(DIGITAL ASSIGNMENT - 1)

CSE4020(MACHINE LEARNING)THEORY:C1+TC1

JANUARY 28, 2022

ANIRUDH VADERA
20BCE2940

ANIRUDH VADERA
(DIGITAL ASSIGNMENT - 1)

QUESTION:

Prepare a table with the list of learning rules used in neural network.

NAME OF THE LEARNING RULE	FORMULA	SUPERVISED /UNSUPERVISED /REINFORCEMENT LEARNING	CHARACTERISTICS
HEBBIAN LEARNING RULE	Formula in Artificial Neural Network $W_{mn} = x_m * x_n$ Here, **W_{mn}** = Increment by which weight of connection increases. **X_m** = The input value from pre-synaptic neuron. **X_n** = The output of pre-synaptic neuron.	**UNSUPERVISED**	• We can use it to identify how to improve the weights of nodes of a network. • **If two neighbour neurons activated and deactivated at the same time. Then the weight connecting these neurons should increase.** • For neurons operating in the opposite phase, the weight between them should decrease. If there is no signal correlation, the weight should not change. • **At the start, values of all weights are set to zero.**
PERCEPTRON LEARNING RULE	Formula in Artificial Neural Network $\sum_m \sum_n (E_{mn} - O_{mn})$ ^2	**SUPERVISED**	• **Assigns a random value to each weight.** • Calculate the output value on the basis of a set of records for which we can know the expected output value. • **The network then compares the calculated output value with the expected value. Next calculates an error function ∈, which can be the sum of squares of the errors occurring for each individual in the learning sample.**
DELTA LEARNING RULE	Formula in Artificial Neural Network $\Delta w = \eta(t - y)x_i$ **(t-y)** = The difference between the desired/target output and the actual output	**SUPERVISED**	• **This rule states that the modification in sympatric weight of a node is equal to the multiplication of error and the input.** • **The aim of applying the delta rule is to reduce the**

	X_i = The input value from pre-synaptic neuron. **n**= the positive and constant learning rate **Δw** = Weight change for i^{th} pattern.		**difference between the actual and expected output that is the error.** • For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference
CORRELATION LEARNING RULE	Formula in Artificial Neural Network $\Delta w_{ij} = \eta x_i d_j$ The learning signal r is simply equal to desired output di **r = d_i**	**SUPERVISED**	• **It assumes that weights between responding neurons should be more positive, and weights between neurons with opposite reaction should be more negative.** • **Similar to Hebbian learning rule** • This training algorithm usually starts with the initialization of weights to zero.
OUT STAR LEARNING RULE	Formula in Artificial Neural Network **If node j wins the competition:** $\Delta W_j = \alpha(d - w_j)$ Here **d** is the desired neuron output and α is the learning rate. **If node j loses the competition:** $\Delta W_j = 0$	**SUPERVISED**	• **We assume that nodes or neurons in a network arranged in a layer. Here the weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights.** The out start rule produces the desired response **t** for the layer of n nodes. • This rule is applied over the neurons arranged in a layer. It is specially designed to produce a desired output **d** of the layer of **p** neurons.

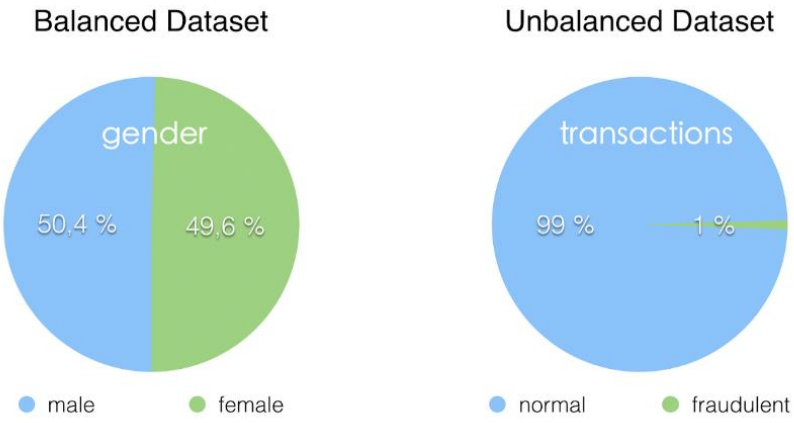
Learning rule or **Learning process** is a method or a mathematical logic. It improves the **Artificial Neural Network's** performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment.

Applying learning rule is an iterative process. It helps a neural network to learn from the existing conditions and improve its performance.

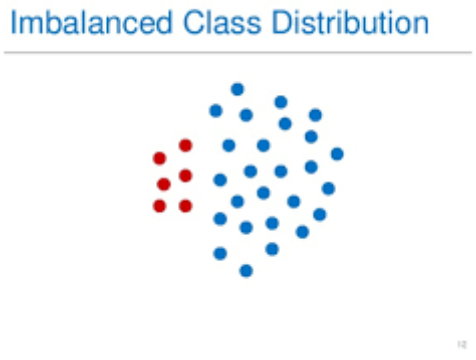
QUESTION:

What is unbalanced dataset. List out the techniques to convert it to balanced data set. Also, implement it using of those techniques to classify using Naïve Bayes classification and show the difference in performance metrics of precision, recall and F1 score.

Unbalanced dataset:



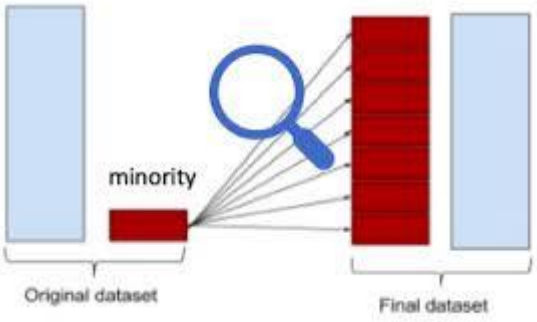
In simple terms, an unbalanced dataset is one in which the target variable has more observations in one specific class than the others. It typically refers to a problem with classification problems where the classes are not represented equally.



Techniques to convert Unbalanced dataset to a balanced dataset:

Experimentally Verifying our basis:

Over-sampling (Up Sampling): This technique is used to modify the unequal data classes to create balanced datasets. When the quantity of data is insufficient, the oversampling method tries to balance by incrementing the size of rare samples.



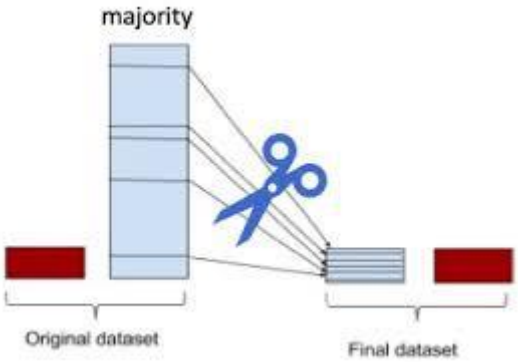
Advantages

- No loss of information
- Mitigate overfitting caused by oversampling.

Disadvantages

- Overfitting

Under-sampling (Down Sampling): Unlike oversampling, this technique balances the imbalance dataset by reducing the size of the class which is in abundance. There are various methods for classification problems such as cluster centroids and Tomek links. The cluster centroid methods replace the cluster of samples by the cluster centroid of a K-means algorithm and the Tomek link method removes unwanted overlap between classes until all minimally distanced nearest neighbours are of the same class.



Advantages

- Run-time can be improved by decreasing the amount of training dataset.
- Helps in solving the memory problems

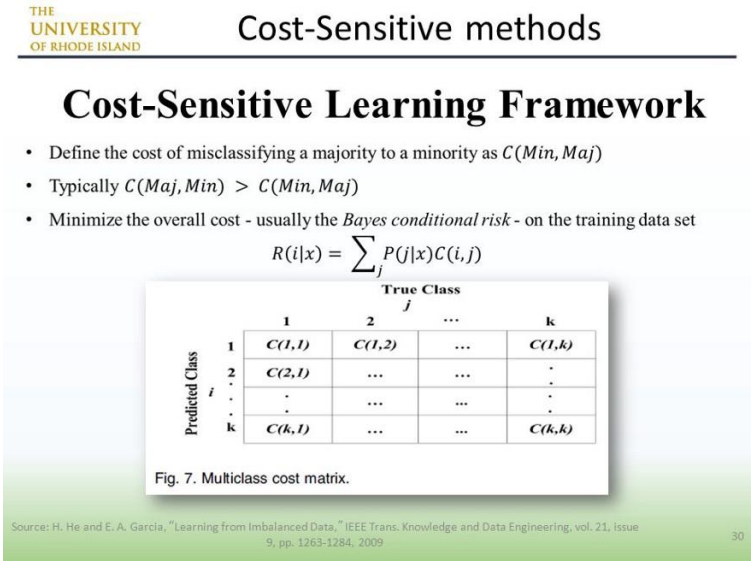
Disadvantages

- Losing some critical information

Feature selection: On both the positive and negative classes, we generate one-sided metrics such as correlation coefficient (CC) and odds ratios (OR) or two-sided metrics such as information gain (IG) and chi-square (CHI) to address the imbalance problem. We next determine the significant features from each class based on the scores and add them together to get the final collection of features. The data is then used to classify the issue.

Identifying these characteristics will assist us in generating a clear decision boundary for each class. This aids the models in appropriately classifying the data. This serves as intelligent subsampling and may aid in the reduction of the imbalance problem.

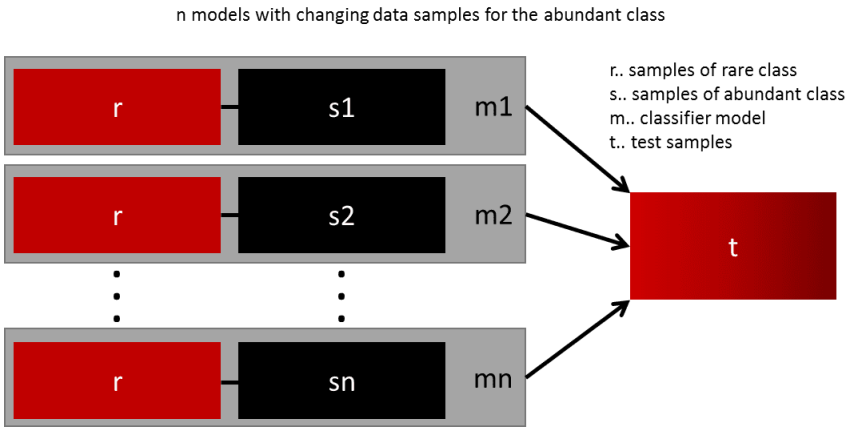
Cost-Sensitive Learning Technique: By minimising the total cost, Cost-Sensitive Learning (CSL) considers the costs of misclassification. This technique's primary purpose is to achieve high accuracy in classifying examples into a set of known classes. It is one of the most important components of machine learning algorithms, as well as real-world data mining applications.



Advantages:

This technique avoids pre-selection of parameters and auto-adjust the decision hyperplane

Ensemble Learning Techniques: Another way for dealing with imbalanced data sets is the ensemble-based method, which combines the results or performance of numerous classifiers to improve the performance of a single classifier. By combining different classifiers, this technique improves the generalisation power of individual classifiers. It mostly integrates the outputs of a number of different base learners. Bagging, Boosting, and other approaches to ensemble learning are examples.



Advantages:

- This is a more stable model
- The prediction is better

Smote: Synthetic Minority Oversampling Technique or **SMOTE** is another technique to oversample the minority class. Simply adding duplicate records of minority class often don't add any new information to the model. In SMOTE new instances are synthesized from the existing data. If we explain it in simple words, SMOTE looks into minority class instances and use *k* nearest neighbor to select a random nearest neighbor, and a synthetic instance is created randomly in feature space.

Code:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.datasets import load_breast_cancer

from collections import Counter

from matplotlib import pyplot as plt

from sklearn.metrics import confusion_matrix


data = load_breast_cancer()

df = pd.DataFrame(data.data)
```

```
print("The dataframe we will be using is breast cancer dataset : ")
print(df)

df["Target"]=data.target
grp = df.groupby("Target")
df0 = grp.get_group(0)
df1 = grp.get_group(1)
df0 = df0.sample(frac=1)[:195]
df1 = df1.sample(frac=1)[:5]

# Before Balancing our dataset
unbal = pd.concat([df0,df1]).reset_index()
unbal = unbal.drop(labels="index",axis=1)
unbal["Target"].value_counts().plot(kind="bar",title="Counter(Targer)")
x= unbal.drop(labels="Target",axis=1)
y= unbal["Target"]


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.35, random_state= 1009)
# Using Naïve Bayes
gnb = GaussianNB()
y_pred = gnb.fit(x_train, y_train).predict(x_test)

print("Before Balancing our dataset : ")
print(Counter(y_train))

print('Unbalanced Precision: %.3f' % precision_score(y_test, y_pred))
print('Unbalanced Recall: %.3f' % recall_score(y_test, y_pred))
```

print('Unbalanced F1 Score: %.3f' % f1_score(y_test, y_pred))

conf_mat = confusion_matrix(y_true=y_test, y_pred=y_pred)

print('Confusion matrix:\n', conf_mat)

labels = ['Class 0', 'Class 1']

fig = plt.figure()

ax = fig.add_subplot(111)

cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)

fig.colorbar(cax)

ax.set_xticklabels([''] + labels)

ax.set_yticklabels([''] + labels)

plt.xlabel('Predicted')

plt.ylabel('Expected')

plt.show()

Balancing our Unbalanced dataset:

bal =
pd.concat([df0,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1,df1]).reset_index()

bal = bal.drop(labels="index",axis=1)

bal["Target"].value_counts().plot(kind="bar",title="Count(Target)")

X= bal.drop(labels="Target",axis=1)

Y= bal["Target"]

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size= 0.35, random_state= 1009)

 # Using Naïve Bayes

gnb = GaussianNB()

Y_pred = gnb.fit(X_train, Y_train).predict(X_test)

```
labels = ['Class 0', 'Class 1']
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)
ax.set_xticklabels(['']  + labels)
ax.set_yticklabels(['']  + labels)
plt.xlabel('Predicted')
plt.ylabel('Expected')
plt.show()
```

```
print("After Balancing our dataset : ")
print(Counter(Y_train))
print('balanced Precision: %.3f' % precision_score(Y_test, Y_pred))
print('balanced Recall: %.3f' % recall_score(Y_test, Y_pred))
print('balanced F1 Score: %.3f' % f1_score(Y_test, Y_pred))
conf_mat = confusion_matrix(y_true=Y_test, y_pred=Y_pred)
print('Confusion matrix:\n', conf_mat)
```

Output:

Dataset Used:

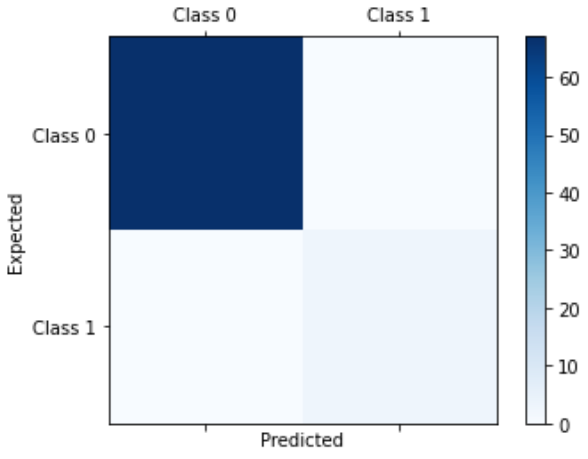
```
The dataframe we will be using is breast cancer dataset :
        0      1       2       3   ...      26      27      28       29
0    17.99  10.38  122.80  1001.0  ...  0.7119  0.2654  0.4601  0.11890
1    20.57  17.77  132.90  1326.0  ...  0.2416  0.1860  0.2750  0.08902
2    19.69  21.25  130.00  1203.0  ...  0.4504  0.2430  0.3613  0.08758
3    11.42  20.38   77.58   386.1  ...  0.6869  0.2575  0.6638  0.17300
4    20.29  14.34  135.10  1297.0  ...  0.4000  0.1625  0.2364  0.07678
..     ...    ...     ...     ...  ...     ...     ...     ...      ...
564  21.56  22.39  142.00  1479.0  ...  0.4107  0.2216  0.2060  0.07115
565  20.13  28.25  131.20  1261.0  ...  0.3215  0.1628  0.2572  0.06637
566  16.60  28.08  108.30   858.1  ...  0.3403  0.1418  0.2218  0.07820
567  20.60  29.33  140.10  1265.0  ...  0.9387  0.2650  0.4087  0.12400
568   7.76  24.54   47.92   181.0  ...  0.0000  0.0000  0.2871  0.07039
```

```
'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv',
'data_module': 'sklearn.datasets.data'}
```

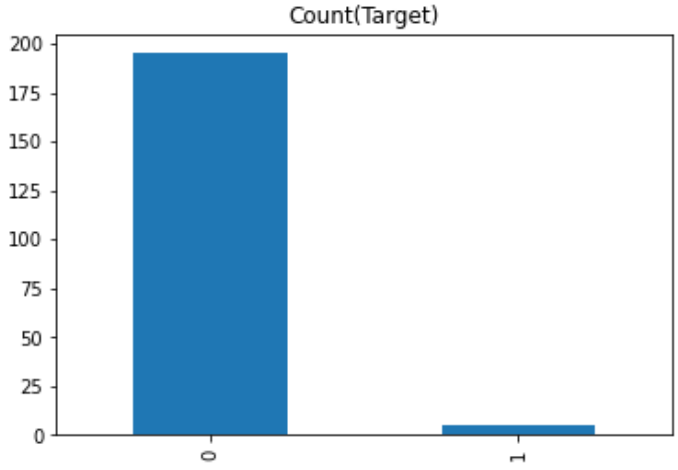
Before Balancing the dataset:

```
Before Balancing our dataset :
Counter({0: 128, 1: 2})
Unbalanced Precision: 0.500
Unbalanced Recall: 0.333
Unbalanced F1 Score: 0.400
Confusion matrix:
 [[66  1]
 [ 2  1]]
```

Confusion Matrix:



The BarGraph is as following:

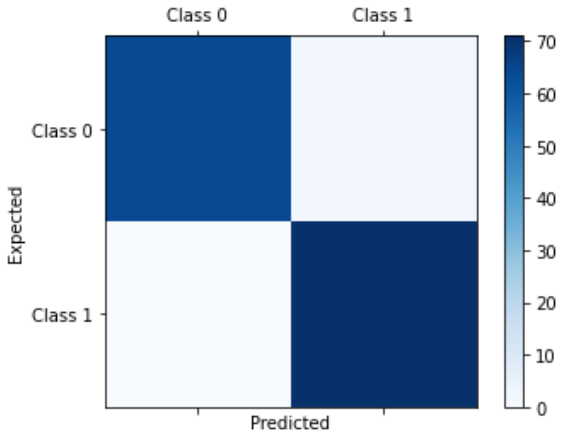


- **We have used Naïve Bayes Classification as specified**
- **We can see the dataset is clearly unbalanced as the Class: 0 has a count of around 128 whereas the class : 1 has a count of 2.**
- **The dataset is more aligned to Class: 0 than to Class: 1**
- **The F1_Score is: 40% which is not that good**

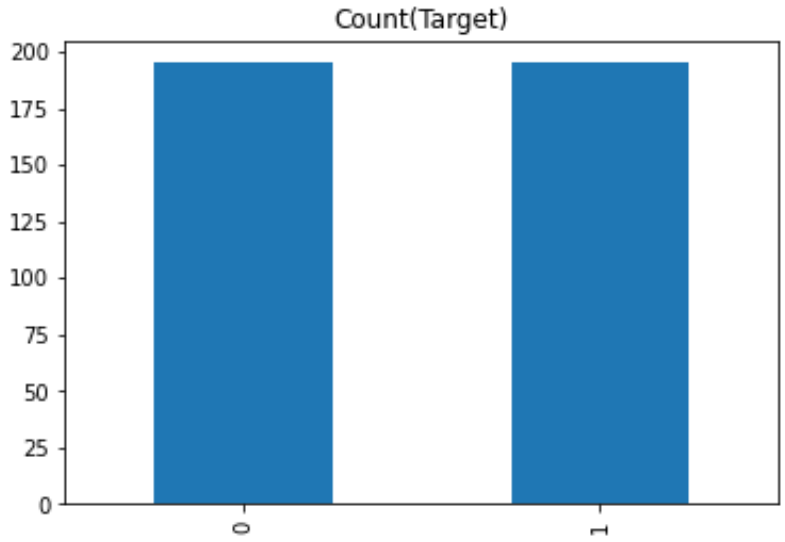
After Balancing the dataset

```
After Balancing our dataset :
Counter({0: 129, 1: 124})
balanced Precision: 0.986
balanced Recall: 1.000
balanced F1 Score: 0.993
Confusion matrix:
 [[65  1]
 [ 0 71]]
```

Confusion Matrix:



The BarGraph is as following:



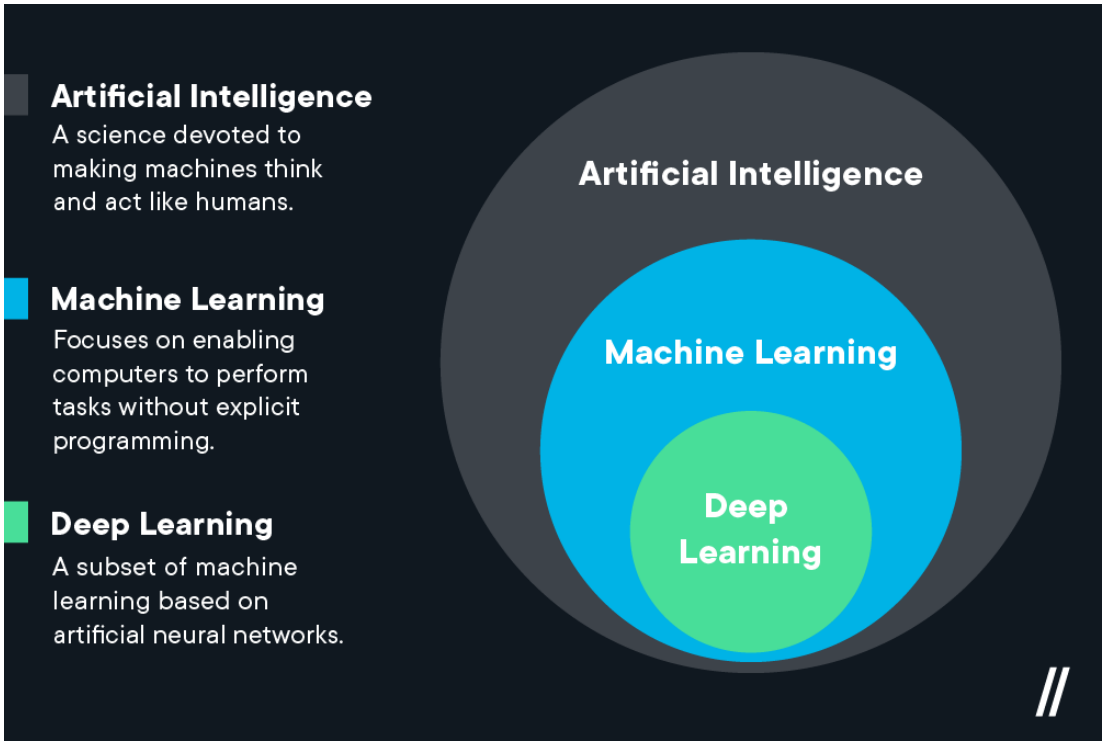
- **We have used Naïve Bayes Classification as specified**
- **We can see the dataset is clearly unbalanced as the Class: 0 has a count of around 129 whereas the Class: 1 has a count of 124.**
- **The dataset is almost equally aligned to Class: 0 and to Class: 1**
- **The F1_Score is: 99.3% which is almost perfect.**

QUESTION:

Differentiate between Machine Learning and Deep Learning. Elaborate with suitable example.

Machine learning employs a set of algorithms to analyse and interpret data, learn from it, and make the best judgments possible based on those learnings**. Deep learning**, on the other hand, divides algorithms into numerous layers to build a "**artificial neural network**." This neural network is capable of self-learning and making intelligent decisions.

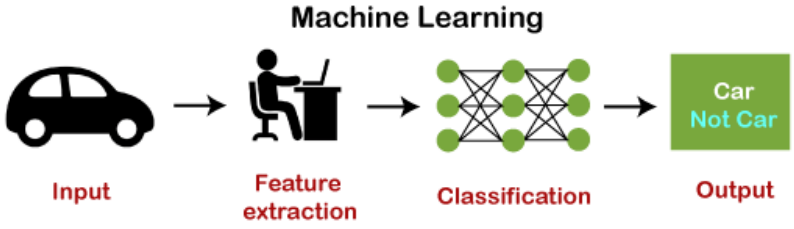
Machine Learning: Machine learning is a subset of Artificial Intelligence (AI) that allows a system to learn and grow from its experiences without having to be programmed to that level. Data is used by Machine Learning to learn and get accurate outcomes. Machine learning is concerned with the creation of a computer software that can access data and learn from it.
Deep Learning: Deep Learning is a subset of Machine Learning where the artificial neural network, the recurrent neural network comes in relation. The algorithms are created exactly just like machine learning but it consists of many more levels of algorithms. All these networks of the algorithm are together called as the artificial neural network. In much simpler terms, it replicates just like the human brain as all the neural networks are connected in the brain, exactly is the concept of deep learning. It solves all the complex problems with the help of algorithms and its process.



Machine Learning	Deep Learning
Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned	Deep learning structures algorithms in layers to create an “artificial neural network” that can learn and make intelligent decisions on its own
Can train on lesser training data	Requires large data sets for training
Takes less time to train	Takes longer time to train
Trains on CPU	Trains on GPU for proper training
The output is in numerical form for classification and scoring applications	The output can be in any form including free form elements such as free text and sound
Limited tuning capability for hyper parameter tuning	Can be tuned in various ways
Machine Learning is a superset of Deep Learning	Deep Learning is a subset of Machine Learning
The data represented in Machine Learning is quite different as compared to Deep Learning as it uses structured data	The data representation is used in Deep Learning is quite different as it uses neural networks(ANN).
Machine Learning is an evolution of AI	Deep Learning is an evolution to Machine Learning. Basically it is how deep is the machine learning.
Uses various types of automated algorithms that turn to model functions and predict future action from data.	Uses neural network that passes data through processing layers to the interpret data features and relations.
Algorithms are detected by data analysts to examine specific variables in data sets.	Algorithms are largely self-depicted on data analysis once they’re put into production.
Machine Learning is highly used to stay in the competition and learn new things.	Deep Learning solves complex machine learning issues.

Example:

The working of machine learning models can be understood by the example of identifying the image of a cat or dog. To identify this, the ML model takes images of both cat and dog as input, extracts the different features of images such as shape, height, nose, eyes, etc., applies the classification algorithm, and predict the output. Consider the below image:



The deep learning model takes the images as the input and feed it directly to the algorithms without requiring any manual feature extraction step. The images pass to the different layers of the artificial neural network and predict the final output.
In deep learning, models use different layers to learn and discover insights from the data.

Some popular deep learning models are:

- **Convolutional Neural Network**
- **Recurrent Neural Network**
- **Autoencoders**
- **Classic Neural Networks, etc.**