

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Report submitted for the subject**

**Artificial Intelligence- CS53**

**Semester- 5**

Submitted by

Hrithik B Muttin 1MS19CS051

K J Sundeep Kumar 1MS19CS057

Mahesh 1MS19CS066

Manoj Kumar M 1MS19CS068

Manojkumar G S 1MS19CS069

*in partial fulfillment for the award of the degree of*

**Bachelor of Engineering in Computer Science & Engineering**

Under the guidance of

Dr. Sangeetha V

Assistant Professor, Dept. of CSE

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu/), February 2022

**Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

# Department of Computer Science & Engineering



**CERTIFICATE**

This is to certify that the Project work carried out by **Hrithik B Muttin(1MS19CS051), K J SundeepKumar(1MS19CS057),Mahesh(1MS19CS066), Manoj Kumar M(1MS19CS068) Manojkumar G S(1MS19CS069)** as 20-mark component for the course Artificial Intelligence (CS53), V semester B.E, CSE during the academic year Oct 2021- Feb 2022 satisfies the academic requirements for awarding the marks.

### Signature of the Faculty

#### **Contents**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **PART A – AI Lab Program(5M)** | **Page No.** |
| 1 | Demonstrate the training, testing and prediction of machine learning models for Play Cricket Dataset using below algorithms. Tabulate accuracy, precision, Recall and F1- score. a. Decision Tree  b. Random Forest  c. SVM d. Naïve Bayes |  |
| 2 | Demonstrate the training, testing and prediction of machine learning models for Yelp Restaurant Dataset using below algorithms. Tabulate accuracy, precision, Recall and F1-score.  a. Decision Tree  b. Random Forest  c. SVM  d. Naïve Bayes |  |
| 3 | Implement the below algorithms given a weighted graph and heuristic values.  a. Depth First Search  b. Breadth First Search  c. A\* |  |
|  | **PART B – Project Report(15M)** |  |

**PART A PROGRAM 1**

**Play Cricket Dataset using Random Forest:**

**#Importing required libraries**

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import precision\_recall\_curve

from sklearn.metrics import plot\_precision\_recall\_curve

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

**#Read dataset**

play\_cricket = pd.read\_csv("CricketDataset.csv")

play\_cricket.head()

play\_cricket.tail()

**#data processing**

number = LabelEncoder()

play\_cricket['Outlook'] = number.fit\_transform(play\_cricket['Outlook'])

play\_cricket['Humidity'] = number.fit\_transform(play\_cricket['Humidity'])

play\_cricket['Wind'] = number.fit\_transform(play\_cricket['Wind'])

print(play\_cricket)

**#data splitting**

features = ["Outlook","Humidity","Wind"]

target = "Playcricket"

features\_train, features\_test, target\_train, target\_test = train\_test\_split(play\_cricket[features],play\_cricket[target],test\_size = 0.25,random\_state = 47)

model=LogisticRegression()

model.fit(features\_train,target\_train)

pred=model.predict(features\_test)

precision=precision\_score(target\_test,pred,average=None,zero\_division=1)  **#recall**

recall= recall\_score(target\_test,pred,average=None,zero\_division=1)  **#precision score**

print(precision[1])

print(recall[1])

print(2\*((precision\*recall)/(precision+recall))[1])

clf = RandomForestClassifier(n\_estimators=20, criterion='entropy')

clf.fit(features\_train, target\_train)

y\_pred = clf.predict(features\_test)

print(y\_pred)

accuracy = accuracy\_score(target\_test, y\_pred) **#accuracy**

print(accuracy)

cm = confusion\_matrix(target\_test, y\_pred)  **#confusion matrix**

print(cm)

print (clf.predict([[1,0,0]]))  **#predict**

**Play Cricket Dataset using Decision Tree:**

**#Importing required libraries**

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import confusion\_matrix

**#reading dataset**

play\_cricket = pd.read\_csv("CricketDataset.csv")

play\_cricket.head()

play\_cricket.tail()

**#data processing**

number = LabelEncoder()

play\_cricket['Outlook'] = number.fit\_transform(play\_cricket['Outlook'])

play\_cricket['Humidity'] = number.fit\_transform(play\_cricket['Humidity'])

play\_cricket['Wind'] = number.fit\_transform(play\_cricket['Wind'])

print(play\_cricket)

features = ["Outlook","Humidity","Wind"]

target = "Playcricket"

features\_train,features\_test,target\_train,target\_test = train\_test\_split(play\_cricket[features],play\_cricket[target],test\_size = 0.25,random\_state = 47)

**#required classifier**

clf = DecisionTreeClassifier(criterion = 'entropy')

clf.fit(features\_train, target\_train)

y\_pred = clf.predict(features\_test)

accuracy = accuracy\_score(target\_test, y\_pred)

print(accuracy)

print (clf.predict([[0,0,0], [1,1,1]]))

**Play Cricket Dataset using SVM:**

**#Importing required libraries**

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from sklearn import SVM

from sklearn.metrics import confusion\_matrix

**#reading dataset**

play\_cricket = pd.read\_csv("CricketDataset.csv")

play\_cricket.head()

play\_cricket.tail()

**#data processing**

number = LabelEncoder()

play\_cricket['Outlook'] = number.fit\_transform(play\_cricket['Outlook'])

play\_cricket['Humidity'] = number.fit\_transform(play\_cricket['Humidity'])

play\_cricket['Wind'] = number.fit\_transform(play\_cricket['Wind'])

print(play\_cricket)

features = ["Outlook","Humidity","Wind"]

target = "Playcricket"

features\_train,features\_test,target\_train,target\_test = train\_test\_split(play\_cricket[features],play\_cricket[target],test\_size = 0.25,random\_state = 47)

**#required classifier**

clf =SVM.SVC()

clf.fit(features\_train, target\_train)

y\_pred = clf.predict(features\_test)

accuracy = accuracy\_score(target\_test, y\_pred)

print(accuracy)

tn,fp,fn,tp = confusion-matrix(target\_test,y\_pred).ravel()

recall = tp\*100/(tp+fn) **#recall**

precision = tp\*100/(tp+fp) **#precision**

f1\_score=(2\*recall\*precision)/(precision+recall)  **#f1\_score**

print(recall)

print(precision)

print(f1\_score)

print(clf.predict([1,0,0]))  **#Predict**

**Play Cricket Dataset using Naive Bayes:**

**#Importing required libraries**

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from sklearn.naive-bayes import GaussianNB

from sklearn.metrics import confusion\_matrix

**#reading dataset**

play\_cricket = pd.read\_csv("CricketDataset.csv")

play\_cricket.head()

play\_cricket.tail()

**#data processing**

number = LabelEncoder()

play\_cricket['Outlook'] = number.fit\_transform(play\_cricket['Outlook'])

play\_cricket['Humidity'] = number.fit\_transform(play\_cricket['Humidity'])

play\_cricket['Wind'] = number.fit\_transform(play\_cricket['Wind'])

print(play\_cricket)

features = ["Outlook","Humidity","Wind"]

target = "Playcricket"

features\_train,features\_test,target\_train,target\_test = train\_test\_split(play\_cricket[features],play\_cricket[target],test\_size = 0.25,random\_state = 47)

**#required classifier**

clf =GaussianNB()

clf.fit(features\_train, target\_train)

y\_pred = clf.predict(features\_test)

accuracy = accuracy\_score(target\_test, y\_pred)

print(accuracy)

tn,fp,fn,tp = confusion-matrix(target\_test,y\_pred).ravel()

recall = tp\*100/(tp+fn) **#recall**

precision = tp\*100/(tp+fp) **#precision**

f1\_score=(2\*recall\*precision)/(precision+recall)  **#f1\_score**

print(recall)

print(precision)

print(f1\_score)

print(clf.predict([1,0,0]))  **#Predict**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classifier** | **Accuracy** | **Precision** | **Recall** | **F1\_score** |
| **Decision tree** | **75%** | **75** | **100** | **85.714** |
| **Random Forest** | **75%** | **75** | **100** | **85.714** |
| **SVM** | **75%** | **75** | **100** | **85.714** |
| **Naive Bayes** | **50%** | **66.667** | **66.667** | **66.667** |

**PROGRAM 2**

**Yelp Restaurant Dataset using Decision Tree:**

**#Importing required libraries**

import pandas as pd

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

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics

import warnings

warnings.filterwarnings('ignore')

from sklearn.tree import DecisionTreeClassifier

**#reading dataset**

res = pd.read\_csv("/content/drive/MyDrive/ailab/Resturant.csv")

res.head()

res.tail()

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

print(xtrain)

count\_vect=CountVectorizer()

xtrain\_dtm=count\_vect.fit\_transform(xtrain)

xtest\_dtm=count\_vect.transform(xtest)

print(count\_vect.get\_feature\_names\_out())

df=pd.DataFrame(xtrain\_dtm.toarray(),columns=count\_vect.get\_feature\_names\_out())

print(df)

print(xtrain\_dtm)

**#required classifier**

clf = DecisionTreeClassifier(criterion = 'entropy')

clf.fit(xtrain\_dtm,ytrain)

predicted = clf.predict(xtest\_dtm)

print(predicted)

print('Confusion matrix')

print(metrics.confusion\_matrix(ytest,predicted))

print('Accuracy of the classifier is',metrics.accuracy\_score(ytest,predicted))

Precision=metrics.precision\_score(ytest,predicted)

print(Precision)

Recall=metrics.recall\_score(ytest,predicted)

print(Recall)

F1score = 2 \* (Precision \* Recall) / (Precision + Recall)

print(F1score)

**Yelp Restaurant Dataset using Random Forest:**

**#Importing required libraries**

import pandas as pd

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

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics import warnings

warnings.filterwarnings('ignore')

from sklearn.ensemble import RandomForestClassifier

**#reading dataset**

res = pd.read\_csv("/content/drive/MyDrive/ailab/Resturant.csv")

res.head()

res.tail()

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

print(xtrain)

count\_vect=CountVectorizer()

xtrain\_dtm=count\_vect.fit\_transform(xtrain)

xtest\_dtm=count\_vect.transform(xtest)

print(count\_vect.get\_feature\_names\_out())

df=pd.DataFrame(xtrain\_dtm.toarray(),columns=count\_vect.get\_feature\_names\_out())

print(df)

print(xtrain\_dtm)

**#required classifier**

clf = RandomForestClassifier(n\_estimators=20, criterion='entropy')

clf.fit(xtrain\_dtm,ytrain)

predicted = clf.predict(xtest\_dtm)

print(predicted)

print('Confusion matrix')

print(metrics.confusion\_matrix(ytest,predicted))

print('Accuracy of the classifier is',metrics.accuracy\_score(ytest,predicted))

Precision=metrics.precision\_score(ytest,predicted)

print(Precision)

Recall=metrics.recall\_score(ytest,predicted)

print(Recall)

F1score = 2 \* (Precision \* Recall) / (Precision + Recall)

print(F1score)

**Yelp Restaurant Dataset using SVM:**

**#Importing required libraries**

import pandas as pd

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

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import metrics import warnings

warnings.filterwarnings('ignore')

from sklearn.pipeline import make\_pipeline

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler

**#required Dataset**

res = pd.read\_csv("/content/drive/MyDrive/ailab/Resturant.csv")

res.head()

res.tail()

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

print(xtrain)

count\_vect=CountVectorizer()

xtrain\_dtm=count\_vect.fit\_transform(xtrain)

xtest\_dtm=count\_vect.transform(xtest)

print(count\_vect.get\_feature\_names\_out())

df=pd.DataFrame(xtrain\_dtm.toarray(),columns=count\_vect.get\_feature\_names\_out())

print(df)

print(xtrain\_dtm)

**#required classifier**

clf = make\_pipeline(StandardScaler(with\_mean=False), SVC(gamma='auto'))

clf.fit(xtrain\_dtm,ytrain)

predicted = clf.predict(xtest\_dtm)

print(predicted)

print('Confusion matrix')

print(metrics.confusion\_matrix(ytest,predicted))

print('Accuracy of the classifier is',metrics.accuracy\_score(ytest,predicted))

Precision=metrics.precision\_score(ytest,predicted)

print(Precision)

Recall=metrics.recall\_score(ytest,predicted)

print(Recall)

F1score = 2 \* (Precision \* Recall) / (Precision + Recall)

print(F1score)

**Yelp Restaurant Dataset using Naive Beyes:**

#Import Libraries

import pandas as pd

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

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn import metrics import warnings warnings.filterwarnings('ignore')

**#required Dataset**

res = pd.read\_csv("/content/drive/MyDrive/ailab/Resturant.csv")

res.head()

res.tail()

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y)

print(xtrain)

count\_vect=CountVectorizer()

xtrain\_dtm=count\_vect.fit\_transform(xtrain)

xtest\_dtm=count\_vect.transform(xtest)

print(count\_vect.get\_feature\_names\_out())

df=pd.DataFrame(xtrain\_dtm.toarray(),columns=count\_vect.get\_feature\_names\_out())

print(df)

print(xtrain\_dtm)

**#required classifier**

clf = MultinomialNB().fit(xtrain\_dtm,ytrain)

predicted = clf.predict(xtest\_dtm)

print(predicted)

print('Confusion matrix')

print(metrics.confusion\_matrix(ytest,predicted))

print('Accuracy of the classifier is',metrics.accuracy\_score(ytest,predicted))

Precision=metrics.precision\_score(ytest,predicted)

print(Precision)

Recall=metrics.recall\_score(ytest,predicted)

print(Recall)

F1score = 2 \* (Precision \* Recall) / (Precision + Recall)

print(F1score)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classifier** | **Accuracy** | **Precision** | **Recall** | **F1\_score** |
| **Decision tree** | **76.4%** | **80.2%** | **67.5%** | **73.3** |
| **Random Forest** | **80%** | **79.4%** | **75.6%** | **77.6%** |
| **NaiveBayes** | **79.6%** | **80%** | **81.8%** | **80.89%** |
| **SVM** | **73.2%** | **72.7%** | **81.75%** | **85.71%** |

**PROGRAM 3**

**DFS:**

def dfs(visited, graph, node):  **#function for dfs**

if node not in visited:

print (node, end=" ")

visited.add(node)

for neighbour in graph[node]:

dfs(visited, graph, neighbour)

graph = { '5' : ['3','7'], '3' : ['2', '4'], '7' : ['8'], '2' : [], '4' : ['8'], '8' : [] }

visited = set()

print("Following is the Depth-First Search")

dfs(visited, graph, '5')

**OUTPUT:**

Following is the Depth-First Search 5 3 2 4 8 7

**BFS:**

def bfs(visited, graph, node):

visited.append(node)

queue.append(node)

while queue:

m = queue.pop(0)

print (m, end = " ")

for neighbour in graph[m]:

if neighbour not in visited:

visited.append(neighbour)

queue.append(neighbour)

graph = { '5' : ['3','7'], '3' : ['2', '4'], '7' : ['8'], '2' : [], '4' : ['8'], '8' : [] }

visited = []

queue = []

print("Following is the Breadth-First Search") bfs(visited, graph, '5')

**OUTPUT:**

Following is the Breadth-First Search 5 3 7 2 4 8

**A\* algorithm:**

def aStarAlgo(start\_node, stop\_node):

open\_set = set(start\_node)

closed\_set = set()

g = {}

parents = {}

g[start\_node] = 0

parents[start\_node] = start\_node

while len(open\_set) > 0:

n = None

for v in open\_set:

if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):

n = v

if n == stop\_node or Graph\_nodes[n] == None:

pass

else:

for (m, weight) in get\_neighbors(n):

if m not in open\_set and m not in closed\_set:

open\_set.add(m)

parents[m] = n

g[m] = g[n] + weight

else:

if g[m] > g[n] + weight:

g[m] = g[n] + weight

parents[m] = n

if m in closed\_set:

closed\_set.remove(m)

open\_set.add(m)

if n == None:

print('Path does not exist!')

return None

if n == stop\_node:

path = []

while parents[n] != n:

path.append(n)

n = parents[n]

path.append(start\_node)

path.reverse()

print('Path found:')

return path

open\_set.remove(n)

closed\_set.add(n)

print('Path does not exist!')

return None

def get\_neighbors(v):

if v in Graph\_nodes:

return Graph\_nodes[v]

else:

return None

def heuristic(n):

H\_dist = {'A': 5,'B': 3,'C': 4,'D': 2,'E': 4,'F': 6}

return H\_dist[n]

Graph\_nodes = {'A': [('B', 1), ('F', 10)],'B': [('C', 2), ('D', 1)],'C': [('E', 5)],'D': [('E', 3), ('F', 4)],'E': [('F', 2)]}

aStarAlgo('A', 'F')

**Output:**

Path found: ['A', 'B', 'D', 'F']

**PART B**

**TITLE - Detecting Fake Movie Reviews Through Sentimental Analysis Using SVM and KNN**

# Abstract:

# Abstract— Recently, Sentiment Analysis (SA) has become one of the most interesting topics in text analysis, due to its promising commercial benefits. One of the main issues facing SA is how to extract emotions inside the opinion, and how to detect fake positive reviews and fake negative reviews from opinion reviews. Moreover, the opinion reviews obtained from users can be classified into positive or negative reviews, which can be used by a consumer to select a product. This paper aims to classify movie reviews into groups of positive or negative polarity by using machine learning algorithms. In this study, we analyze online movie reviews using SA methods in order to detect fake reviews. SA and text classification methods are applied to a dataset of movie reviews. More specifically, we compare five supervised machine learning algorithms: Naïve Bayes (NB), Support Vector Machine (SVM), K-Nearest Neighbors (KNN-IBK), K Star (K\*) and Decision Tree (DT-J48) for sentiment classification of reviews using two different datasets, including movie review dataset V2.0 and movie reviews dataset V1.0. The measured results of our experiments show that the SVM algorithm outperforms other algorithms, and that it reaches the highest accuracy not only in text classification, but also in detecting fake reviews.

*Keywords: KNN, SVM MODEL, FAKE MOVIE REVIEWS, SENTIMENT ANALYSIS, CLASSIFICATION, CLASSIFICATION OF REVIEWS, SENTIMENTAL REVIEW ANALYSIS*

1. **INTRODUCTION**
2. **LITERATURE SURVEY**
3. **METHOD and MATERIAL**
4. **RESULTS and DISCUSSION**
5. **CONCLUSION**
6. **SCOPE FOR FUTURE WORK**

**REFERENCES**

**Introduction**

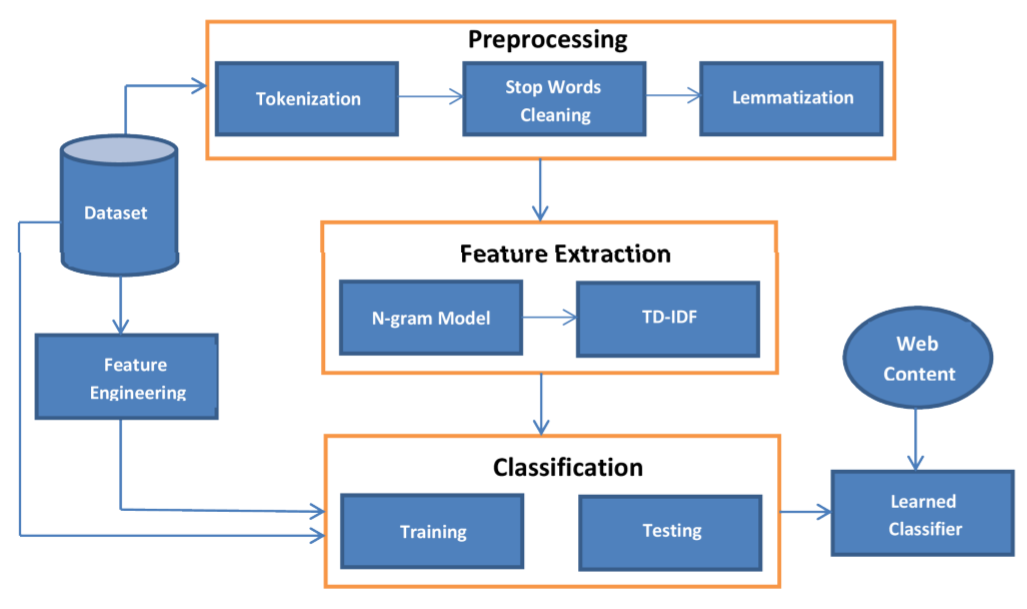
Sentiment Analysis is the process of determining whether a piece of writing/text is positive, negative or neutral. Reviews of any item/app are very now-a-days because they give the information to the user about the product, Its quality and many more things. In today’s world people deliberately post the fake reviews in order to distract the costumer and get a lead from market competitor . Hence by the fake reviews people and mostly merchants can get in a big trouble. In this research paper we aim to detect the reviews of different users in negative, positive and neutral using sentiment analysis . More particularly, we try to compare two machine learning algorithms namely: KNN and Support Vector Machine (SVM) for sentiment classification of reviews

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ref**  **No.** | **Year** | **Title of the Paper** | **Methodology used** | **Outcome** | **Issues** |
| [1] | 2017 | Detecting Fake Reviews through Sentiment Analysis Using Machine Learning Techniques. | Using Weka tool Compared Different Sentiment Algorithms. | SVM is Considered To be best algorithm For Classifying Reviews. | Extracting emotions Inside the Opinion. Negative In a Situation  Might Be Considered Positive In Another Situation. |

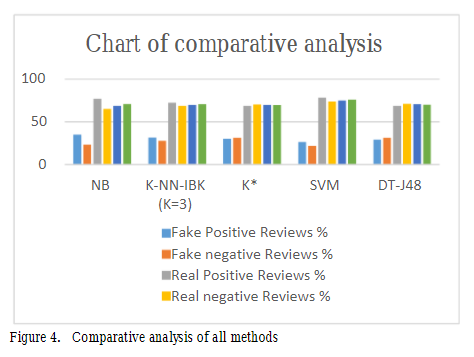
**Literature Survey**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [2] | 2017 | Detect Fake Online Reviews Using Machine Learning. | Length of the review, Average word Length Of The Reviewer  Number of Sentences Percentage of Numerals Percentage of Capitalized Words. | Using User Behavior Data is Better than Using Text Data When It Comes to Detecting Fake Reviews | A.I. Bots Are Better Trained And Now Can Write More Human-Realistic Reviews, Which Make It Even Harder To Catch Fake Reviews |
| [3] | 2020 | Survey on Fake Online Review Using Machine Learning Algorithms. | Experimental methodology. Here the data set has been considered for compression and security purpose. | Random Forest algorithm is much better when compared with several primitive algorithms | Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable |
| [4] | 2021 | Fake Reviews Detection using Supervised Machine Learning. | KNN,SVM, Logistic Regression , Random Forest , Naive Bayes | Logistic Regression Gives Highest Accuracy of 87.87% SVM , Random Forest Have Close Accuracy Of 87.82% | Behavioral Features Are Also Very Important And Cannot Be Ignored As They Have a High Impact On The Performance Of The Fake Review Detection Process. |

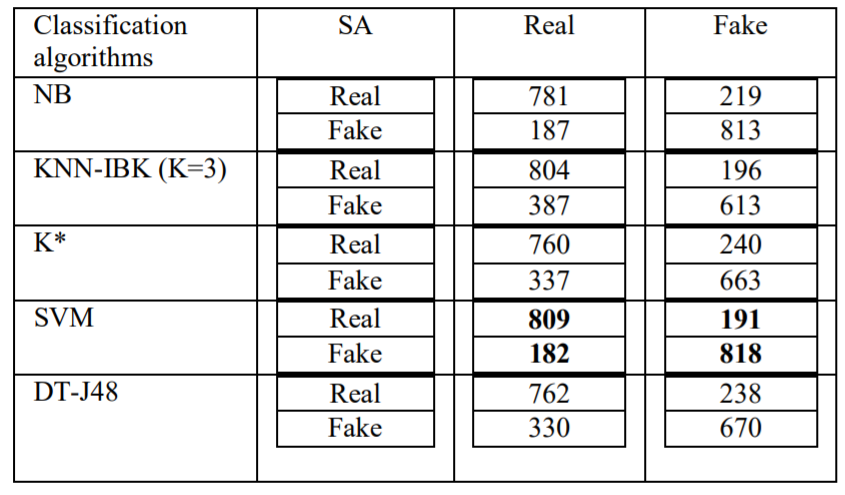
**METHOD and MATERIAL**



**RESULTS DISCUSSION**



**Confusion Matrix**



**CONCLUSION**

* Five supervised learning algorithms to classifying sentiment of our datasets have been compared in this paper: NB, K-NN, K\*, SVM, and DT-J48. Using the accuracy analysis for these five techniques, we found that SVM algorithm is the most accurate for correctly classifying the reviews in movie reviews datasets.

**REFERENCES**

[1]. Ahmed M. Elmogy, Usman Tariq, Atif Ibrahim, (2021) .”Fake Reviews Detection using Supervised Machine Learning”. (IJACSA) International Journal of Advanced Computer Science and Applications ,Vol. 12, No. 1, 2021.

[2]. Elshrif Elmurngi, Abdelouahed Gherbi, (2017) .” Detecting Fake Reviews through Sentiment Analysis Using Machine Learning Techniques “. DATA ANALYTICS 2017 : The Sixth International Conference on Data Analytics.

[3]. Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. “Automated crowd turfing attacks and defense in online review systems”. arxiv preprint arXiv:1708.08151, 2017.

[4]. M. Hu and B. Liu, “Mining and summarizing customer reviews,” . *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* , pp. 168–177, Seattle, WA, USA, August 2020.

[5]. Pilkaka Anusha, Kaki Leela Prasad, (2020) . “Survey on Fake Online Review Using Machine Learning Algorithms”, Journal of Critical Reviews , ISSN- 2394-5125 Vol7 ,Issue 18 , 2020 .