**Title:**

## Assignment No: 10

Download Pima Indians Diabetes dataset. Use Naive Bayes‟ Algorithm for classification

* + - * Load the data from CSV file and split it into training and test datasets.
      * summarize the properties in the training dataset so that we can calculate probabilities and make predictions.
      * Classify samples from a test dataset and a summarized training dataset

Implement a classification algorithm that is Naïve Bayes. Implement the following operations:

1. Split the dataset into Training and Test dataset.
2. Calculate conditional probability of each feature in training dataset.
3. Classify sample from a test dataset.
4. Display confusion matrix with predicted and actual values.

### Fundamentals of R -Programming Languages

To learn the concept of Naïve Bayes classification algorithm, Bayes theorem.

### Bayes Theorem:

Bayes‟ Theorem is a way of finding a probability when we know certain other probabilities.

The formula is: P(A|B) = *P(A) P(B|A)***P(B)**

P(A|B): how often A happens given that B happens, written **P(A|B)**, P(B|A): how often B happens given that A happens, written **P(B|A)** P(A): and how likely A is on its own, written **P(A)**

P(B): and how likely B is on its own, written **P(B)**

Let us say P(Fire) means how often there is fire, and P(Smoke) means how often we see smoke, then:

P(Fire|Smoke) means how often there is fire when we can see smoke P(Smoke|Fire) means how often we can see smoke when there is fire

So the formula kind of tells us "forwards" P(Fire|Smoke) when we know "backwards" P(Smoke|Fire)

Example: If dangerous fires are rare (1%) but smoke is fairly common (10%) due to



barbecues, and 90% of dangerous fires make smoke then:

P(Fire|Smoke) =*P(Fire) P(Smoke|Fire)***P(Smoke)**

=*1% x 90%***10%**

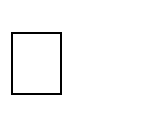
=9%

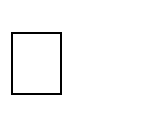
So the "Probability of dangerous Fire when there is Smoke" is 9%

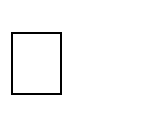
### Naive Bayes Classification

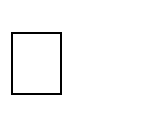
**Naive Bayes** is a simple, yet effective and commonly-used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting. It can also be represented using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification, and are a traditional solution for problems such as spam detection. Windows/Linux Operating Systems, RStudio, jdk.

### Applications:

**Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast.Thus, it could be used for making predictions in real time.

**Multi class Prediction:** This algorithm is also well known for multi class predictionfeature. Here we can predict the probability of multiple classes of target variable.

**Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiersmostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

**Recommendation System:** Navie Bayes Classifier Collaborative [Filtering](https://en.wikipedia.org/wiki/Collaborative_filtering) together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

### Input:

Structured Dataset : Pima Indians Diabetes Dataset File: PimaIndiansDiabetes.csv

### Output:

* 1. Splitted dataset according to Split ratio.
  2. Conditional probability of each feature.
  3. visualization of the performance of an algorithm with confusion matrix

### Conclusion:

Hence, we have studied classification algorithm that is Naïve Bayes classification.

Code:

# Importing Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Importing Dataset

data\_set = pd.read\_csv('PimaIndiansDiabetes.csv')

X = data\_set.iloc[:, :-1] # Independent Variables separated as X

y = data\_set.iloc[:, -1] # Dependent Variables into y

print("IV".center(40, "\_"), '\n', X.head())

print("DV".center(40, "\_"), '\n', y.head())

# Check for Missing Values

print(X.isnull().any())

print(y.isnull().any())

# Summary

print(X.info())

# Turning into Raw / NP format

X = X.values

y = y.values

# Noted the High Variance between All fields

# Scaling required

# 1. Splitting X,y into Train & Test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=0)

# 2. Scaling

from sklearn.preprocessing import StandardScaler

scaler\_X = StandardScaler()

X\_train = scaler\_X.fit\_transform(X\_train)

X\_test = scaler\_X.transform(X\_test)

# Machine: Classifier | NB: Gaussian Naive Bayes

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predictions

y\_pred = classifier.predict(X\_test)

# Validating Predictions using Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

from sklearn.metrics import precision\_recall\_fscore\_support

prf = precision\_recall\_fscore\_support(y\_test, y\_pred)

print('\t\t\t\t ZERO\t\t\tONE')

print('Precision\t:', prf[0]\*100)

print('Recall\t\t:', prf[1]\*100)

print('F1 Measure\t:', prf[2]\*100)

print('Support\t\t:', prf[3])

//OUTPUT

