|  |  |
| --- | --- |
| **TITLE** | **Data Analytics III** |
| **PROBLEM STATEMENT/ DEFINITION** | Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset. |
| **OBJECTIVE** | To understand how Naïve Bayes classification algorithm works on the given dataset |
| **OUTCOME** | To find the best scenario for the result to be achieved for a given data set using logistic regression. |
| **S/W PACKAGES AND HARDWARE APPARATUS USED** | Core 2 DUO/i3/i5/i7 64-bit processor  OS-LINUX 64 bit OS  Editor-gedit/Eclipse  S/w- Jupyter Notebook/ Weka/ Python |
| **REFERENCES** | 1. Chirag Shah, “A Hands-On Introduction To Data Science”, Cambridge University Press, (2020), ISBN : ISBN 978-1-108-47244-9. Curriculum for Third Year of Computer Engineering (2019 Course), Savitribai Phule Pune University http://collegecirculars.unipune.ac.in/sites/documents/Syllabus2020/Forms/AllItems.aspx #57/87 2. Giuseppe Bonaccorso, “ Machine Learning   Algorithms”, Packt Publishing Limited, ISBN-10:  1785889621, ISBN-13: 978-1785889622 |
| **STEPS** | **Refer to student activity flow chart if found necessary by subject teacher and relevant to the subjectmanual.**  **Describe steps only.** |
| **INSTRUCTIONS FOR WRITING JOURNAL** | 1. title 2. Problem statement 3. Learning objective 4. Learning outcome 5. Theory (includes methods, libraries and functions, 6. Analysis (as per assignment), 7. conclusion. |

P:F:-LTL-UG / 03 /R1

**ASSIGNMENT-NO-6**

**TITLE- Data Analytics III**

**PROBLEM STATEMENT/ DEFINITION-**Implement logistic regression using Python /R to perform classification on Social\_Network\_Ads.csv dataset.

ComputeConfusionmatrixtofindTP,FP,TN,FN,Accuracy,Errorrate,Precision,

Recall on the given dataset.

**LEARNING OBJECTIVE-**

To understand how logistic regression works on the given dataset.

**LEARNING OUTCOME**

To find the best scenario for the result to be achieved for a given data set using logistic regression

**THEORY-**

**(includes** methods, libraries and functions,

A naive Bayes classifier is called so because it's based on a naive condition, which implies

the conditional independence of causes. This can seem very difficult to accept in many

contexts where the probability of a particular feature is strictly correlated to another one.

**For example**, in spam filtering, a text shorter than 50 characters can increase the probability

of the presence of an image, or if the domain has been already blacklisted for sending the

same spam emails to million users, it's likely to find particular keywords.

Following three classification methods can be applied to the data set.

* 1. BernoulliNB()
  2. GaussianNB()
  3. MultinomialNB()

Bernoulli naive Bayes expects binary feature vectors; however, the class BernoulliNB has a

binarize parameter, which allows us to specify a threshold that will be used internally to

transform the features:

from sklearn.datasets import make\_classification

>>> nb\_samples = 300

>>> X, Y = make\_classification(n\_samples=nb\_samples, n\_features=2,

n\_informative=2, n\_redundant=0)

from sklearn.naive\_bayes import BernoulliNB

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)

>>> bnb = BernoulliNB(binarize=0.0)

>>> bnb.fit(X\_train, Y\_train)

>>> bnb.score(X\_test, Y\_test)

from sklearn.datasets import load\_digits

from sklearn.model\_selection import cross\_val\_score

>>> digits = load\_digits()

>>> gnb = GaussianNB()

>>> mnb = MultinomialNB()

Analysis (as per assignment)

**Classification metrics**

A classification task can be evaluated in many different ways to achieve specific objectives.

Of course, the most important metric is the accuracy, often expressed as:



In scikit-learn, it can be assessed using the built-in accuracy\_score() function:

from sklearn.metrics import accuracy\_score

>>> accuracy\_score(Y\_test, lr.predict(X\_test))

Let us understand the confusion matrix. In many cases, it's necessary to be able to differentiate between different kinds of misclassifications (we're considering the binary case with the conventional notation: 0-negative, 1-positive), because the relative weight is quite different. For this reason, we introduce the following definitions:

**True positive: A positive sample correctly classified**

**False positive: A negative sample classified as positive**

**True negative: A negative sample correctly classified**

**False negative: A positive sample classified as negative**

Scikit learn supports the following method to compute the confusion matrix and calculating the precision and recall

from sklearn.metrics import confusion\_matrix

>>> cm = confusion\_matrix(y\_true=Y\_test, y\_pred=lr.predict(X\_test))

cm[::-1, ::-1]

from sklearn.metrics import precision\_score

>>> precision\_score(Y\_test, lr.predict(X\_test))

from sklearn.metrics import recall\_score

>>> recall\_score(Y\_test, lr.predict(X\_test))

**CONCLUSION-Thus**, Naïve Bay’s Classifier model on the given data set is applied .The results shows the classification using various methods with precision and recall.