EXPERIMENT No: 03 DATE: 18-11-20

PROBLEM STATEMENT

Classify the messages as spam and ham using naive bayes algorithm on sms dataset.

OBJECTIVE : To implement naive bayes algorithm to classify messages into spam and ham. on sms dataset.

THEORY :

Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem, used in a wide variety of classification tasks. Naive Bayes classifiers are a collection of classification algorithms based on 'Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle. i.e every pair of features being classified is independent of each other. The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome.

⇒ Bayes Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes theorem is stated mathematically as the following equation :

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

where A and B are events and $P(B) \neq 0$.

⇒ finding the probability of event A, given the event B is true. Event B is also termed as evidence.

⇒ P(A) is the prior of A. (probability of event before evidence is seen)

→ P(A|B) is a posterior probability of B, i.e probability of event after evidence is seen.

How Naive Bayes Algorithm Works?

Step 1: Convert the data set into a frequency table

Step 2: Create likelihood table by finding the probabilities.

Step 3: Now, use Naive Bayes equation to calculate the posterior probability for each class. The class with the higher posterior probability is the outcome of prediction.

DATASET USED: SMS Dataset

The SMS spam collection is a set of SMS tagged messages that have been collected for sms spam research. It contains one set of sms messages in English of 5574 messages, tagged according being ham or spam. The files contains one message per line. Each line is composed by two columns:

V1 contains the label (ham or spam) and

V2 contains the raw text.

PROGRAM:

```
install.packages("tm")  #textmining
install.packages("wordcloud")
install.packages("e1071")

library(tm)
library(wordcloud)
library(e1071)

sms_spam_df <- read.csv(file="E:\\Lenovo\\Users\\
 sms_spam_csv", stringsAsfactors = F) str(sms_spam_
 df)

#head(sms_spam_df) head display first 5 rows
#table(sms_spam_df$category) total count of
 2 variables.

# creating a corpus(collection of documents)
 sms_corpus <- VCorpus(VectorSource(sms_spam_df$text
print(sms_corpus)

#view a summary of the first and second sms
 msgs in the corpus
#inspect (sms_corpus[1:2])
#data pre-processing
clean_corpus <- tm_map(sms_corpus, content_
 transformer(tolower))
#as.character(clean_corpus[[1]]).
```

#remove_numbers.
clean_corpus <- tm_map(clean_corpus, removeNumbers)
clean_corpus <- tm_map(clean_corpus, removePunc-
-tuation)

#remove stopwords.
#stopwords()[1:15]
clean_corpus <- tm_map(clean_corpus, removeWords,
stopwords())

#remove whitespaces
clean_corpus <- tm_map(clean_corpus, stripWhite
-space)
#display detailed information
inspect(clean_corpus[1:3])
#tokenize each msg into words to build key
structure
sms_dtm <- DocumentTermMatrix(clean_corpus)
str(sms_dtm)

#the which() function will return the position of
elements
spam_indices <- which(sms_spam_df$category
== "spam")
ham_indices <- which(sms_spam_df$category == "ham")
#spam_indices
#ham_indices.

```
wordcloud(clean_corpus[ham_indices], min.freq=40)
#lock at most 40 common words.
wordcloud(clean_corpus[spam_indices], min.freq=40)

#split out training and test cases.
#split raw data
sms_raw_train <- sms_spam_df[1:4169,]
sms_raw_test <- sms_spam_df[4170:5559,]

#split document term matrix
sms_dtm_train <- sms_dtm[1:4169,]
sms_dtm_test <- sms_dtm[4170:5559,]
#split.corpus.
sms_corpus_train <- clean_corpus[1:4169]
sms_corpus_test <- clean_corpus[4170:5559]
#create separate corpuses for spam and
ham.

spam <- subset(sms_raw_train, category ==
                "spam")
ham <- subset(sms_raw_train, category == "ham")

#reducing DTM.
five_times_words <- findFreqTerms(sms_dtm_
        train, 5)
#word should appear inatleast 5 msg.

sms_train <- DocumentTermMatrix(sms_corpus_
train, control = list(dictionary = five_times_words))
```

```
sms_test <- DocumentTermMatrix(sms_corpus_test,
control=list(dictionary = five_times_words))

convert_count <- function(x) {
 y <- ifelse(x > 0, 1, 0)
 y <- factor(y, levels = c(0,1), labels = c("No", "Yes"))
 y
}

sms_train <- apply(sms_train, 2, convert_count)
sms_test <- apply(sms_test, 2, convert_count)

sms_classifier <- naiveBayes(sms_train, factor(
 sms_raw_train$category))
sms_test_pred <- predict(sms_classifier,
newdata = sms_test)
k=table(sms_test_pred, sms_raw_test$category)
k
accuracy = sum(diag(k)) /sum(k) * 100
accuracy
```

CONCLUSION : Was able to understand Naive bayes algorithm implementation and also classify messages as spam and ham on sms dataset.

REFERENCES :

⇒ Cathy O'Neil, Rachel Schitt "Doing data Science" O Reilly Media Inc
⇒ Siaan Ozdemeir, Principles of data science packet pubbisher.

EXPERIMENT No:04 DATE:04-12-20

PROBLEM STATEMENT

Implement logistic regression algorithm on the iris flower dataset to classify the flower into different types.

OBJECTIVE OF THE EXPERIMENT: To understand the implementation of the logistic regression algorithm on the iris flower dataset to classify the flower into different types

THEORY:

Logistic regression is one of the most popular machine learning algorithms for binary classification. This is because it is a simple algorithm that performs very well on a wide range of problems. In statistics the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose. This can be extended to model several classes of events such as determining whether an image contains an object. Each object detected in the image is assigned a probability between 0 & 1. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model.

Logistic function is defined as $T = \frac{1}{(1+e^{-x})}$

'e' is Eulers constant and 'x' is an input

The logistic regression model takes real-valued inputs and makes a prediction as to the probability of the input belonging to the default class (class 0). If the probability is <0.5 we can take the output as a prediction for the default class (class 0), otherwise the prediction is for the other class (class 1)

DATASET USED:

IRIS FLOWER DATA SET

The iris flower data set or 'fisher's iris data set is a multivariate data set introduced by the British statistician, biologist Ronald fisher.
The Dataset contains a set of 150 records under five attributes - sepal length, sepal width, petal length, petal width and species. The data set contains 50 samples from each of three species of Iris (Iris setosa, Iris virginica, Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals in centimeters.

PROGRAM1:

```
library (dplyr)
Library (ggplot2)

iris.small ← filter (iris, Species %in% c ("virginica",
                                           "versicolor"))
#logistic regression.
glm.out ← glm (Species ~ Sepal.Width + Sepal.Length +
  Petal.Width + Petal.Length, data = iris.small,
  family = binomial)
summary (glm.out)

glm (formula = Species ~ Sepal.Width + Sepal.Length +
  Petal.Width + Petal.Length, family = binomial,
  data = iris.small)

glm.out ← glm (Species ~ Sepal.Width + Petal.Width +
  Petal.Length, family = binomial, data = iris.small)
summary (glm.out)

# make a plot of fitted probability as a function.
lr-data ← data.frame (predictor = glm.out$ linear
  predictors, prob = glm.out $ fitted.values, Species =
  iris.small$Species)
ggplot (lr-data, aes (x = predictor, y = prob, color =
      Species)) + geom_point ()

#make a density plot.
ggplot (lr-data, aes (x = predictor, fill = Species)) +
  geom_density (alpha = 5)
```

```
plant1 <- data.frame(Sepal.Length=6.4, Sepal.Width=
  2.8, Petal.Length=4.6, Petal.Width=1.8)
plant2 <- data.frame(Sepal.Length=6.3,
  Sepal.Width=2.5, Petal.Width=Petal.Length=4.1
  Petal.Width=1.7)
plant3 <- data.frame(Sepal.Length=6.7, Sepal.Width
  =3.3, Petal.Length=5.2, Petal.Width=2.3)

predict(glm.out, plant1, type="response")
predict(glm.out, plant2, type="response")
predict(glm.out, plant3, type="response").
```

CONCLUSION: After performing this experiment would understand about logistic regression algorithms and implement the same to classify the flower of different types on iris flower dataset.

REFERENCES:

⇒ Cathy O'Neil, Rachel Schutt "Doing Data Science", O'Reilly Media, Inc.
⇒ Sinan Ozdemir, "Principles of Data Science". Packt publisher December 2016