ASSIGNMENT

"Train a model to classify a movie review into good (positive) and bad (negative) categories"

Submitted By:

SHOBHA H R

21GACS1013

M.Tech, CSE

University Visvesvaraya College of Engineering

MODEL ARCHITECTURE

1. **Multinomial Naïve Bayes Algorithm:**

   The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance. The Naive Bayes classifier is made up of a number of algorithms that all have one thing in common: each feature being classed is unrelated to any other feature. A feature's existence or absence has no bearing on the inclusion or exclusion of another feature.

```
TRAINMULTINOMIALNB(C, D)
 1  V ← EXTRACTVOCABULARY(D)
 2  N ← COUNTDOCS(D)
 3  for each c ∈ C
 4  do N_c ← COUNTDOCSINCLASS(D, c)
 5     prior[c] ← N_c / N
 6     text_c ← CONCATENATETEXTOFALLDOCSINCLASS(D, c)
 7     for each t ∈ V
 8     do T_ct ← COUNTTOKENSOFTERM(text_c, t)
 9     for each t ∈ V
10     do condprob[t][c] ← (T_ct + 1) / Σ_t'(T_ct' + 1)
11  return V, prior, condprob

APPLYMULTINOMIALNB(C, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ C
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg max_c∈C score[c]
```

Fig1: Multinomial Naïve Bayes Algorithm

How Multinomial Naive Bayes works?

Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes. To understand Naive Bayes theorem's working, it is important to understand the Bayes theorem concept first as it is based on the latter.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

P(A|B) = P(A) * P(B|A)/P(B)

Where we are calculating the probability of class A when predictor B is already provided.

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability

This formula helps in calculating the probability of the tags in the text.

Result:

```
In [44]:
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion
import warnings
warnings.filterwarnings('ignore')

In [47]:
mnb = MultinomialNB()
mnb.fit(x_train, y_train)
mnb_pred = mnb.predict(x_test)
mnb_acc = accuracy_score(mnb_pred, y_test)
print("Test accuracy: {:.2f}%".format(mnb_acc*100))

Test accuracy: 86.11%

In [48]:
print(confusion_matrix(y_test, mnb_pred))
print("\n")
print(classification_report(y_test, mnb_pred))

[[6276 1195]
 [ 871 6533]]


              precision    recall  f1-score   support

           1       0.88      0.84      0.86      7471
           2       0.85      0.88      0.86      7404

    accuracy                           0.86     14875
   macro avg       0.86      0.86      0.86     14875
weighted avg       0.86      0.86      0.86     14875
```

Fig2: Accuracy using MNB Model is 86.11%

2. **Logistic Regression Algorithm:**

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for a given set of features (or inputs), X. Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as "1". Just like linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

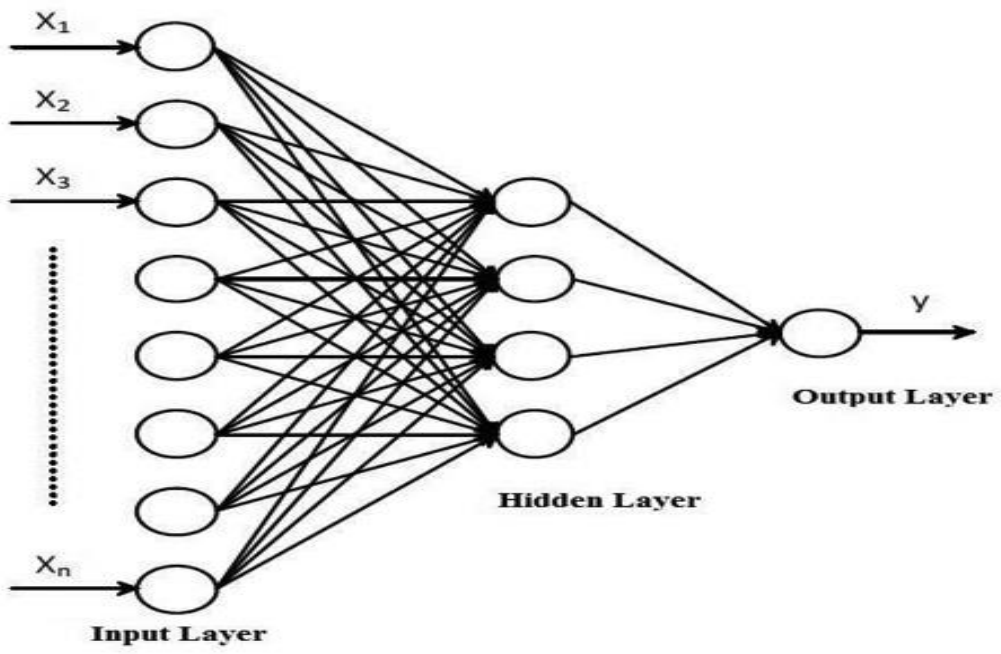
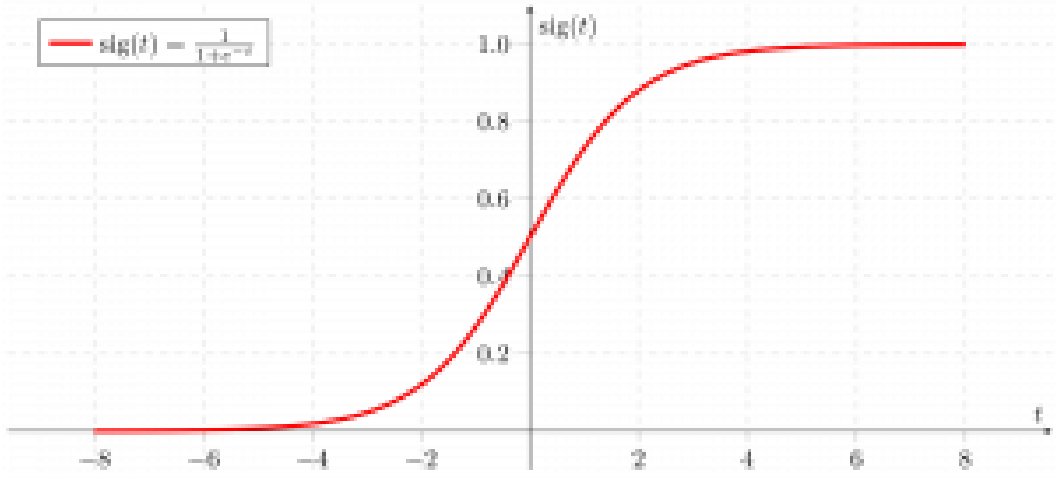


Figure 1: A feed forward neural network architecture



Result:

In [49]:
```
1 logreg = LogisticRegression()
2 logreg.fit(x_train, y_train)
3 logreg_pred = logreg.predict(x_test)
4 logreg_acc = accuracy_score(logreg_pred, y_test)
5 print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 89.60%
```


In [50]:
```
1 print(confusion_matrix(y_test, logreg_pred))
2 print("\n")
3 print(classification_report(y_test, logreg_pred))
```

```
[[6771  700]
 [ 847 6557]]


              precision    recall  f1-score   support

           1       0.89      0.91      0.90      7471
           2       0.90      0.89      0.89      7404

    accuracy                           0.90     14875
   macro avg       0.90      0.90      0.90     14875
```

Fig3: Accuracy using Logistic Regression Model is 89.60%