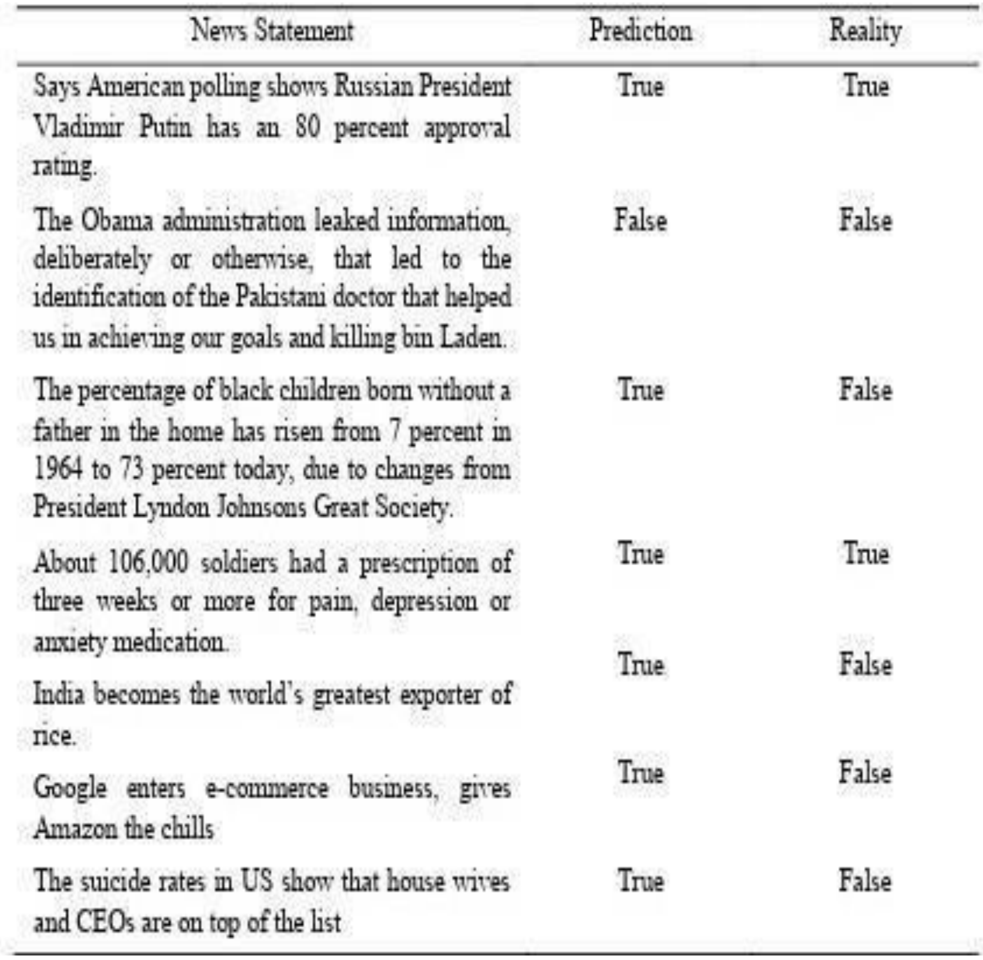
FAKE NEWS DETECTION USING NLP

DATE	**26 oct 2023**
TEAM ID	**394**
PROJECT NAME	**Fake news detection using NLP**

TEST CASES FOR NEWS :



News Statement	Prediction	Reality
Says American polling shows Russian President Vladimir Putin has an 80 percent approval rating.	True	True
The Obama administration leaked information, deliberately or otherwise, that led to the identification of the Pakistani doctor that helped us in achieving our goals and killing bin Laden.	False	False
The percentage of black children born without a father in the home has risen from 7 percent in 1964 to 73 percent today, due to changes from President Lyndon Johnsons Great Society.	True	False
About 106,000 soldiers had a prescription of three weeks or more for pain, depression or anxiety medication.	True	True
India becomes the world's greatest exporter of rice.	True	False
Google enters e-commerce business, gives Amazon the chills	True	False
The suicide rates in US show that house wives and CEOs are on top of the list	True	False

PROGRAM :

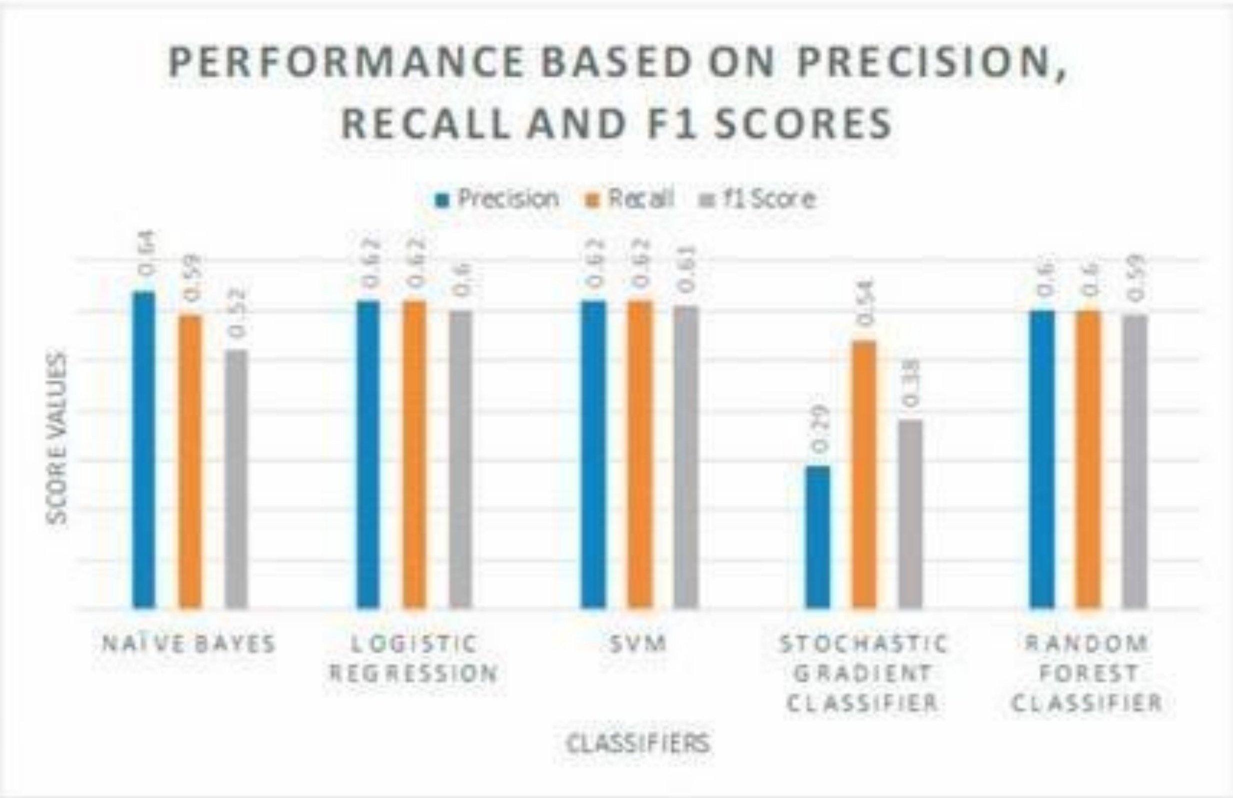
```
import pandas as pd
import matplotlib.pyplot as plt
import spacy
from spacy.util import minibatch, compounding
import random

nlp = spacy.load('el__core__news__md')
df1 = pd.read__csv('../data/jtp__fake__news.csv')
df1.replace(to__replace='[ \ n \ r \ t]', value=' ', regex=True,
                                                inplace=True)
def load__data(train__data, limit=0, split=0.8):
    random.shuffle(train__data)
    train__data = train__data[-limit:]
    texts, labels = zip(*train__data)
     cats = [{"REAL": not bool(y), "FAKE": bool(y)} for y in l
                                                            abels]
    split = int(len(train__data) * split)

    return (texts[:split], cats[:split]), (texts[split:], cats[split:])
# - - - - - - - - - - - - - - - - - - evaluate function defined
                                        below- - - - - - - - - - - -
def evaluate(tokenizer, textcat, texts, cats):
    docs = (tokenizer(text) for text in texts)
    tp = 0.0 # True positives
    fp = 1e-8 # False positives
    fn = 1e-8 # False negatives
    tn = 0.0 # True negatives
    for i, doc in enumerate(textcat.pipe(docs)):
        gold = cats[i]
        for the label, score in doc.cats.items():
            if the label is not in gold:
                continue
            if label = = "FAKE":
                continue
            if score > = 0.5 and gold[label] > = 0.5:
                tp + = 1.0
            elif score > = 0.5 and gold[label] < 0.5:
                fp + = 1.0
            elif score < 0.5 and gold[label] < 0.5:
                tn + = 1
            elif score < 0.5 and gold[label] > = 0.5:
                fn + = 1
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
#- - - - - - - - - - - -if conditions for precision recall - - - - - - - - -
    if (precision + recall) = = 0:
        f__score = 0.0
    else:
        f__score = 2 * (precision * recall) / (precision + recall)
        return {"textcat__p": precision, "textcat__r": recall,
"textcat__f": f__score}

    In [3]:

    df1.info()

    <class 'pandas.core.frame.DataFrame'>

    RangeIndex: 100 entries, 0 to 99

    Data columns (total five columns):

    #   Column   Non-Null Count  Dtype

    --  -   - - - - - -     - - - - - - - - - - - - - - - - - - -

    0   title    100 non-null    object

    One text     100 non-nullobject

    Two sources 100 non-null    object

    Three url      100 non-null    object

    4   is__fake  100 non-null    int64

    dtypes: int64(1), object(4)

    memory usage: 4.0+ KB

    textcat=nlp.create__pipe( "textcat",
config={"exclusive__classes": True, "architecture":
"simple__cnn"})

    nlp.add__pipe(textcat, last=True)

    nlp.pipe__names

    ['tagger', 'parser', 'ner', 'textcat']

    textcat.add__label("REAL")

    textcat.add__label("FAKE")

    df1['tuples'] = df1.apply(lambda row: (row['text'],
row['is__fake']), axis=1)

    train = df1['tuples'].tolist()
    (train__texts, train__cats), (dev__texts, dev__cats) = load__data(train, split=0.9)


    train__data = list(zip(train__texts,[{'cats': cats} for cats in train__cats]))

    n__iter = 20

    # - - - - - - - - - - - - Disabling other components- - - - - - - - - - - - -

    other__pipes = [pipe for pipe in nlp.pipe__names if pipe != 'textcat']

    with nlp.disable__pipes(*other__pipes):  # only train textcat

        optimizer = nlp.begin__training()


        print("Training the model...")

        print('{:^5}\t{:^5}\t{:^5}\t{:^5}'.format('LOSS', 'P', 'R', 'F'))
```

OUTPUT:

```
    array([1716, 1722, 122, 363, 311, 322, 236, 228, 220,
226, 223, 220, 206, 202, 283, 282, 280, 278, 275, 266, 266,
261, 262, 256, 255, 253, 252, 215, 211, 213, 237, 233, 232,
232, 230, 226, 228, 225, 221, 223, 222, 222, 220, 226, 228,
227, 226, 221, 222, 220, 206, 208, 206, 205, 201, 203, 202,
202, 200, 66, 68, 67, 66, 65, 61, 63, 62, 60, 86, 88, 87, 86, 81,
83, 82, 76, 78, 77, 76, 75, 71, 73, 72, 72, 70, 66, 68, 67, 66, 65,
61, 63, 62, 62, 60, 56, 58, 57, 56, 55, 51, 53, 52, 52, 50, 16, 18,
17, 16, 15, 11, 13, 12, 12, 10, 36, 38, 37, 36, 35, 31, 33, 32, 32,
```

30, 26, 28, 27, 26, 25, 21, 23, 22, 221, 223, 222, 222, 220, 226, 228, 227, 226, 221, 222, 220, 206, 208, , 280, 278, 275, 266, 266, 261, 262, 256, 255, 253, 252, 215, 211, 213, 237, 233, 232, 232, 230, 226, 228, 225, 221, 223, 222, 222, 220, 226, 228, 227, 226, 221, 222, 206, 205, 201, 203, 202, 202, 200, 66, 68, 67, 66, 65, 61, 63, 62, 60, 86, 88, 87, 86, 81, 83, 82, 76, 78, 77, 76, 22, 20, 26, 28, 27, 26, 25, 21, 23, 22, 22, 20, 6, 8, 7, 6, 5, 1, 3, 2, 2])

PERFORMANCE GRAPHS OF CLASSIFIERS :



REFERENCES :

1• ShaoC. Ciampaglia . . V arol . lamminiA . encer . (2023). The spread o a e ne s by socialbots. arXiv preprint arXiv:1707.075929 6-104

2• unt E. (2023). hat is a ne s o to spot it and hat you can do to stop it. The uardian 17.

3• Shu . S liva A. ang S. Tang . iu . (2023). a e ne s detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter 19(1) 22-36.

4• uchans N. Seo S. iu Y. (2017 November). Csi: A hybrid deep model or a ne s detection. n Proceedings of the 2023 ACM on Conference on Information and Knowledge Management (pp. 797-806). AC

5• Vol ovaS . S ha er . ang . Y. odas N. (2023 july). Separating acts romiction: inguistic models to classiy suspicious and trusted ne s posts on titter. n Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 647-653).

6• ang . Y. (2023). liar liar pants on ire : A ne benchmar dataset or a ne s detection. arXiv preprint arXiv:1705.00648.

7• eis . C. CorreiaA . urai . V elosoA. B enevenuto . Cambria E. (2023). Supervised earning or a e Ne s Detection. EEE ntelligent Systems 34(2) 76-81.

8• PalS . umar T. S. PalS . (2023). Applying achine earning to Detect a e Ne s. ndian ournal o Computer Science4(1)