Objective: The main objective is to detect the fake news, build a Machine Learning model to differentiate between “Real” news and “Fake” news.

Strategies used:

The initial focus was to create a training dataset which is devoid of any redundancy and is such that computation reduces. For that all the stop words are removed from all the text. Lemmatizing the data removed all the unwanted characters without sacrificing the meaning. TF-IDF method was used for vectorisation so that semantic quality of words is maintained leading to better dataset.

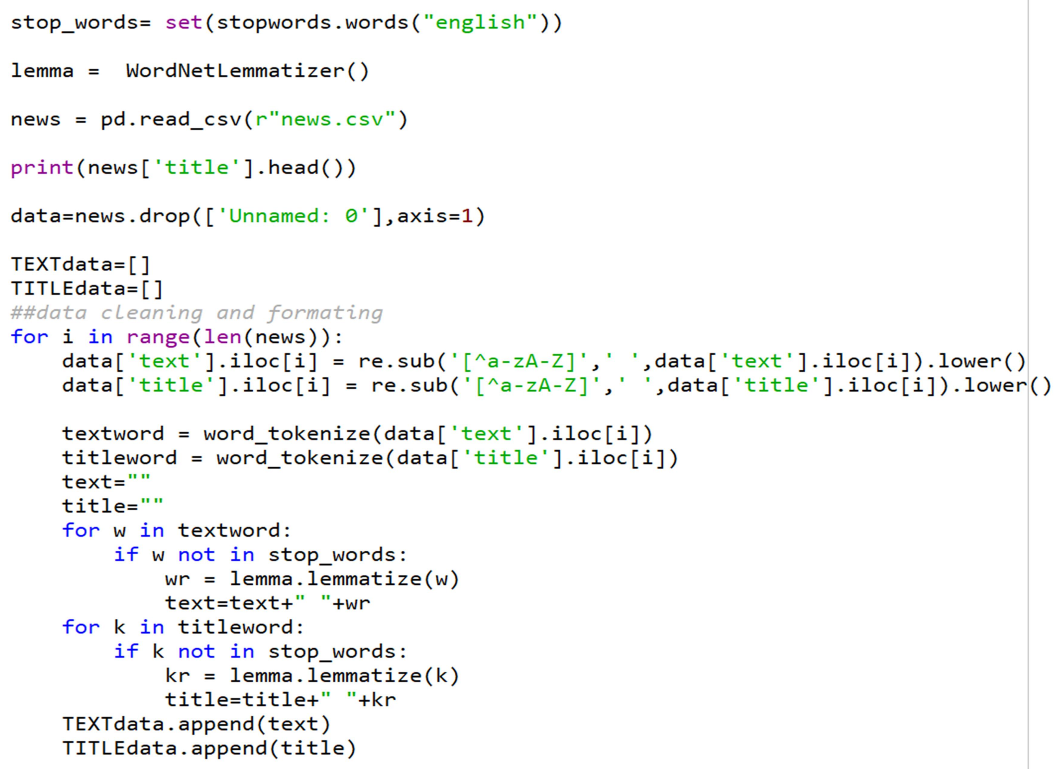
For the model a deep learning model was chosen in order to get better accuracy. It has three hidden layers with 1024, 512, 1024 nodes respectively. All the layer was batch normalised and relu was used as the activation function except for the output layer where sigmoid was used. The batch normalisation not only standardises the data but also reduces the training time required by the model to converge. Binary cross-entropy loss was chosen as loss function and Adam algorithm as an optimiser.

Codes:

1. Importing libraries

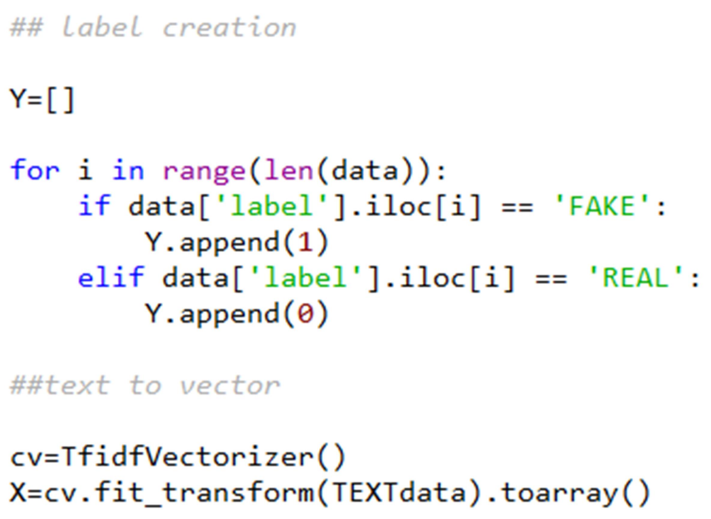
```
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import rcParams
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re
import tensorflow as tf
import keras
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report
```

2. Data cleaning and preparation



```
stop_words= set(stopwords.words("english"))

lemma =  WordNetLemmatizer()

news = pd.read_csv(r"news.csv")

print(news['title'].head())

data=news.drop(['Unnamed: 0'],axis=1)

TEXTdata=[]
TITLEdata=[]
##data cleaning and formating
for i in range(len(news)):
    data['text'].iloc[i] = re.sub('[^a-zA-Z]',' ',data['text'].iloc[i]).lower()
    data['title'].iloc[i] = re.sub('[^a-zA-Z]',' ',data['title'].iloc[i]).lower()

    textword = word_tokenize(data['text'].iloc[i])
    titleword = word_tokenize(data['title'].iloc[i])
    text=""
    title=""
    for w in textword:
        if w not in stop_words:
            wr = lemma.lemmatize(w)
            text=text+" "+wr
    for k in titleword:
        if k not in stop_words:
            kr = lemma.lemmatize(k)
            title=title+" "+kr
    TEXTdata.append(text)
    TITLEdata.append(title)
```

3. Vectorisation of data to produce training data and labels



```
## label creation

Y=[]

for i in range(len(data)):
    if data['label'].iloc[i] == 'FAKE':
        Y.append(1)
    elif data['label'].iloc[i] == 'REAL':
        Y.append(0)

##text to vector

cv=TfidfVectorizer()
X=cv.fit_transform(TEXTdata).toarray()
```

4. Splitting data into train and test data into 80:20 ratio

Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,Y,test_size=0.2)

5. The deep learning classification model

```
##model
model=tf.keras.Sequential()
model.add(tf.keras.layers.Dense(1024,input_shape=(X.shape[1],),activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.1))
model.add(tf.keras.layers.Dense(512,input_shape=(1024,),activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Dense(1024,input_shape=(512,),activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(1,input_shape=(1024,),activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
history=model.fit(Xtrain,Ytrain,batch_size=64 ,epochs=100)
```

6. Plotting of training accuracy and cost curve

```
##training curves
plt.plot(history.history['acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')

plt.show()
```

7. Evaluation of model on testing data

```
Ypred=model.predict(Xtest)
Ypred=(Ypred>0.5).astype(np.uint8)

CF=confusion_matrix(Ytest,Ypred)
print("CONFUSION MATRIX:\n",CF)
CLFR= classification_report(Ytest,Ypred)

print ("CLASSIFICATION REPORT:\n",CLFR)
```

OUTPUT:

1. **Training curves:**

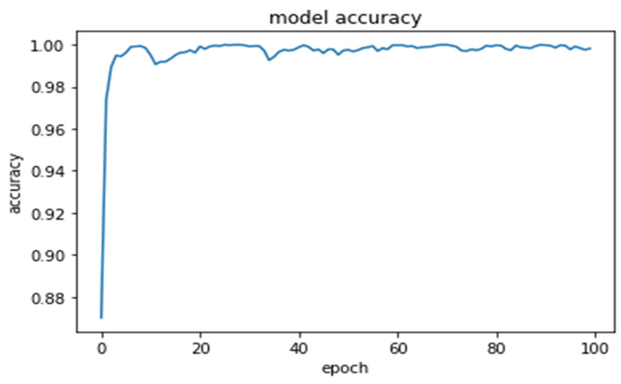


Fig: training accuracy curve

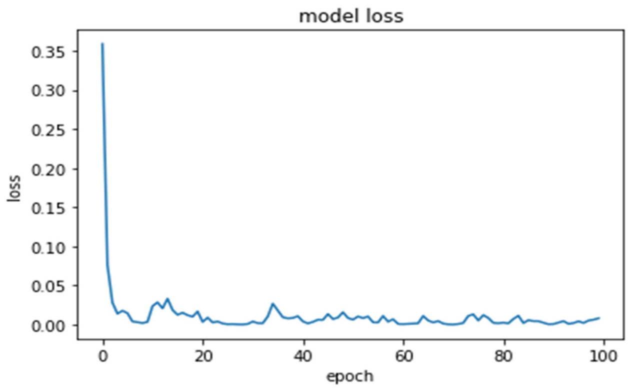
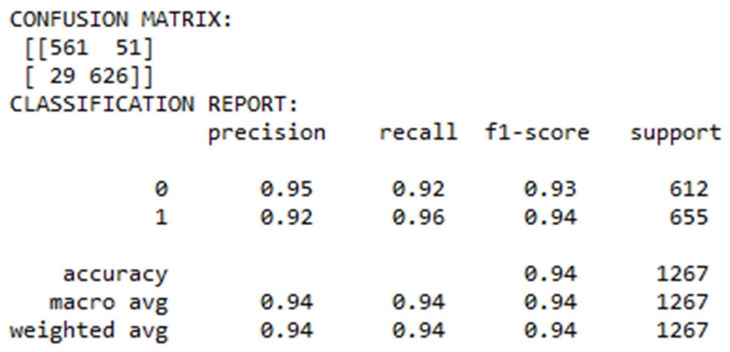


Fig: training loss curve

2. Model evaluation:



```
CONFUSION MATRIX:
 [[561  51]
 [ 29 626]]
CLASSIFICATION REPORT:
               precision    recall  f1-score   support

           0       0.95      0.92      0.93       612
           1       0.92      0.96      0.94       655

    accuracy                           0.94      1267
   macro avg       0.94      0.94      0.94      1267
weighted avg       0.94      0.94      0.94      1267
```

Fig: confusion matrix and classification report