))

print(metrics.confusion\_matrix(Y\_setosa\_test, y\_pred\_setosa))

y\_pred\_versicolor = lr\_versicolor.predict(X\_versicolor\_test)

print("Accuracy:", metrics.accuracy\_score(y\_pred\_versicolor, Y\_versicolor\_test))

print(metrics.classification\_report(Y\_versicolor\_test, y\_pred\_versicolor))

print(metrics.confusion\_matrix(Y\_versicolor\_test, y\_pred\_versicolor))

y\_pred\_virginica = lr\_virginica.predict(X\_virginica\_test)

print("Accuracy:", metrics.accuracy\_score(y\_pred\_virginica, Y\_virginica\_test))

print(metrics.classification\_report(Y\_virginica\_test, y\_pred\_virginica))

print(metrics.confusion\_matrix(Y\_virginica\_test, y\_pred\_virginica))

y\_pred\_final = np.append(y\_pred\_setosa,y\_pred\_versicolor)

y\_pred\_final = np.append(y\_pred\_final,y\_pred\_virginica)

y\_pred\_final

y\_test\_final = np.append(Y\_setosa\_test,Y\_versicolor\_test)

y\_test\_final = np.append(y\_test\_final, Y\_virginica\_test)

y\_test\_final

print("Accuracy:", metrics.accuracy\_score(y\_pred\_final, y\_test\_final))

print(metrics.classification\_report(y\_test\_final, y\_pred\_final))

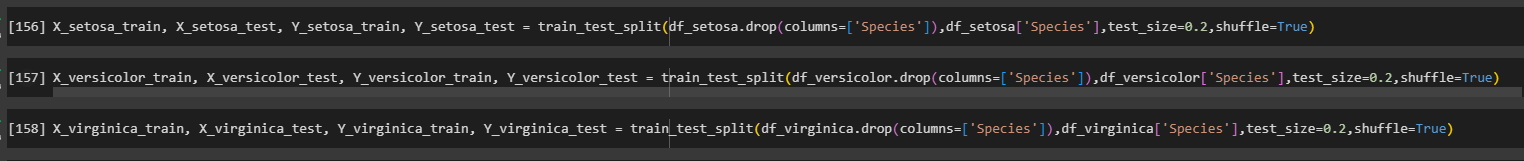
print(metrics.confusion\_matrix(y\_test\_final, y\_pred\_final))

**Output:**

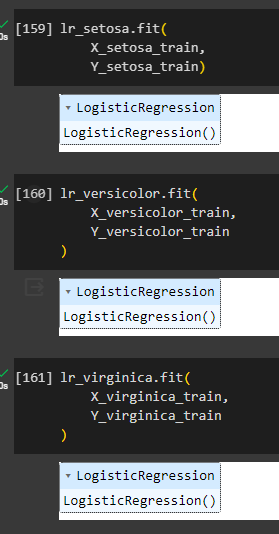
**Main Dataset divided into 3 Datasets:**

****

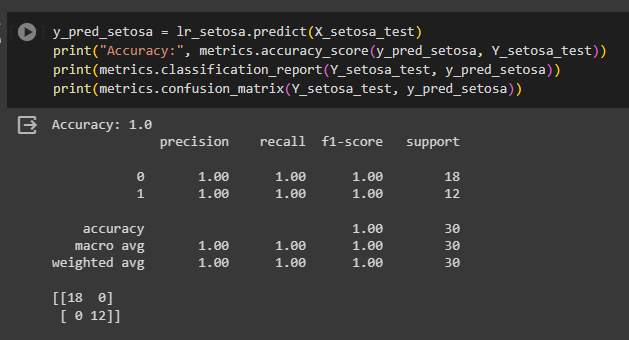
**Test\_train\_split:**

****

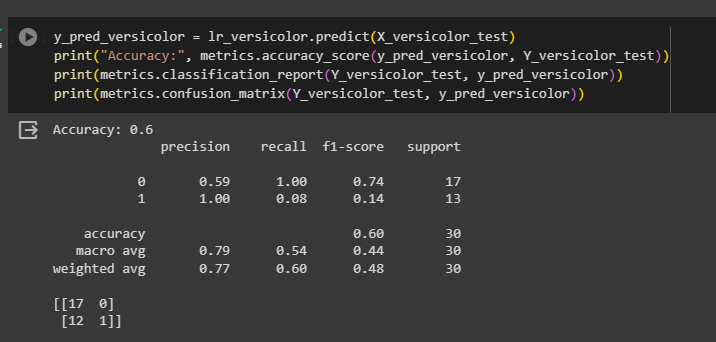
**Model Fitting:**

****

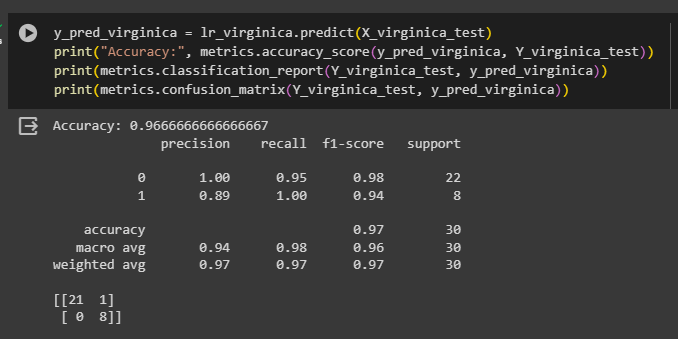
**Setosa Accuracy:**

****

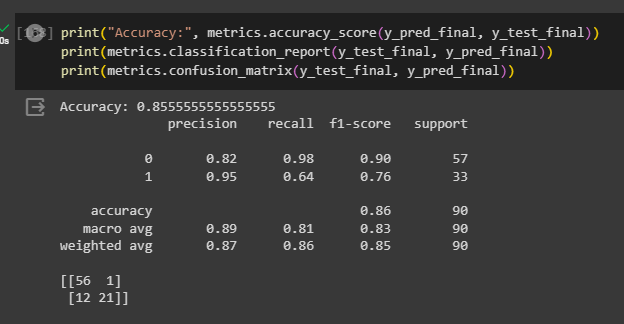
**Versicolor Accuracy:**

****

**Virginica Accuracy:**

****

**Concat Metrics:**

****

**B.2 Observations and learning:**

* **It is a good way to implement multi-class classification without using neural networks**
* **Accuracy is good enough for 3 individual models for the same dataset**

**B.3 Conclusion:**

* Learnt and practiced multi-class classification with individual binary Classification models for each class.