REAL TIME FORGERY ANALYSIS USING NOVEL MULTI LAYER PERCEPTION

D.SUPRIYA
B.MUNI KUMAR
A.KOTIMYTHREYI
K.SATYA SAI VENKAT

GUIDED BY:
DR.TANVEER SARDAR SIR
ASST PROFESSOR
GITAM

CONTENTS

- Problem definition
- Literature Survey
- Step by step process
- Classifiers used
- Dataset
- Methodology
- Advantages
- Future work
- Conclusion

PROBLEM DEFINITION:

Detection of fake reviews out of a massive collection of reviews having various distinct categories like Home and Office, Sports, etc. with each review having a corresponding rating, label i.e. CG(Computer Generated Review) and OR(Original Review generated by humans) and the review text.
Main task is to detect whether a given review is fraudulent or not. If it is computer generated, it is considered fake otherwise not.

× × × ×

LITERATURE SURVEY:

Author NAme,Journal name,Year	**title of the paper**	**inference**	**Research Gap**	**relevance to present work**
Md Mahadi Hassan Sohan, Mohammad Monirujjaman Khan, Rajesh Dey, Ipseeta Nanda **Date of Conference: 06-09 June 2022** **Date Added to IEEE Xplore: 13 July 2022**	**Fake Product Review Detection Using Machine Learning**	**in this paper we learned about the semi supervised learning approach**	**we can use non supervised learning approaches for detecting fake reviews**	**EDA is used to the visual techniques Many classfiers are used but most probably multi layer perceptron**
Nikhil Chandra Sai Ram, Gowtham Vakati, Jagadesh Varma Nadimpalli , Yash Sah , Sai Karthik Datla Publish Date : 2022-05-24 Paper Id : IJRASET43202	**Fake Reviews Detection Using Supervised Machine Learning**	in this paper we learned about how naivie bayes is implemented and also about django frame work	random forest and other approaches can be used for better accuracy	logistic regression is implemented

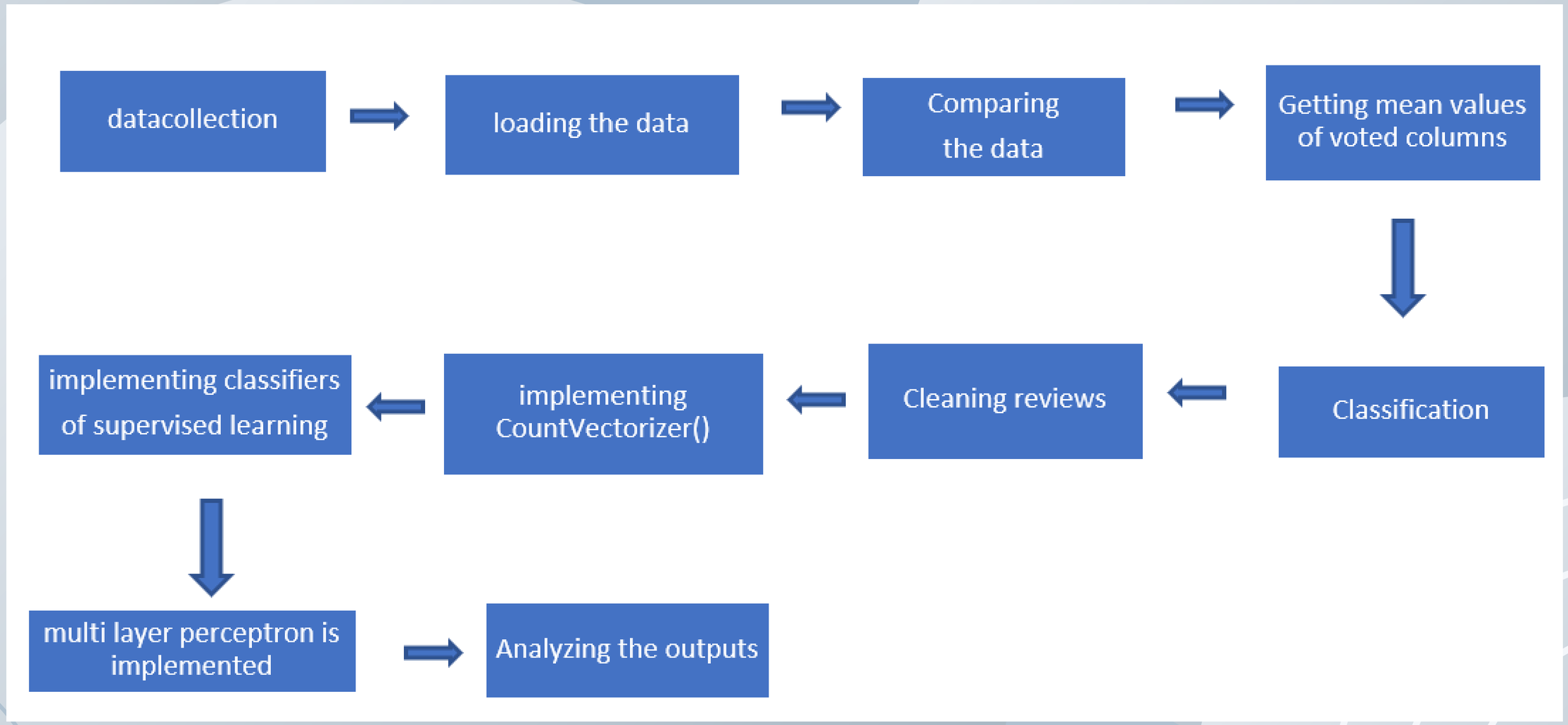
STEP BY STEP PROCESS

- **Input:** Dataset K.
- **Output:** A testing dataset class
- **Step 1:** data set gathering
- **Step 2:** analyzing the data collected
- **Step 3 :** start an approach for data minimization
- **Step 4:** importing all the packages needed
- **Step 5 :** loading the dataset
- **Step 6 :** creating a new column in the data set for the number of words in the review
- **Step 7:** Compare text length to stars
- **Step 8:** Calculating mean vaues of voted columns
- **Step 9:** Classification
- **Step 10:** Splitting the dataset into two parts and removing the punctuation marks.
- **Step 11:** implementing countVectorizer()
- **Step 12:** implementing classifiers.

DATA SET:

We gathered this data set from kaggle
https://www.kaggle.com/code/rkypragada/fake-reveiw-detection/data
this data set consists of different parameters like review id, stars, voted columns which are very important to detect fake reviews

Methodology

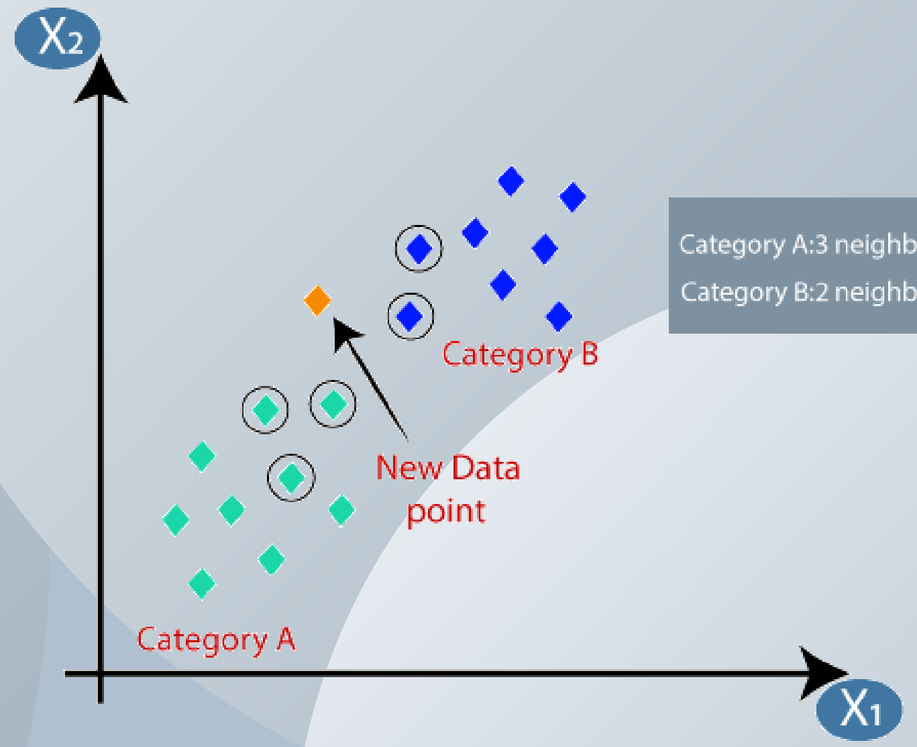


CLASSIFIERS USED

- KNeighbors Classifier
- Support vector machine
- Random forest
- Novel multilayer perceptron
- decision tree classifier
- Gradient boosting

KNEIGHBORS CLASSIFIER:

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

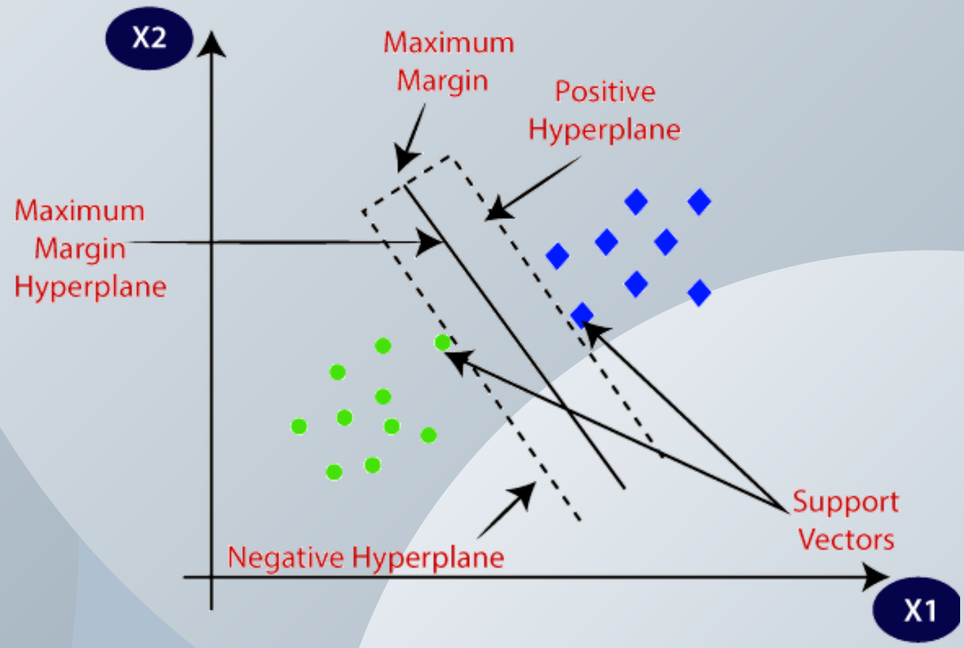


from sklearn.neighbors import KNeighborsClassifier

× × × ×

SUPPORT VECTOR MACHINE:

SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs



from sklearn.svm import SVC

× × × ×

RANDOM FOREST ALGORITHM

Random Forests is a Machine Learning algorithm that tackles one of the biggest problems with Decision Trees: variance. Even though Decision Trees is simple and flexible, it is greedy algorithm. It focuses on optimizing for the node split at hand, rather than taking into account how that split impacts the entire tree.

from sklearn.ensemble import RandomForestClassifier

× × × ×

NOVEL MULTILAYER PERCEPTRON:

This algorithm is one of the machine learning algorithm in which it is used to detect the forgery analysis to predict the future profits *multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one.*

from sklearn.neural_network import MLPClassifier

× × × ×

DECISION TREE CLASSIFIER

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

from sklearn.tree import DecisionTreeClassifier

× × × ×

GRADIENT BOOSTING

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.

from sklearn.ensemble import GradientBoostingClassifier

× × × ×

COUNT VECTORIZER

CountVectorizer is used to transform a corpora of text to a vector of term / token counts. It also provides the capability to preprocess your text data prior to generating the vector representation making it a highly flexible feature representation module for text.

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

OUTPUT EXPLANATION:

We got 3 * 3 confusion matrix as output of executed code. In this matrix rows comes under predicted values and columns comes under actual values. this matrix is divided into three classes as A,B,C respectively. Diagonal elements are correctly classified as they belong to same class and other or misclassified as they are not in same class

	Predicted			
Actual	A	B	C	Total
A	**12**	10	140	162
B	3	**33**	256	292
C	8	12	**636**	656
Total	23	55	1032	1110

OUTPUT EXPLANATION:

Precision:

The quality of a positive prediction made by the model.

$$\frac{correctly\ predicted}{total\ predicted}$$

Recall:

Recall measures the proportion of actual positive labels correctly identified by the model.

$$\frac{Correctly\ classified}{Actual}$$

OUTPUT EXPLANATION:

F1 score:

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.

$$2 * \frac{precision * recall}{precision + recall}$$

weightedavg:

an ensemble machine learning approach that combines the predictions from multiple models, where the contribution of each model is weighted proportionally to its capability or skill.

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

OUTPUT EXPLANATION:

Macro avg:
the arithmetic mean of the individual class related to precision, memory, and f1 score.

Macro average of Precision:

$$\frac{precision\ class\ A + precision\ class B + precision\ class\ c}{3}$$

KNN CALCULATION

K-Nearest neighbour Algorithm:

Confusion matrix:

$$\begin{matrix} 12 & 10 & 140 \\ 3 & 33 & 256 \\ 8 & 12 & 636 \end{matrix}$$

Predicted

Actual	A	B	C	Total
A	**12**	10	140	162
B	3	**33**	256	292
C	8	12	**636**	656
Total	23	55	1032	1110

Precision= $\frac{correctly\ predicted}{total\ predicted}$

Class A: $\frac{12}{23}$ = 0·52 Class B : $\frac{33}{55}$ = 0·60

Class C: $\frac{636}{1032}$ =0·62

Recall:

$$\frac{Correctly\ Classified}{Actual}$$

Class A: $\frac{12}{162}$ = 0·07

Class B: $\frac{33}{292}$ = 0·11

Class C: $\frac{636}{656}$ = 0·97

F1 Score:

$$2*\ \frac{Precision*Recall}{Precision+Recall}$$

Class A: $2*\frac{0.52*0.07}{0.52+0.07}$=0.13

Class B: $2*\frac{0.6*0.11}{0.6+0.11}$=0.19

Class C: $2*\frac{0.62*0.97}{0.62+0.97}$=0.75

Accuracy: $\frac{Total\ correctly\ classified}{Actual}$

$\frac{12+33+636}{1110}$=0.61

KNN CALCULATION

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

$$\frac{162}{1110} * 0.52 + \frac{292}{1110} * 0.6 + \frac{656}{1110} * 0.62$$

0.07+0.157+0.366=0.60

Weighted average of recall:

Actual class A instances *recall of class A+
Actual class B instances *recall of class B+
Actual class C instances *recall of class C

$$\frac{162}{1110} * 0.07 + \frac{292}{1110} * 0.11 + \frac{656}{1110} * 0.97$$

0.01+0.02+0.57=0.61

Weighted average for f1 score:

Actual class A instances *F1 score of class A+
Actual class B instances *F1 score of class B+
Actual class C instances *F1 score of class C

$$\frac{162}{1110} * 0.13 + \frac{292}{1110} * 0.19 + \frac{656}{1110} * 0.75 = 0.51$$

Macro average of
Precision:

$$\frac{precision\ class\ A + precision\ classB + precision\ class\ c}{3}$$

$$\frac{0.52+0.60+0.62}{3} = 0.58$$

Macro average of recall:

$$\frac{recall\ class\ A + recall\ classB + recall\ class\ c}{3}$$

$$\frac{0.07+0.11+0.97}{3} = 0.39$$

Macro average of F1 score:

$$\frac{f1\ score\ class\ A + f1\ score\ classB + f1\ score\ class\ c}{3}$$

$$\frac{0.13+0.19+0.75}{3} = 0.36$$

KNN OUTPUT

```
# K Nearest Neighbour Algorithm
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(x_train,y_train)
predknn = knn.predict(x_test)
print("Confusion Matrix for K Neighbors Classifier:")
print(confusion_matrix(y_test,predknn))
print("Score: ",round(accuracy_score(y_test,predknn)*100,2))
print("Classification Report:")
print(classification_report(y_test,predknn))
```

```
Confusion Matrix for K Neighbors Classifier:
[[ 12  10 140]
 [  3  33 256]
 [  8  12 636]]
Score:  61.35
Classification Report:
              precision    recall  f1-score   support

           1       0.52      0.07      0.13       162
           3       0.60      0.11      0.19       292
           5       0.62      0.97      0.75       656

    accuracy                           0.61      1110
   macro avg       0.58      0.39      0.36      1110
weighted avg       0.60      0.61      0.51      1110
```

SVM CALCULATION

Support vector machine:

Confusion matrix:

$$\begin{matrix} 31 & 23 & 108 \\ 5 & 122 & 165 \\ 1 & 19 & 636 \end{matrix}$$

Actual	A	B	C	Total
A	**31**	23	108	162
B	5	**122**	165	292
C	1	19	**636**	656
Total	37	164	909	1110

Precision= $\frac{correctly\ predicted}{total\ predicted}$

Class A: $\frac{31}{37}$ = 0.84 Class B : $\frac{122}{164}$ = 0.74

Class C: $\frac{636}{909}$ =0.70

Recall:

$$\frac{Correctly\ Classified}{Actual}$$

Class A: $\frac{31}{162}$ = 0.19 Class B: $\frac{122}{292}$ = 0.42

Class C: $\frac{636}{656}$ = 0.97

F1 Score:

$$2* \frac{Precision*Recall}{Precision+Recall}$$

Class A: $2*\frac{0.84*0.19}{0.84+0.19}$=0.31

Class B: $2*\frac{0.74*0.42}{0.74+0.42}$=0.54

Class C: $2*\frac{0.70*0.97}{0.70+0.97}$=0.81

Accuracy: $\frac{Total\ correctly\ classified}{Actual}$

$\frac{31+122+636}{1110}$=0.71

SVM CALCULATION

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

$$\frac{162}{1110} * 0.84 + \frac{292}{1110} * 0.74 + \frac{656}{1110} * 0.70$$

0.122+0.194+0.413=0.73

Weighted average of recall:

Actual class A instances *recall of class A+
Actual class B instances *recall of class B+
Actual class C instances *recall of class C

$$\frac{162}{1110} * 0.19 + \frac{292}{1110} * 0.42 + \frac{656}{1110} * 0.97$$

0.02+0.110+0.573=0.71

Weighted average for f1 score:

Actual class A instances *F1 score of class A+
Actual class B instances *F1 score of class B+
Actual class C instances *F1 score of class C

$$\frac{162}{1110} * 0.31 + \frac{292}{1110} * 0.54 + \frac{656}{1110} * 0.81 = 0.67$$

Macro average of
Precision:

$$\frac{precision\ class\ A + precision\ classB + precision\ class\ c}{3}$$

$$\frac{0.84+0.74+0.70}{3} = 0.76$$

Macro average of recall:

$$\frac{recall\ class\ A + recall\ classB + recall\ class\ c}{3}$$

$$\frac{0.19+0.42+0.97}{3} = 0.53$$

Macro average of F1 score:

$$\frac{f1\ score\ class\ A + f1\ score\ classB + f1\ score\ class\ c}{3}$$

$$\frac{0.31+0.54+0.81}{3} = 0.55$$

SVM OUTPUT

```
# Support Vector Machine
from sklearn.svm import SVC
svm = SVC(random_state=101)
svm.fit(x_train,y_train)
predsvm = svm.predict(x_test)
print("Confusion Matrix for Support Vector Machines:")
print(confusion_matrix(y_test,predsvm))
print("Score:",round(accuracy_score(y_test,predsvm)*100,2))
print("Classification Report:",classification_report(y_test,predsvm))
```

```
Confusion Matrix for Support Vector Machines:
[[ 31  23 108]
 [  5 122 165]
 [  1  19 636]]
Score: 71.08
Classification Report:                precision    recall  f1-score   support

           1       0.84      0.19      0.31       162
           3       0.74      0.42      0.54       292
           5       0.70      0.97      0.81       656

    accuracy                           0.71      1110
   macro avg       0.76      0.53      0.55      1110
weighted avg       0.73      0.71      0.67      1110
```

RANDOM FOREST CALCULATION

Random forest algorithm:

Confusion matrix:

$$\begin{matrix} 33 & 34 & 95 \\ 3 & 92 & 197 \\ 1 & 14 & 641 \end{matrix}$$

Actual	A	B	C	Total
A	**33**	34	95	162
B	3	**92**	197	292
C	1	14	**641**	656
Total	37	140	933	1110

Precision= $\frac{correctly\ predicted}{total\ predicted}$

Class A: $\frac{33}{37}$ = 0.89 Class B : $\frac{92}{140}$ = 0.66

Class C: $\frac{641}{933}$ =0.69

Recall:

$$\frac{Correctly\ Classified}{Actual}$$

Class A: $\frac{33}{162}$ = 0.20 Class B: $\frac{92}{292}$ = 0.32

Class C: $\frac{641}{656}$ = 0.98

F1 Score:

$$2*\ \frac{Precision*Recall}{Precision+Recall}$$

Class A: $2*\frac{0.89*0.20}{0.89+0.20}$=0.33

Class B: $2*\frac{0.66*0.32}{0.66+0.32}$=0.43

Class C: $2*\frac{0.69*0.98}{0.69+0.98}$=0.64

Accuracy: $\frac{Total\ correctly\ classified}{Actual}$

$\frac{33+92+641}{1110}$=0.69

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

RANDOM FOREST CALCULATION

$$\frac{162}{1110} * 0.89 + \frac{292}{1110} * 0.66 + \frac{656}{1110} * 0.69$$

0.129+0.173+0.407=0.71

Weighted average of recall:

Actual class A instances *recall of class A+
Actual class B instances *recall of class B+
Actual class C instances *recall of class C

$$\frac{162}{1110} * 0.20 + \frac{292}{1110} * 0.32 + \frac{656}{1110} * 0.98$$

0.02+0.084+0.579=0.69

Weighted average for f1 score:

Actual class A instances *F1 score of class A+
Actual class B instances *F1 score of class B+
Actual class C instances *F1 score of class C

$$\frac{162}{1110} * 0.33 + \frac{292}{1110} * 0.43 + \frac{656}{1110} * 0.81$$

0.048+0.113+0.478=0.64

Macro average of
Precision:

$$\frac{precision\ class\ A + precision\ classB + precision\ class\ c}{3}$$

$\frac{0.89+0.66+0.69}{3}$=0.75

Macro average of recall:

$$\frac{recall\ class\ A + recall\ classB + recall\ class\ c}{3}$$

$\frac{0.20+0.32+0.98}{3}$=0.50

Macro average of F1 score:

$$\frac{f1\ score\ class\ A + f1\ score\ classB + f1\ score\ class\ c}{3}$$

$\frac{0.33+0.43+0.81}{3}$ =0.52

RANDOM FOREST OUTPUT

```
#random forest
from sklearn.ensemble import RandomForestClassifier
rmfr = RandomForestClassifier()
rmfr.fit(x_train,y_train)
predrmfr = rmfr.predict(x_test)
print("Confusion Matrix for Random Forest Classifier:")
print(confusion_matrix(y_test,predrmfr))
```

ps://colab.research.google.com/drive/1gWUKVGSDyH68bZVO1mysDczHwlQfc5q1#scrollTo=kFl872dovk_S&printMode=true

4/23, 1:55 PM major project code.ipynb - Colaboratory

```
print("Score:",round(accuracy_score(y_test,predrmfr)*100,2))
print("Classification Report:",classification_report(y_test,predrmfr))
```

```
Confusion Matrix for Random Forest Classifier:
[[ 30  29 103]
 [  3 100 189]
 [  1  22 633]]
Score: 68.74
Classification Report:
```

	precision	recall	f1-score	support
1	0.88	0.19	0.31	162
3	0.66	0.34	0.45	292
5	0.68	0.96	0.80	656
accuracy			0.69	1110
macro avg	0.74	0.50	0.52	1110
weighted avg	0.71	0.69	0.64	1110

DECISION TREE CALUCATION

Decision tree classifier:

Confusion matrix:

$$\begin{matrix} 60 & 48 & 54 \\ 37 & 139 & 116 \\ 44 & 110 & 502 \end{matrix}$$

Actual	A	B	C	Total
A	**60**	48	54	162
B	37	**139**	116	292
C	44	110	**502**	656
Total	141	297	672	1110

Precision= $\frac{correctly\ predicted}{total\ predicted}$

Class A: $\frac{60}{141}$ = 0.43 Class B : $\frac{139}{297}$ = 0.47

Class C: $\frac{502}{672}$ =0.75

Recall:

$$\frac{Correctly\ Classified}{Actual}$$

Class A: $\frac{60}{162}$ = 0.37 Class B: $\frac{139}{292}$ = 0.48

Class C: $\frac{502}{656}$ = 0.77

F1 Score:

$$2*\ \frac{Precision*Recall}{Precision+Recall}$$

Class A: $2*\frac{0.43*0.37}{0.43+0.37}$=0.40

Class B: $2*\frac{0.47*0.48}{0.47+0.48}$=0.47

Class C: $2*\frac{0.75*0.77}{0.75+0.77}$=0.76

Accuracy: $\frac{Total\ correctly\ classified}{Actual}$

$\frac{60+139+502}{1110}$=0.63

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

DECISION TREE CALUCATION

$$\frac{162}{1110} * 0.43 + \frac{292}{1110} * 0.47 + \frac{656}{1110} * 0.75$$

0.062+0.123+0.443=0.63

Weighted average of recall:

Actual class A instances *recall of class A+
Actual class B instances *recall of class B+
Actual class C instances *recall of class C

$$\frac{162}{1110} * 0.37 + \frac{292}{1110} * 0.48 + \frac{656}{1110} * 0.77$$

0.054+0.126+0.455=0.63

Weighted average for f1 score:

Actual class A instances *F1 score of class A+
Actual class B instances *F1 score of class B+
Actual class C instances *F1 score of class C

$$\frac{162}{1110} * 0.40 + \frac{292}{1110} * 0.47 + \frac{656}{1110} * 0.76$$

0.058+0.123+0.449=0.63

Macro average of
Precision:

$$\frac{precision\ class\ A + precision\ classB + precision\ class\ c}{3}$$

$\frac{0.43+0.47+0.75}{3}$=0.55

Macro average of recall:

$$\frac{recall\ class\ A + recall\ classB + recall\ class\ c}{3}$$

$\frac{0.37+0.48+0.77}{3}$=0.54

Macro average of F1 score:

$$\frac{f1\ score\ class\ A + f1\ score\ classB + f1\ score\ class\ c}{3}$$

$\frac{0.40+0.47+0.76}{3}$ =0.54

DECISION TREE OUTPUT

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
preddt = dt.predict(x_test)
print("Confusion Matrix for Decision Tree:")
print(confusion_matrix(y_test,preddt))
print("Score:",round(accuracy_score(y_test,preddt)*100,2))
print("Classification Report:",classification_report(y_test,preddt))
```

```
Confusion Matrix for Decision Tree:
[[ 60  48  54]
 [ 37 139 116]
 [ 44 110 502]]
Score: 63.15
Classification Report:                precision    recall  f1-score   support

           1       0.43      0.37      0.40       162
           3       0.47      0.48      0.47       292
           5       0.75      0.77      0.76       656

    accuracy                           0.63      1110
   macro avg       0.55      0.54      0.54      1110
weighted avg       0.63      0.63      0.63      1110
```

GRADIENT BOOSTING CALUCATION

Gradient Boosting:

Confusion matrix:

$$\begin{matrix} 61 & 33 & 68 \\ 9 & 139 & 144 \\ 4 & 3 & 619 \end{matrix}$$

Actual	A	B	C	Total
A	**61**	33	68	162
B	9	**139**	144	292
C	4	33	**619**	656
Total	74	205	831	1110

Precision= $\frac{correctly\ predicted}{total\ predicted}$

Class A: $\frac{61}{74}$ = 0.82 Class B : $\frac{139}{205}$ = 0.68

Class C: $\frac{619}{831}$ =0.74

Recall:

$$\frac{Correctly\ Classified}{Actual}$$

Class A: $\frac{61}{162}$ = 0.38 Class B: $\frac{139}{292}$ = 0.48

Class C: $\frac{619}{656}$ = 0.94

F1 Score:

$$2*\frac{Precision*Recall}{Precision+Recall}$$

Class A: $2*\frac{0.82*0.38}{0.82+0.38}$=0.52

Class B: $2*\frac{0.68*0.48}{0.68+0.48}$=0.56

Class C: $2*\frac{0.74*0.94}{0.74+0.94}$=0.83

Accuracy: $\frac{Total\ correctly\ classified}{Actual}$

GRADIENT BOOSTING CALUCATION

$\frac{61+139+619}{1110}$=0.74

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

$$\frac{162}{1110} * 0.82 + \frac{292}{1110} * 0.68 + \frac{656}{1110} * 0.74$$

0.119+0.178+0.437=0.74

Weighted average of recall:

Actual class A instances *recall of class A+
Actual class B instances *recall of class B+
Actual class C instances *recall of class C

$$\frac{162}{1110} * 0.38 + \frac{292}{1110} * 0.48 + \frac{656}{1110} * 0.94$$

0.055+0.126+0.555=0.74

Weighted average for f1 score:

Actual class A instances *F1 score of class A+
Actual class B instances *F1 score of class B+
Actual class C instances *F1 score of class C

$$\frac{162}{1110} * 0.52 + \frac{292}{1110} * 0.56 + \frac{656}{1110} * 0.83$$

0.075+0.147+0.490=0.71

Macro average of
Precision:

$$\frac{precision\ class\ A+precision\ classB+precision\ class\ c}{3}$$

$\frac{0.82+0.68+0.74}{3}$=0.75

Macro average of recall:

$$\frac{recall\ class\ A + recall\ classB + recall\ class\ c}{3}$$

$\frac{0.38+0.48+0.94}{3}$=0.60

Macro average of F1 score:

$\frac{f1\ score\ class\ A+f1\ score\ classB+f1\ score\ class\ c}{3}$=

$\frac{0.52+0.56+0.83}{3}$ =0.64

GRADIENT BOOSTING OUTPUT

```
#Gradient Boosting Classifier
from sklearn.ensemble import GradientBoostingClassifier
"""# parameter evaluation
gbe = GradientBoostingClassifier(random_state=0)
parameters = {
     'learning_rate': [0.05, 0.1, 0.5],
    'max_features': [0.5, 1],
    'max_depth': [3, 4, 5]}
gridsearch=GridSearchCV(gbe,parameters,cv=100,scoring='roc_auc')
```

https://colab.research.google.com/drive/1gWUKVGSDyH68bZVO1mysDczHwIQfc5q1#scrollTo=kFl872dovk_S&printMode=true

3/4/23, 1:55 PM major project code.ipynb - Colaboratory

```
gridsearch.fit(x,y)
print(gridsearch.best_params_)
print(gridsearch.best_score_)"""
#Boosting
gbi = GradientBoostingClassifier(learning_rate=0.1,max_depth=5,max_features=0.5,random_state=999999)
gbi.fit(x_train,y_train)
predgbi = gbi.predict(x_test)
print("Confusion Matrix for Gradient Boosting Classifier:")
print(confusion_matrix(y_test,predgbi))
print("Score:",round(accuracy_score(y_test,predgbi)*100,2))
print("Classification Report:",classification_report(y_test,predgbi))
```

```
Confusion Matrix for Gradient Boosting Classifier:
[[ 61  33  68]
 [  9 139 144]
 [  4  33 619]]
Score: 73.78
Classification Report:              precision    recall  f1-score   support

           1       0.82      0.38      0.52       162
           3       0.68      0.48      0.56       292
           5       0.74      0.94      0.83       656

    accuracy                           0.74      1110
   macro avg       0.75      0.60      0.64      1110
weighted avg       0.74      0.74      0.71      1110
```

MULTI-LAYER PERCEPTRON

Multi-layer perceptron:

Confusion matrix:

$$\begin{matrix} 95 & 36 & 31 \\ 24 & 184 & 84 \\ 12 & 63 & 581 \end{matrix}$$

Actual	A	B	C	Total
A	**95**	36	31	162
B	24	**184**	84	292
C	12	63	**581**	656
Total	131	283	696	1110

Precision= $\frac{correctly\ predicted}{total\ predicted}$

Class A: $\frac{95}{131}$ = 0.73 Class B : $\frac{184}{283}$ = 0.65

Class C: $\frac{581}{696}$ =0.83

Recall:

$$\frac{Correctly\ Classified}{Actual}$$

Class A: $\frac{95}{162}$ = 0.59 Class B: $\frac{184}{292}$ = 0.63

Class C: $\frac{581}{656}$ = 0.89

F1 Score:

$$2*\ \frac{Precision*Recall}{Precision+Recall}$$

Class A: $2*\frac{0.73*0.59}{0.73+0.59}$=0.65

Class B: $2*\frac{0.65*0.63}{0.65+0.63}$=0.64

Class C: $2*\frac{0.83*0.89}{0.83+0.89}$=0.86

Accuracy: $\frac{Total\ correctly\ classified}{Actual}$

$\frac{95+184+581}{1110}$=0.77

Weighted average of precision:

Actual class A instances *precision of class A+
Actual class B instances *precision of class B+
Actual class C instances *precision of class C

MULTI-LAYER PERCEPTRON

$$\frac{162}{1110} * 0.73 + \frac{292}{1110} * 0.65 + \frac{656}{1110} * 0.83$$

0.106+0.170+0.490=0.77

Weighted average of recall:

Actual class A instances *recall of class A+
Actual class B instances *recall of class B+
Actual class C instances *recall of class C

$$\frac{162}{1110} * 0.59 + \frac{292}{1110} * 0.63 + \frac{656}{1110} * 0.89$$

0.086+0.165+0.525=0.77

Weighted average for f1 score:

Actual class A instances *F1 score of class A+
Actual class B instances *F1 score of class B+
Actual class C instances *F1 score of class C

$$\frac{162}{1110} * 0.65 + \frac{292}{1110} * 0.64 + \frac{656}{1110} * 0.86$$

0.094+0.168+0.508=0.77

Macro average of
Precision:

$$\frac{precision\ class\ A + precision\ classB + precision\ class\ c}{3}$$

$\frac{0.73+0.65+0.83}{3}$=0.74

Macro average of recall:

$$\frac{recall\ class\ A + recall\ classB + recall\ class\ c}{3}$$

$\frac{0.59+0.63+0.89}{3}$=0.70

Macro average of F1 score:

$$\frac{f1\ score\ class\ A + f1\ score\ classB + f1\ score\ class}{3}$$

$\frac{0.65+0.64+0.86}{3}$ =0.72

MULTI-LAYER PERCEPTRON

```
# MULTILAYER PERCEPTRON CLASSIFIER
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier()
mlp.fit(x_train,y_train)
predmlp = mlp.predict(x_test)
print("Confusion Matrix for Multilayer Perceptron Classifier:")
print(confusion_matrix(y_test,predmlp))
print("Score:",round(accuracy_score(y_test,predmlp)*100,2))
print("Classification Report:")
print(classification_report(y_test,predmlp))
```

```
Confusion Matrix for Multilayer Perceptron Classifier:
[[ 95  36  31]
 [ 24 184  84]
 [ 12  63 581]]
Score: 77.48
Classification Report:
              precision    recall  f1-score   support

           1       0.73      0.59      0.65       162
           3       0.65      0.63      0.64       292
           5       0.83      0.89      0.86       656

    accuracy                           0.77      1110
   macro avg       0.74      0.70      0.72      1110
weighted avg       0.77      0.77      0.77      1110
```

OVERALL DATA OF ALL CLASSIFIERS

KNN :

	Precision	recall	F1 score
Class A	0.52	0.07	0.13
Class B	0.60	0.11	0.19
Class C	0.62	0.97	0.75
Weighted Avg	0.60	0.61	0.51
Macro Avg	0.58	0.39	0.36

SVM:

	Precision	recall	F1 score
Class A	0.84	0.19	0.31
Class B	0.74	0.42	0.54
Class C	0.70	0.97	0.81
Weighted Avg	0.73	0.71	0.67
Macro Avg	0.76	0.53	0.55

OVERALL DATA OF ALL CLASSIFIERS

Random Forest:

	Precision	recall	F1 score
Class A	0.88	0.19	0.31
Class B	0.66	0.34	0.45
Class C	0.68	0.96	0.80
Weighted Avg	0.71	0.69	0.64
Macro Avg	0.74	0.50	0.52

Decision Tree:

	Precision	recall	F1 score
Class A	0.43	0.37	0.40
Class B	0.47	0.48	0.47
Class C	0.75	0.77	0.76
Weighted Avg	0.63	0.63	0.63
Macro Avg	0.55	0.54	0.54

OVERALL DATA OF ALL CLASSIFIERS

Gradient Boosting :

	Precision	recall	F1 score
Class A	0.82	0.38	0.52
Class B	0.68	0.48	0.56
Class C	0.74	0.94	0.83
Weighted Avg	0.74	0.74	0.71
Macro Avg	0.75	0.60	0.64

Multi Layer Perceptron:

	Precision	recall	F1 score
Class A	0.73	0.59	0.65
Class B	0.65	0.63	0.64
Class C	0.83	0.89	0.86
Weighted Avg	0.77	0.77	0.77
Macro Avg	0.74	0.70	0.72

ACCURACY OF ALL CLASSIFIERS:

Classifier name	Accuracy
knn	61
svm	71
Random forest	69
decision tree	63
gradient boosting	74
novel multi layer perceptron	77

ADVANTAGES

- Increased accuracy
- Better prediction
- Cost effective
- Better classification
- Easily scalable

DISADVANTAGE

- The drawback of this method is, some process need to be optimized, so it can detect a fake review in a short amount of time.

FUTURE WORK:

- To use a real time/ time based datasets which will allow us to compare the user’s timestamps of the reviews to find if a certain user is posting too many reviews in a short period of time.
- To use and compare other machine learning algorithms like logistic regression to extend the research to deep learning techniques.
- To build real time models with the help of django frame work along with deep learning

× × × ×

CONCLUSION

We have created a model which is used for fake review detection using deep learning approach along with some supervised learning techniques like knn,svm etc.
we got better accuracy for novel multilayer perceptron

× × × ×

Thank You

× × × ×