```
from tkinter import *
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import numpy as np
import os
from tkinter import messagebox
import pandas as pd
from datetime import datetime
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt

sid = SentimentIntensityAnalyzer()

main = tkinter.Tk()
main.title("FRAUD DETECTION OF MOBILE APPS")
main.geometry("1200x1200")

global filename, dataset, total_apps, fraud_count



def uploadDataset():
    global filename
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="Dataset")
    text.insert(END,str(filename)+" Dataset Loaded\n\n")
    pathlabel.config(text=str(filename)+" Dataset Loaded\n\n")

def processDataset():
    global filename, dataset, total_apps
    text.delete('1.0', END)
```

```
    dataset = pd.read_csv(filename, usecols=["content","score","thumbsUpCount","at","appId"])
    text.insert(END,str(dataset.head()))
    unique, count = np.unique(dataset["appId"],return_counts=True)
    total_apps = len(unique)
    height = count
    bars = unique
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.title("Totals Apps with Reviews, Rating & Rank Count")
    plt.xticks(rotation=90)
    plt.show()

def getReviewSentiment(review): #calculate review sentiment
    sentiment_dict = sid.polarity_scores(review)
    compound = sentiment_dict['compound']
    result = ''
    if compound >= 0.05 :
        result = 'Positive'

    elif compound < 0.05 :
        result = 'Negative'
    return result

def getTimePeriod(start, end): #calculate time gap between start and next review
    fmt = '%Y-%m-%d %H:%M:%S'
    tstamp1 = datetime.strptime(start, fmt)
    tstamp2 = datetime.strptime(end, fmt)
    if tstamp1 > tstamp2:
        td = tstamp1 - tstamp2
    else:
```

```
        td = tstamp2 - tstamp1
    td_mins = int(round(td.total_seconds() / 60))
    return td_mins



def fraudDetection(): #function to detect fraud from dataset
    global dataset, fraud_count
    fraud_count = 0
    text.delete('1.0', END)
    app_id = np.unique(dataset["appId"]) #finding all uniques app from review history
    dataset = dataset.values
    for j in range(len(app_id)):
        old_session = None
        old_rating = 0
        old_rank = 0
        count = 0
        fraud_event = 0
        old_review = None
        text.insert(END,"Analysing App: "+app_id[j]+"\n")
        text.update_idletasks()
        for i in range(len(dataset)): #extracting review, rating, ranking from past history dataset
            review = dataset[i,0]
            ranking = dataset[i,1]
            ratings = dataset[i,2]
            session_time = dataset[i,3]
            app_name = dataset[i,4]
            #getting review sentiment whether user is positive or negative and if continuous positive
received from negative then fraud detected
            current_review = getReviewSentiment(review)
            if count == 0 and app_id[j] == app_name:
                old_session = session_time #getting first session time
```

```
            old_rating = ratings
            old_rank = ranking
            old_review = current_review
            count = 1
        elif app_id[j] == app_name and count > 0:
            lead_session = getTimePeriod(old_session, session_time) #calculating leading session time
            #if mobile app review has less than 3000 second time period gap between old new review
            #if continuous ranking is increasing from low rank
            #if continous rating increase from low rating and suddenly start receiving positive review then fraud detected
            if lead_session < 3000 and ratings > old_rating and ranking > old_rank and current_review == 'Positive':
                old_session = session_time
                old_rating = ratings
                old_rank = ranking
                old_review = current_review
                fraud_event = fraud_event + 1
    if fraud_event > 1:
        fraud_count = fraud_count + 1
        text.insert(END,app_id[j]+" Detected as Fraud\n\n")
        text.update_idletasks()
        print("detected fraud "+str(fraud_event)+" "+app_id[j])



def graph():
    height = [total_apps,fraud_count]
    bars = ["Totals Apps","Fraud App Detected"]
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.title("Fraud Apps Detection Graph")
    plt.xticks(rotation=90)
```

```
    plt.show()



def close():
    main.destroy()

font = ('times', 14, 'bold')
title = Label(main, text='FRAUD DETECTION OF MOBILE APPS')
title.config(bg='DarkGoldenrod1', fg='black')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=5,y=5)

font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload Mobile Reviews Dataset", command=uploadDataset)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=560,y=100)

matchButton = Button(main, text="Read & Process Dataset", command=processDataset)
matchButton.place(x=50,y=150)
matchButton.config(font=font1)

fraudButton = Button(main, text="Run Rating, Ranking & Review Based Fraud Detection",
command=fraudDetection)
fraudButton.place(x=50,y=200)
fraudButton.config(font=font1)
```

```
graphButton = Button(main, text="Detection Graph", command=graph)
graphButton.place(x=50,y=250)
graphButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)
exitButton.place(x=50,y=300)
exitButton.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=25,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=520,y=150)
text.config(font=font1)


main.config(bg='LightSteelBlue1')
main.mainloop()
```