**SAMPLE CODE**

**Main.py**

**from tkinter import messagebox**

**from tkinter import \***

**from tkinter import simpledialog**

**import tkinter**

**import matplotlib.pyplot as plt**

**import numpy as np**

**from tkinter import ttk**

**from tkinter import filedialog**

**import pandas as pd**

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

**from string import punctuation**

**from nltk.corpus import stopwords**

**import nltk**

**from nltk.stem import WordNetLemmatizer**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**from sklearn.preprocessing import LabelEncoder**

**from keras.models import Sequential**

**from keras.layers.core import Dense,Activation,Dropout**

**from sklearn.preprocessing import OneHotEncoder**

**import keras.layers**

**from keras.models import model\_from\_json**

**import pickle**

**import os**

**from sklearn.preprocessing import normalize**

**from keras.models import Sequential**

**from keras.layers import Dense, Dropout, Flatten, LSTM**

**main = Tk()**

**main.title("DETECTION OF FAKE NEWS THROUGH IMPLEMENTATION OF DATA SCIENCE APPLICATION")**

**main.geometry("1300x1200")**

**global filename**

**global X, Y**

**global tfidf\_X\_train, tfidf\_X\_test, tfidf\_y\_train, tfidf\_y\_test**

**global tfidf\_vectorizer**

**global accuracy,error**

**stop\_words = set(stopwords.words('english'))**

**lemmatizer = WordNetLemmatizer()**

**textdata = []**

**labels = []**

**global classifier**

**def cleanPost(doc):**

**tokens = doc.split()**

**table = str.maketrans('', '', punctuation)**

**tokens = [w.translate(table) for w in tokens]**

**tokens = [word for word in tokens if word.isalpha()]**

**tokens = [w for w in tokens if not w in stop\_words]**

**tokens = [word for word in tokens if len(word) > 1]**

**tokens = [lemmatizer.lemmatize(token) for token in tokens]**

**tokens = ' '.join(tokens)**

**return tokens**

**def uploadDataset():**

**global filename**

**text.delete('1.0', END)**

**filename = filedialog.askopenfilename(initialdir="TwitterNewsData")**

**textdata.clear()**

**labels.clear()**

**dataset = pd.read\_csv(filename)**

**dataset = dataset.fillna(' ')**

**for i in range(len(dataset)):**

**msg = dataset.get\_value(i, 'text')**

**label = dataset.get\_value(i, 'target')**

**msg = str(msg)**

**msg = msg.strip().lower()**

**labels.append(int(label))**

**clean = cleanPost(msg)**

**textdata.append(clean)**

**text.insert(END,clean+" ==== "+str(label)+"\n")**

**def preprocess():**

**text.delete('1.0', END)**

**global X, Y**

**global tfidf\_vectorizer**

**global tfidf\_X\_train, tfidf\_X\_test, tfidf\_y\_train, tfidf\_y\_test**

**stopwords=stopwords = nltk.corpus.stopwords.words("english")**

**tfidf\_vectorizer = TfidfVectorizer(stop\_words=stopwords, use\_idf=True, ngram\_range=(1,2),smooth\_idf=False, norm=None, decode\_error='replace', max\_features=200)**

**tfidf = tfidf\_vectorizer.fit\_transform(textdata).toarray()**

**df = pd.DataFrame(tfidf, columns=tfidf\_vectorizer.get\_feature\_names())**

**text.insert(END,str(df))**

**print(df.shape)**

**df = df.values**

**X = df[:, 0:df.shape[1]]**

**X = normalize(X)**

**Y = np.asarray(labels)**

**le = LabelEncoder()**

**Y = le.fit\_transform(Y)**

**indices = np.arange(X.shape[0])**

**np.random.shuffle(indices)**

**X = X[indices]**

**Y = Y[indices]**

**Y = Y.reshape(-1, 1)**

**print(X.shape)**

**encoder = OneHotEncoder(sparse=False)**

**#Y = encoder.fit\_transform(Y)**

**X = X.reshape((X.shape[0], X.shape[1], 1))**

**print(Y)**

**print(Y.shape)**

**print(X.shape)**

**tfidf\_X\_train, tfidf\_X\_test, tfidf\_y\_train, tfidf\_y\_test = train\_test\_split(X, Y, test\_size=0.2)**

**text.insert(END,"\n\nTotal News found in dataset : "+str(len(X))+"\n")**

**text.insert(END,"Total records used to train machine learning algorithms : "+str(len(tfidf\_X\_train))+"\n")**

**text.insert(END,"Total records used to test machine learning algorithms : "+str(len(tfidf\_X\_test))+"\n")**

**def runLSTM():**

**text.delete('1.0', END)**

**global classifier**

**if os.path.exists('model/model.json'):**

**with open('model/model.json', "r") as json\_file:**

**loaded\_model\_json = json\_file.read()**

**classifier = model\_from\_json(loaded\_model\_json)**

**classifier.load\_weights("model/model\_weights.h5")**

**classifier.\_make\_predict\_function()**

**print(classifier.summary())**

**f = open('model/history.pckl', 'rb')**

**data = pickle.load(f)**

**f.close()**

**acc = data['accuracy']**

**acc = acc[9] \* 100**

**text.insert(END,"LSTM Fake News Detection Accuracy : "+str(acc)+"\n\n")**

**text.insert(END,'LSTM Model Summary can be seen in black console for layer details\n')**

**with open('model/model.txt', 'rb') as file:**

**classifier = pickle.load(file)**

**file.close()**

**else:**

**lstm\_model = Sequential()**

**lstm\_model.add(LSTM(128, input\_shape=(X.shape[1:]), activation='relu', return\_sequences=True))**

**lstm\_model.add(Dropout(0.2))**

**lstm\_model.add(LSTM(128, activation='relu'))**

**lstm\_model.add(Dropout(0.2))**

**lstm\_model.add(Dense(32, activation='relu'))**

**lstm\_model.add(Dropout(0.2))**

**lstm\_model.add(Dense(2, activation='softmax'))**

**lstm\_model.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**hist = lstm\_model.fit(X, Y, epochs=10, validation\_data=(tfidf\_X\_test, tfidf\_y\_test))**

**classifier = lstm\_model**

**classifier.save\_weights('model/model\_weights.h5')**

**model\_json = classifier.to\_json()**

**with open("model/model.json", "w") as json\_file:**

**json\_file.write(model\_json)**

**accuracy = hist.history**

**f = open('model/history.pckl', 'wb')**

**pickle.dump(accuracy, f)**

**f.close()**

**acc = accuracy['accuracy']**

**acc = acc[9] \* 100**

**text.insert(END,"LSTM Accuracy : "+str(acc)+"\n\n")**

**text.insert(END,'LSTM Model Summary can be seen in black console for layer details\n')**

**print(lstm\_model.summary())**

**def graph():**

**f = open('model/history.pckl', 'rb')**

**data = pickle.load(f)**

**f.close()**

**acc = data['accuracy']**

**loss = data['loss']**

**plt.figure(figsize=(10,6))**

**plt.grid(True)**

**plt.xlabel('Epcchs')**

**plt.ylabel('Accuracy/Loss')**

**plt.plot(acc, 'ro-', color = 'green')**

**plt.plot(loss, 'ro-', color = 'blue')**

**plt.legend(['Accuracy','Loss'], loc='upper left')**

**#plt.xticks(wordloss.index)**

**plt.title('LSTM Model Accuracy & Loss Graph')**

**plt.show()**

**def predict():**

**testfile = filedialog.askopenfilename(initialdir="TwitterNewsData")**

**testData = pd.read\_csv(testfile)**

**text.delete('1.0', END)**

**testData = testData.values**

**testData = testData[:,0]**

**print(testData)**

**for i in range(len(testData)):**

**msg = testData[i]**

**msg1 = testData[i]**

**print(msg)**

**review = msg.lower()**

**review = review.strip().lower()**

**review = cleanPost(review)**

**testReview = tfidf\_vectorizer.transform([review]).toarray()**

**predict = classifier.predict(testReview)**

**print(predict)**

**if predict == 0:**

**text.insert(END,msg1+" === Given news predicted as GENUINE\n\n")**

**else:**

**text.insert(END,msg1+" == Given news predicted as FAKE\n\n")**

**font = ('times', 15, 'bold')**

**title = Label(main, text='DETECTION OF FAKE NEWS THROUGH IMPLEMENTATION OF DATA SCIENCE APPLICATION')**

**title.config(bg='gold2', fg='thistle1')**

**title.config(font=font)**

**title.config(height=3, width=120)**

**title.place(x=0,y=5)**

**font1 = ('times', 13, 'bold')**

**ff = ('times', 12, 'bold')**

**uploadButton = Button(main, text="Upload Fake News Dataset", command=uploadDataset)**

**uploadButton.place(x=20,y=100)**

**uploadButton.config(font=ff)**

**processButton = Button(main, text="Preprocess Dataset", command=preprocess)**

**processButton.place(x=20,y=150)**

**processButton.config(font=ff)**

**dtButton = Button(main, text="Run LSTM Algorithm", command=runLSTM)**

**dtButton.place(x=20,y=200)**

**dtButton.config(font=ff)**

**graphButton = Button(main, text="Accuracy & Loss Graph", command=graph)**

**graphButton.place(x=20,y=250)**

**graphButton.config(font=ff)**

**predictButton = Button(main, text="Test News Detection", command=predict)**

**predictButton.place(x=20,y=300)**

**predictButton.config(font=ff)**

**font1 = ('times', 12, 'bold')**

**text=Text(main,height=30,width=100)**

**scroll=Scrollbar(text)**

**text.configure(yscrollcommand=scroll.set)**

**text.place(x=330,y=100)**

**text.config(font=font1)**

**main.config(bg='DarkSlateGray1')**

**main.mainloop()**