App Insights

Main.py

import InsightsDesign as Over

import tkinter as tk

Over.startingScreen(tk.Tk())

Insights design.py

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

import DesignToInsertdataset1 as ds

#import Tkinter as tk

#import ttk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.cm as cm

def overviewClick(event):

print('')

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def sizeClick(event):

print("")

def machineClick(event):

global screen

import InsightsDesignForMachineLearningModels as mac

mac.startingScreen(screen)

def tableFormate(data,table\_frame):

global screen,df

global string\_table

x=60

y=30

string="App Name\t\tCategory\tRating \tReviews\t\tSize\tInstalls\t\tType\tPrice\tContent Rating\t Geners\t\tLast Update\tCurr ver Android Ver\n"

tk.Label(table\_frame,text=string,font=("Helvetica",11,'bold'),fg='black',bg='#d8dce3').place (x=x-3,y=y)

for index in range(len(data)):

if index%2==0:

color='#ffffff'

else:

color='#f5f5f5'

y+=30

string ="{}...\t\t{}\t {} \t{}\t\t{}\t{} \t{}\t{}\t{} \t{}\t{}\t{}\t {} \n".format(data['App'][index][0:13],str(data['Category'][index])[0:10],str(data['Rating'][index]) ,str(data['Reviews'][index]),str(data['Size'][index]),str(data['Installs'][index])[0:9],str(data['Type'][index]),str(data['Price'][index]),str(data['Content Rating'][index]),str(data['Genres'][index])[0:12],str(data['Last Updated'][index])[0:10],str(data['Current Ver'][index])[0:6],str(data['Android Ver'][index])[0:9])

tk.Label(table\_frame,text=string,font=("Helvetica",11,'bold'),fg='black',bg=color, borderwidth=2, relief="groove").place(x=x,y=y)

summary = "Summary\t\t Rows : %d Columns : %d "%(df.shape[0],df.shape[1])

tk.Label(table\_frame,text=summary,font=("Helvetica",11,'bold'),fg='black',bg="white" ).place(x=x,y=y+45)

#insert\_button = tk.Button(table\_frame,fg="white",font=('Helvetica',10,'bold'),width=13,text="+ADD",bg="#7aaffa",command=mouseClick).place(x=1315,y=y+42)

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(ws,hs))

window.configure(background='white')

def startingScreen(root):# main window nothing to be changed over here

global screen,df

df=pd.read\_csv('DATA SET-2.csv')

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="OVERVIEW",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_overview = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

lbl\_overview.bind("<Button-1>",overviewClick)

lbl\_overview.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame = tk.Frame(screen,bg='white',width='1520',height='730',bd=4,relief=RIDGE)

big\_frame.place(x=3,y=100)

"""this frame is to display the data table"""

table\_frame = tk.Frame(big\_frame,bg='white',width='1505',height='410',bd=4,relief=RIDGE)

table\_frame.place(x=10,y=310)

tk.Label(table\_frame,text="App's Data Table",width=15,height='1',font=("Calibri",11,'bold'),fg='black',bg='white').place(x=0,y=0)

tableFormate(df[0:10],table\_frame)

insert\_button = tk.Button(table\_frame,bd=2,fg="white",font=('Helvetica',9,'bold'),width=16,text="+ADD",bg="#3869d1",command=lambda :ds.startingScreen(screen)).place(x=1300,y=373)

canvas\_frame1 = tk.Frame(big\_frame,bg='gray',width=512,height=308,bd=4,relief=RIDGE)

canvas\_frame1.place(x=10,y=0)

#canvas1 = tk.Canvas(canvas\_frame1,bg="pink",height=307,width=498)

#canvas1.place(x=11,y=106)

df=df.replace(np.NaN,-999)

#print(df.isnull().sum())

#print(df.head(5))

dict\_cat\_count = {}

for index in range(len(df)):

if df['Category'][index]==-999:

continue

if df['Category'][index] in dict\_cat\_count:

dict\_cat\_count[df['Category'][index]]+=1

else:

dict\_cat\_count[df['Category'][index]]=1

#print(len(dict\_cat\_count))

y\_count=[]

x\_label=[]

for i in dict\_cat\_count:

x\_label.append(i)

y\_count.append(dict\_cat\_count[i])

for i in range(len(y\_count)-1):

for j in range(len(y\_count)-1):

if y\_count[j]<y\_count[j+1]:

y\_count[j],y\_count[j+1] = y\_count[j+1],y\_count[j]

x\_label[j],x\_label[j+1] = x\_label[j+1],x\_label[j]

#colors

#colors = ['#ff9999','#66b3ff','#99ff99']#,'#ffcc99']

figure1 = plt.Figure(figsize=(5,3), dpi=100)

color = cm.rainbow(np.linspace(0, 1, 10))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(y\_count[:10], labels=x\_label[:10],colors = color, autopct='%1.1f%%', startangle=90)

ax3.set\_title("Pie chart on Category (Top 10)")

pie\_plot = FigureCanvasTkAgg(figure1, canvas\_frame1)

pie\_plot.get\_tk\_widget().place(x=0,y=0)

#ax3.legend()

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

dict\_Install\_count = {}

for index in range(len(df)):

if df['Installs'][index]==-999:

continue

if df['Installs'][index] in dict\_Install\_count:

dict\_Install\_count[df['Installs'][index]]+=1

else:

dict\_Install\_count[df['Installs'][index]]=1

#print(len(dict\_Install\_count))

y\_count=[]

x\_label=[]

for i in dict\_Install\_count:

x\_label.append(i)

y\_count.append(dict\_Install\_count[i])

for i in range(len(y\_count)-1):

for j in range(len(y\_count)-1):

if y\_count[j]<y\_count[j+1]:

y\_count[j],y\_count[j+1] = y\_count[j+1],y\_count[j]

x\_label[j],x\_label[j+1] = x\_label[j+1],x\_label[j]

canvas\_frame2 = tk.Frame(big\_frame,bg='gray',width=710,height=308,bd=4,relief=RIDGE)

canvas\_frame2.place(x=520,y=0)

#canvas = tk.Canvas(canvas\_frame2,bg="black",height="300",width="308")

#canvas.place(x=1213,y=105)

color = cm.rainbow(np.linspace(0, 1, 10))

#print(y\_count[:10])

#print(x\_label[:10])

for i in range(len(x\_label)):

n = x\_label[i].count("0")

if n==3:

x\_label[i]=x\_label[i][:1]+"k"

if n==4:

x\_label[i]=x\_label[i][:2]+"k"

if n==5:

x\_label[i]=x\_label[i][:3]+"k"

if n==6:

x\_label[i]=x\_label[i][:1]+"M"

if n==7:

x\_label[i]=x\_label[i][:2]+"M"

figure2 = plt.Figure(figsize=(7,3), dpi=100)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

chart.bar(x\_label[:10],y\_count[:10],color=color)

chart.set\_ylabel("Frequency")

chart.set\_xlabel("Installs")

figure2.suptitle("Bar chart on Installs (Top 10)")

chart.legend()

canvas = FigureCanvasTkAgg(figure2, master=canvas\_frame2)

canvas.get\_tk\_widget().place(x=3,y=0)

canvas\_frame3 = tk.Frame(big\_frame,bg='gray',width=400,height=308,bd=4,relief=RIDGE)

canvas\_frame3.place(x=1200,y=0)

df = pd.read\_csv("DATA SET-2.csv")

#print(df.head(5))

#print(df['Type'].head(5))

df=df.replace(np.NaN,-999)

df['Type'] = df['Type'].map(lambda x : 1 if x=="Free" else 0)

count\_free=0

count\_paid=0

for i in df['Type']:

if i == 1:

count\_free+=1

else:

count\_paid+=1

figure3 = plt.Figure(figsize=(4,4), dpi=80)

chart = figure3.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

chart.bar(['FREE','PAID'],[count\_free,count\_paid],color=['red','blue'])

chart.set\_ylabel("Frequency")

chart.set\_xlabel("Type")

figure3.suptitle("Count plot for Free Vs Paid")

chart.legend()

canvas = FigureCanvasTkAgg(figure3, master=canvas\_frame3)

canvas.get\_tk\_widget().place(x=3,y=0)

plt.show()

screen.mainloop()

#startingScreen(tk.Tk())

Insights design for category.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 2 21:46:07 2019

@author: Dharmik joshi

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

#import Tkinter as tk

#import ttk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.cm as cm

def function\_q5(event):

global screen

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='800',height='710',bd=4,relief=RIDGE)

big\_frame.place(x=25,y=60)

w=850

h=790

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("DATA SET-2.csv")

df=df.replace(np.NaN,-999)

dict\_cat\_count = {}

for index in range(len(df)):

if df['Category'][index]==-999:

continue

if df['Category'][index] in dict\_cat\_count:

dict\_cat\_count[df['Category'][index]]+=1

else:

dict\_cat\_count[df['Category'][index]]=1

y\_count=[]

x\_label=[]

for i in dict\_cat\_count:

x\_label.append(i)

y\_count.append(dict\_cat\_count[i])

for i in range(len(y\_count)-1):

for j in range(len(y\_count)-1):

if y\_count[j]<y\_count[j+1]:

y\_count[j],y\_count[j+1] = y\_count[j+1],y\_count[j]

x\_label[j],x\_label[j+1] = x\_label[j+1],x\_label[j]

#colors

#colors = ['#ff9999','#66b3ff','#99ff99']#,'#ffcc99']

figure1 = plt.Figure(figsize=(7,6), dpi=100)

color = cm.rainbow(np.linspace(0, 1, 10))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.barh(x\_label, y\_count,color = color)

ax3.set\_title("CHART ON TREND OF DOWNLOAD OVER THE CATEGORY")

bar\_plot = FigureCanvasTkAgg(figure1, big\_frame)

bar\_plot.get\_tk\_widget().place(x=0,y=0)

ax3.set\_xlabel("Downloads")

#ax3.legend()

root.mainloop()

""" Ques 5 END=============================================================================================="""

""" Ques 3 START=============================================================================================="""

def qn3(category):

global sample

Installs = []

sample.drop(index=10472,axis=0,inplace=True)

sample=sample.replace(np.NaN,0)

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

ans=[]

count = []

for i in category:

total=0

c=0

for j in range(len(sample['Category'])):

if j!=10472:

if sample['Category'][j]==i:

total=total+Installs[j]

c+=1

ans.append(total)

count.append(c)

cat,avg = [],[]

for index in range(len(ans)):

cat.append(category[index])

avg.append(round(ans[index]/count[index]))

return cat,avg

def function\_q3(event):

global screen,sample

sample = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='800',height='710',bd=4,relief=RIDGE)

big\_frame.place(x=25,y=60)

w=850

h=790

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

category=sample['Category'].unique()

cat,avg = qn3(category)

#print(len(cat))

#print(len(avg))

#print(cat[avg.index(max(avg))])

#print(max(avg))

#print(cat[avg.index(min(avg))])

#print(min(avg))

lowest = []

for index in range(len(avg)):

if avg[index]<250000:

lowest.append(category[index])

label = category

val = avg

color = cm.rainbow(np.linspace(0, 1, len(label)))

figure2 = plt.Figure(figsize=(7,6), dpi=100)

chart = figure2.add\_subplot(111)

bar\_plot = chart.barh(label,val,color=color)

chart.set\_ylabel("Category")

chart.set\_xlabel("Average Installs")

figure2.suptitle("Category with Their Average Download")

chart.legend()

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=10,y=0)

String="""

There are 33 Unique Categories are available in the Data set where {} Category has

managed to get the Maximum Downloads across all the years and {} Category has got

the Lowest Downloads Accept {} and {} Category All the categories have managed to get

the average of 2,50,000 Downloads

""".format(cat[avg.index(max(avg))],cat[avg.index(min(avg))],lowest[0],lowest[1])

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=565)

root.mainloop()

""" Ques 3 END================================================================================================="""

"""Feature cat\_download size START============================================================================="""

def feature\_cat\_size(event):

global screen,df

df = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='750',height='690',bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=800

h=790

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("DATA SET-2.csv")

#print(df.head(5))

# Data cleaning for "Size" column

#print(df['Size'].head(5))

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

#print(df['Size'].head(5))

df=df.replace(np.NaN,-999)

#print(df.isnull().sum())

#print(df['Category'].unique())

total = 0

count=0

dict\_cat\_size = {}

for index in range(len(df)):

if df['Category'][index]==-999 or df['Size'][index]==-999 :

continue

if df['Category'][index] in dict\_cat\_size:

dict\_cat\_size[df['Category'][index]][0]+=float(df['Size'][index])

dict\_cat\_size[df['Category'][index]][1]+=1

else:

dict\_cat\_size[df['Category'][index]]=[float(df['Size'][index]),1]

#print(dict\_cat\_size)

for category in dict\_cat\_size:

try:

size = dict\_cat\_size[category][0]

count = dict\_cat\_size[category][1]

dict\_cat\_size[category]=float(size/count)

except:

print('')

#print(dict\_cat\_size)

#print(len(dict\_cat\_size))

x\_label,y\_count = [],[]

for cat in dict\_cat\_size:

x\_label.append(cat)

y\_count.append(dict\_cat\_size[cat])

colors = cm.rainbow(np.linspace(0, 1, len(x\_label)))

figure2 = plt.Figure(figsize=(9,7), dpi=80)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

bar\_plot = chart.barh(x\_label,y\_count,color=colors)

chart.set\_ylabel("CATEGORY")

chart.set\_xlabel("Download size")

figure2.suptitle('Download size per Category')

chart.legend()

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=3,y=0)

#'TRAVEL\_AND\_LOCAL,Family,game,sports

String="""

The Google Play Store shows apps’ actual download sizes.

Game and Family app categories have the highest average download size,

while TOOLS category is the smallest on average.

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=50,y=520)

root.mainloop()

"""Feature cat\_download size END============================================================================="""

"""Feature cat\_rev START============================================================================="""

def feature\_cat\_review(event):

global screen,df

df = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='750',height='690',bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=800

h=790

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("DATA SET-2.csv")

df=df.replace(np.NaN,-999)

dict\_cat\_review = {}

for index in range(len(df)):

if df['Category'][index]==-999:

continue

if df['Category'][index] in dict\_cat\_review:

dict\_cat\_review[df['Category'][index]]+=df['Reviews'][index]

else:

dict\_cat\_review[df['Category'][index]]=df['Reviews'][index]

y\_count=[]

x\_label=[]

for i in dict\_cat\_review:

x\_label.append(i)

y\_count.append(dict\_cat\_review[i])

#print(y\_count)

colors = cm.rainbow(np.linspace(0, 1, len(x\_label)))

figure2 = plt.Figure(figsize=(9,7), dpi=80)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

bar\_plot = chart.barh(x\_label,y\_count,color=colors)

chart.set\_ylabel("CATEGORY")

chart.set\_xlabel("COUNT OF TOTAL REVIEWS")

figure2.suptitle('COUNT OF TOTAL REVIEW UNDER DIFFERENT CATEGORY')

chart.legend()

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=3,y=0)

String="""

It looks like only a small number of users take the time to write

an app review.Our analysis has shown that users tend to leave more

reviews for apps in Games,Social,Communication and Family categories

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=50,y=520)

root.mainloop()

"""Feature cat\_rev END============================================================================="""

""" Ques 4 START================================================================================================================================="""

def function\_q4(event):

global screen,df

df = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='580',height='530',bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=600

h=600

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("DATA SET-2.csv")

df=df.replace(np.NaN,-999)

#print(df.isnull().sum())

#print(df.head(5))

total\_rating = 0

dict\_cat\_review = {}

for index in range(len(df)):

if df['Category'][index]==-999 or df['Rating'][index]==-999:

continue

if df['Category'][index] in dict\_cat\_review:

dict\_cat\_review[df['Category'][index]][0]+=df['Rating'][index]

dict\_cat\_review[df['Category'][index]][1]+=1

total\_rating+=df['Rating'][index]

else:

dict\_cat\_review[df['Category'][index]]=[df['Rating'][index],1]

total\_rating+=df['Rating'][index]

total=0

count=0

for i in df['Rating']:

if i!=-999:

total+=i

count+=1

avg\_rating= total/count

y\_count=[]

x\_label=[]

for i in dict\_cat\_review:

if dict\_cat\_review[i][0]/dict\_cat\_review[i][1]>=avg\_rating:

avg = (dict\_cat\_review[i][0]/dict\_cat\_review[i][1])

x\_label.append(i)

y\_count.append(float(avg))

figure3 = plt.Figure(figsize=(7,4), dpi=80)

ax3 = figure3.add\_subplot(111)

ax3.scatter(y\_count,x\_label,color='r')

scatter3 = FigureCanvasTkAgg(figure3, big\_frame)

scatter3.get\_tk\_widget().place(x=10,y=0)

ax3.legend()

ax3.set\_xlabel("RATING")

ax3.set\_ylabel("CATEGORY")

ax3.set\_title('CATEGORIES WITH HIGHEST MAXIMUM AVERAGE RATING')

String = """

There are Total 33 Unique categories are present in the data set

but out of {} Categories has managed to get the Highest Maximum

Average Rating of From the User

Average Rating from given data set is {:.1f}

""".format(len(x\_label),avg\_rating)

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=300)

root.mainloop()

""" Ques 4 END================================================================================================================================="""

def data():

global frame

for i in range(50):

tk.Label(frame,text=i).grid(row=i,column=0)

tk.Label(frame,text="my text"+str(i)).grid(row=i,column=1)

tk.Label(frame,text="..........").grid(row=i,column=2)

def myfunction(event):

global canvas

canvas.configure(scrollregion=canvas.bbox("all"),width=200,height=200)

def overviewClick(event):

global screen

import InsightsDesign as over

over.startingScreen(screen)

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def machineClick(event):

global screen

import InsightsDesignForMachineLearningModels as mac

mac.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def sizeClick(event):

print("")

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(ws,hs))

window.configure(background='white')

def startingScreen(root):

global screen,df,canvas\_big\_frame,frame

df=pd.read\_csv('DATA SET-2.csv')

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="CATEGORY",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_over = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_over.bind("<Button-1>",overviewClick)

lbl\_over.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame=tk.Frame(screen,bg='#F8E0E0',width='1520',height='730',bd=4,relief=RIDGE)# to change background in category page

big\_frame.place(x=3,y=100)

"""

4) Which category of apps have managed to get the highest maximum average ratings from the users.

3) Which category of apps have managed to get the most,least and an average of 2,50,000 downloads atleast.

"""

q4 = tk.Label(big\_frame,text = "The apps that managed to get the highest maximum rating from the user.",width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q4.bind("<Button-1>", function\_q4)

q4.place(x=50,y=10)

feat\_cat\_rev = tk.Label(big\_frame,text = "Category V/S Review",width=52,height=6,font=("Calibri",17,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

feat\_cat\_rev.bind("<Button-1>", feature\_cat\_review)

feat\_cat\_rev.place(x=750,y=10)

#function\_q11(frame1)

feat\_cat\_size = tk.Label(big\_frame,text = "Category V/S Size",width=52,height=6,font=("Calibri",17,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

feat\_cat\_size.bind("<Button-1>", feature\_cat\_size)

feat\_cat\_size.place(x=50,y=250)

q3 = tk.Label(big\_frame,text = "Category of apps that managed to get most, least an average of 2,50,000 downloads.",width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q3.bind("<Button-1>", function\_q3)

q3.place(x=750,y=250)

q5 = tk.Label(big\_frame,text = "Category wise download trend for the data is available over the period.",width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q5.bind("<Button-1>", function\_q5)

q5.place(x=50,y=490)

screen.mainloop()

#startingScreen(tk.Tk())

Insights design for installs.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 2 21:46:07 2019

@author: Dharmik joshi

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.cm as cm

#import Tkinter as tk

#import ttk

"""Ques 1 START===================================================================================================================="""

def function\_q1(event):

global screen

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='1010',height=750,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=1025

h=800

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Last Update",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

df = pd.read\_csv("DATA SET-2.csv")

df=df.replace(np.NaN,-999)

dict\_cat\_count = {}

for index in range(len(df)):

if df['Category'][index]==-999:

continue

if df['Category'][index] in dict\_cat\_count:

dict\_cat\_count[df['Category'][index]]+=1

else:

dict\_cat\_count[df['Category'][index]]=1

y\_count=[]

x\_label=[]

for i in dict\_cat\_count:

x\_label.append(i)

y\_count.append(dict\_cat\_count[i])

figure1 = plt.Figure(figsize=(10,7), dpi=100)

color = cm.rainbow(np.linspace(0, 1, len(x\_label)))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(y\_count, labels=x\_label,colors = color, autopct='%1.1f%%', startangle=200)

ax3.set\_title("Pie chart on Category")

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, big\_frame)

pie\_plot.get\_tk\_widget().place(x=0,y=0)

#ax3.legend(loc=0)

string = """

FAMILY and GAME were the most common categories, accounting for about 18.2% and 10.6% respectively, of the total number of

apps in our dataset. Parenting,Wheather and Comics were the least prevalent category with less than 1% of the total number

of apps in our dataset.

"""

tk.Label(big\_frame,text=string,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=590)

root.mainloop()

"""Ques 1 END===================================================================================================================="""

"""Ques 10 START================================================================================================================"""

def install():

Installs=[]

global sample

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

return Installs

def qn10part2():

global sample

content=sample['Content Rating']

counter=0

installs=install()

teen=0

mature=0

x=0

for i in content:

if i.strip()=='Teen':

teen=teen+installs[counter]

counter=counter+1

elif i.strip()=='Mature 17+':

mature=mature+installs[counter]

counter=counter+1

return teen,mature

def function\_q10(event):

global sample

sample = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='450',height='500',bd=4,relief=RIDGE)

big\_frame.place(x=30,y=60)

w=500

h=570

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

teen,mature = qn10part2()

#print(teen,mature)

#print(teen/mature)

#plt.bar(['teen','mature'],[teen,mature],color=['pink','blue'])

label = ['teen','mature']

val= [teen,mature]

color = color=['green','yellow']

figure2 = plt.Figure(figsize=(4,4), dpi=100)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

bar\_plot = chart.bar(label,val,color=color)

chart.set\_ylabel("Frequency")

chart.set\_xlabel("Installs")

figure2.suptitle("Installs for Teen")

chart.legend()

"""

for idx,rect in enumerate(bar\_plot):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,val[idx],ha='center', va='bottom', rotation=0)

"""

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=0,y=10)

String ="""The ratio of downloads for the app that

qualifies as teen versus mature17+ is {:.1f}""".format(teen/float(mature))

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=410)

root.mainloop()

"""Ques 10 END ================================================================================================================="""

"""Ques 2 START===================================================================================================================="""

def install():

global sample

Installs=[]

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

return Installs

def compare():

global solution,Installs,sample

r1=0 #number of apps under 10000 to 50000

r2=0 #number of apps under 50000 to 150000

r3=0 #number of apps under 150000 to 500000

r4=0 #number of apps under 500000 to 5000000

r5=0 #number of apps over 5000000

temp1,temp2,temp3,temp4,temp5=[],[],[],[],[]

for i in range(len(Installs)):

if i!=10472:

if Installs[i]>=10000 and Installs[i]<50000:

r1=r1+1

temp1.append(sample['App'][i])

if Installs[i]>=50000 and Installs[i]<150000:

r2=r2+1

temp2.append(sample['App'][i])

if Installs[i]>=150000 and Installs[i]<500000:

r3=r3+1

temp3.append(sample['App'][i])

if Installs[i]>=500000 and Installs[i]<5000000:

r4=r4+1

temp4.append(sample['App'][i])

if Installs[i]>=5000000:

r5=r5+1

temp5.append(sample['App'][i])

solution=[temp1,temp2,temp3,temp4,temp5]

return [r1,r2,r3,r4,r5]

def function\_q2(event):

global screen,Installs,solution,sample

# source for question 2

Installs=[] # list for storing integer based installation for easy comparision

solution=[1,1,1,1,1] #list storing names of apps which falls under particular category of sorting

sample=pd.read\_csv('DATA SET-2.csv')#reading data for the data set

sample=sample.replace(np.NaN,0)

sample.drop(index=[10472],inplace=True)

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

global screen,df,dict\_app\_relation

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='650',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=30,y=60)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

label = ["10k-50k","50k-150k","150k-500k","500k-5M","More than 5M"]

val = compare()

color = cm.rainbow(np.linspace(0, 1, 10))

figure2 = plt.Figure(figsize=(6,4), dpi=100)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

bar\_plot = chart.bar(label,val,color=color)

chart.set\_ylabel("Frequency")

chart.set\_xlabel("Installs")

figure2.suptitle("Month Indicating best downloads")

chart.legend()

for idx,rect in enumerate(bar\_plot):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,val[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=0,y=10)

String="""

The Google Play Store offers a wide range of applications,

Many Applications got 500k-5M and more than 5M Installs

Not a single Application has the Install in range between 150K-500K,

45% of application have got more than 500K Installs

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=40,y=400)

root.mainloop()

"""Ques 2 END===================================================================================================================="""

"""Ques 7 START===================================================================================================================="""

def function\_q7(event):

global screen,df,dict\_app\_relation

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='700',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=30,y=60)

w=750

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("DATA SET-2.csv")

#print(df.head(5))

dict\_varies=[]

dict\_andro=[]

"""These Dictionary will store the key as Category of App and items as name of the app"""

dict\_cat\_app\_var={}

dict\_cat\_app\_andro={}

df.drop(df.index[9148],axis=0, inplace=True)

df.drop(df.index[10472],axis=0,inplace=True)

# Data cleaning for "Installs" column

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

for index in range(len(df['App'])):

try:

if df['Android Ver'][index] == "Varies with device":

dict\_varies.append(df['Installs'][index])

if df['Category'][index] in dict\_cat\_app\_var:

dict\_cat\_app\_var[df['Category'][index]]+=df['Installs'][index]

else:

dict\_cat\_app\_var[df['Category'][index]] = df['Installs'][index]

#dict\_varies[df['App'][index]] = df['Installs'][index]

else:

dict\_andro.append(df['Installs'][index])

#dict\_andro[df['App'][index]] = df['Installs'][index]

if df['Category'][index] in dict\_cat\_app\_andro:

dict\_cat\_app\_andro[df['Category'][index]]+=df['Installs'][index]

else:

dict\_cat\_app\_andro[df['Category'][index]]=df['Installs'][index]

except:

continue

sum\_varies = sum(dict\_varies)

sum\_andro = sum(dict\_andro)

size=[sum\_varies,sum\_andro]

Android\_Ver = ['Varying', 'Not varying']

figure1 = plt.Figure(figsize=(5,4), dpi=80)

color = cm.rainbow(np.linspace(0, 1, 10))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(size, labels=Android\_Ver,colors = color, autopct='%1.1f%%', startangle=200)

ax3.set\_title("""% download of Varying Android \n

version vs Non-varying Android Ver in dataset""",fontsize=10)

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, big\_frame)

pie\_plot.get\_tk\_widget().place(x=0,y=0)

#ax3.legend(loc=0)

figure1 = plt.Figure(figsize=(5,3), dpi=80)

count = [len(dict\_varies),len(dict\_andro)]

color = cm.rainbow(np.linspace(0, 1, 2))

#fig1, ax1 = plt.subplots()

ax2 = figure1.add\_subplot(111)

ax2.pie(count, labels=Android\_Ver,colors = color, autopct='%1.1f%%', startangle=200)

ax2.set\_title("Frequency Varying Android version vs Non-varying Android Ver in dataset",fontsize=9)

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, big\_frame)

pie\_plot.get\_tk\_widget().place(x=100,y=300)

#ax3.legend(loc=0)

String ="""

Given DataSet consist wide range of applications,

With different Android Versions outof which only

12.5% Applications Have varying Android Version

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=310,y=60)

String ="""

Frequency of Varying Android Version is lesser in the data set but

their total download is higher than the Applications with other Android version

This shows that Application with Varying android version has chance to get higher downloads

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=510)

root.mainloop()

"""Ques 7 END===================================================================================================================="""

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def overviewClick(event):

global screen

import InsightsDesign as over

over.startingScreen(screen)

def machineClick(event):

global screen

import InsightsDesignForMachineLearningModels as mac

mac.startingScreen(screen)

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def sizeClick(event):

print('')

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(ws,hs))

window.configure(background='white')

def startingScreen(root):

global screen,df

df=pd.read\_csv("DATA SET-2.csv")

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="INSTALLS",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_over = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_over.bind("<Button-1>",overviewClick)

lbl\_over.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame = tk.Frame(screen,bg='#F8E0E0',width='1520',height='730',bd=4,relief=RIDGE)

big\_frame.place(x=3,y=100)

"""

1) What is the percentage download in each category on the playstore.

2) How many apps have managed to get the following number of downloads

a) Between 10,000 and 50,000

b) Between 50,000 and 150000

c) Between 150000 and 500000

d) Between 500000 and 5000000

e) More than 5000000

7)All those apps , whose android version is not an issue and can work with varying devices , what is the percentage increase or decrease in the downloads.

"""

q1 = tk.Label(big\_frame,text = "Percentage download for each category on the playstore.",width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q1.bind("<Button-1>", function\_q1)

q1.place(x=50,y=10)

q2str='''

Apps Installs

a) Between 10,000 and 50,000

b) Between 50,000 and 150000

c) Between 150000 and 500000

d) Between 500000 and 5000000

e) More than 5000000'''

q2 = tk.Label(big\_frame,text =q2str ,width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q2.bind("<Button-1>", function\_q2)

q2.place(x=750,y=10)

#function\_q11(frame1)

q7 = tk.Label(big\_frame,text = '''All those apps , whose android version is not an issue and can

work with varying devices ,what is the percentage increase or decrease

in the downloads.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q7.bind("<Button-1>", function\_q7)

q7.place(x=50,y=250)

q10 = tk.Label(big\_frame,text = "Question 10 Teen Vs Mature 17+",width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q10.bind("<Button-1>", function\_q10)

q10.place(x=750,y=250)

screen.mainloop()

#startingScreen(tk.Tk())

Installs design for last updated

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 2 21:46:07 2019

@author: Dharmik joshi

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

#import Tkinter as tk

#import ttk

import matplotlib.cm as cm

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import pandas as pd

from matplotlib import pyplot as plt

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

import statsmodels.api as sm

import sys

from collections import OrderedDict

import matplotlib.cm as cm

"""Ques 10 START==========================================================================================================="""

def month(x):

if x[0:3]=='Jan':

return 1

elif x[0:3]=='Feb':

return 2

elif x[0:3]=='Mar':

return 3

elif x[0:3]=='Apr':

return 4

elif x[0:3]=='Ma' or x[0:3]=='May':

return 5

elif x[0:3]=='Jun':

return 6

elif x[0:3]=='Jul':

return 7

elif x[0:3]=='Aug':

return 8

elif x[0:3]=='Sep':

return 9

elif x[0:3]=='Oct':

return 10

elif x[0:3]=='Nov':

return 11

elif x[0:3]=='Dec':

return 12

def install():

Installs=[]

global sample

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

return Installs

def dates\_str\_to\_int():

global sample

dates=sample['Last Updated']

year=[]

counter=0

for i in dates:

year.append([int(i[-8:-6]),month(i[:-9]),int(i[-4:])])

counter=counter+1

return year

def qn10():

year=dates\_str\_to\_int()

installs=install()

category=list(OrderedDict.fromkeys(sample['Category']))

temp=[]

counter=0

for i in category:

temp.append([0,0,0,0,0,0,0,0,0,0,0,0])

for i in sample['Category']:

jcounter=0

for j in category:

if i==j:

if year[counter][1]==1:

temp[jcounter][0]=temp[jcounter][0]+installs[counter]

elif year[counter][1]==2:

temp[jcounter][1]=temp[jcounter][1]+installs[counter]

elif year[counter][1]==3:

temp[jcounter][2]=temp[jcounter][2]+installs[counter]

elif year[counter][1]==4:

temp[jcounter][3]=temp[jcounter][3]+installs[counter]

elif year[counter][1]==5:

temp[jcounter][4]=temp[jcounter][4]+installs[counter]

elif year[counter][1]==6:

temp[jcounter][5]=temp[jcounter][5]+installs[counter]

elif year[counter][1]==7:

temp[jcounter][6]=temp[jcounter][6]+installs[counter]

elif year[counter][1]==8:

temp[jcounter][7]=temp[jcounter][7]+installs[counter]

elif year[counter][1]==9:

temp[jcounter][8]=temp[jcounter][8]+installs[counter]

elif year[counter][1]==10:

temp[jcounter][9]=temp[jcounter][9]+installs[counter]

elif year[counter][1]==11:

temp[jcounter][10]=temp[jcounter][10]+installs[counter]

elif year[counter][1]==12:

temp[jcounter][11]=temp[jcounter][11]+installs[counter]

jcounter=jcounter+1

counter=counter+1

return temp

def qn10try():

global sample

year=dates\_str\_to\_int()

installs=install()

category=list(OrderedDict.fromkeys(sample['Category']))

temp=[0,0,0,0,0,0,0,0,0,0,0,0]

counter=0

for i in year:

if i[1]==1:

temp[0]=temp[0]+installs[counter]

counter=counter+1

elif i[1]==2:

temp[1]=temp[1]+installs[counter]

counter=counter+1

elif i[1]==3:

temp[2]=temp[2]+installs[counter]

counter=counter+1

elif i[1]==4:

temp[3]=temp[3]+installs[counter]

counter=counter+1

elif i[1]==5:

temp[4]=temp[4]+installs[counter]

counter=counter+1

elif i[1]==6:

temp[5]=temp[5]+installs[counter]

counter=counter+1

elif i[1]==7:

temp[6]=temp[6]+installs[counter]

counter=counter+1

elif i[1]==8:

temp[7]=temp[7]+installs[counter]

counter=counter+1

elif i[1]==9:

temp[8]=temp[8]+installs[counter]

counter=counter+1

elif i[1]==10:

temp[9]=temp[9]+installs[counter]

counter=counter+1

elif i[1]==11:

temp[10]=temp[10]+installs[counter]

counter=counter+1

elif i[1]==12:

temp[11]=temp[11]+installs[counter]

counter=counter+1

return temp

def function\_q10(event):

global screen,sample

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='740',height=650,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=750

h=750

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Last Update",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

sample=pd.read\_csv('DATA SET-2.csv')#reading data for the data set

sample=sample.replace(np.NaN,0)

sample.drop(index=[10472],inplace=True)

cate\_month = qn10()

dict\_month = {1:'Jan',2:'Feb',3:'March',4:'April',5:'May',6:'June',7:'July',8:'Aug',9:'Sept',10:'Oct',11:'Nov',12:'Dec'}

categories = []#x-label

months = [] #text

maxinstalls = [] #y value

cat = sample['Category'].unique()

for index in range(len(cat)):

categories.append(cat[index])

maxinstalls.append(max(cate\_month[index]))

m = (cate\_month[index].index(max(cate\_month[index]))+1)

months.append(dict\_month[m])

#print(categories)

#print(qn10try())#for month list[month]=download index+1= month

colors = cm.rainbow(np.linspace(0, 1, len(categories)))

figure2 = plt.Figure(figsize=(9,7), dpi=80)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

bar\_plot = chart.barh(categories,maxinstalls,color=colors)

chart.set\_ylabel("CATEGORY")

chart.set\_xlabel("COUNT OF TOTAL REVIEWS")

figure2.suptitle('Maximum download month across all the year for each category')

chart.legend()

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=3,y=0)

#print(max(maxinstalls))

#print(categories[maxinstalls.index(max(maxinstalls))])

#print(months[maxinstalls.index(max(maxinstalls))])

install\_month =qn10try()

#print(max(install\_month))

#print(install\_month.index(max(install\_month)))

#print(dict\_month[install\_month.index(max(install\_month))+1])

String="""Out of All categories present in data set Game category has seen the maximum downloads

Maximum downloads for Game Category came in July month across all the years"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=530)

String="""Across all the years ,which {} month has seen the maximum downloads from each of the category.""".format(dict\_month[install\_month.index(max(install\_month))+1])

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=580)

root.mainloop()

"""Ques 10 END==========================================================================================================="""

"""Ques 6 START==========================================================================================================="""

def function\_q6(event):

global screen

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='black',width='700',height=550,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=720

h=650

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Last Update",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

df = pd.read\_csv("DATA SET-2.csv")

#print(df.head(5))

#df.drop(9148,axis=0, inplace=True)

#df.drop(10472,axis=0,inplace=True)

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

#print((df['year'][5]))

#6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the

dict\_2016 = {}

dict\_2017 = {}

dict\_2018 = {}

Category = []

for cat in df['Category'].unique():

Category.append(cat)

dict\_2016[cat]=0

dict\_2017[cat]=0

dict\_2018[cat]=0

#print(Category)

for index in range(len(df)):

if df['year'][index]==2016:

dict\_2016[df['Category'][index]] += df['Installs'][index]

if df['year'][index]==2017:

dict\_2017[df['Category'][index]] += df['Installs'][index]

if df['year'][index]==2018:

dict\_2018[df['Category'][index]] += df['Installs'][index]

#print(len(dict\_2016))

#print(len(dict\_2017))

#print(len(dict\_2018))

#print(dict\_2016)

#print(dict\_2017)

#print(dict\_2018)

max\_2016\_install = ["",0]

max\_2017\_install = ["",0]

max\_2018\_install = ["",0]

min\_2016\_install = ["",99999999999]

min\_2017\_install = ["",99999999999]

min\_2018\_install = ["",99999999999]

for cat in dict\_2016:

if max\_2016\_install[1] < dict\_2016[cat]:

max\_2016\_install[1] = dict\_2016[cat]

max\_2016\_install[0] = cat

if max\_2017\_install[1] < dict\_2017[cat]:

max\_2017\_install[1] = dict\_2017[cat]

max\_2017\_install[0] = cat

if max\_2018\_install[1] < dict\_2018[cat]:

max\_2018\_install[1] = dict\_2018[cat]

max\_2018\_install[0] = cat

if min\_2016\_install[1] > dict\_2016[cat]:

min\_2016\_install[1] = dict\_2016[cat]

min\_2016\_install[0] = cat

if min\_2017\_install[1] > dict\_2017[cat]:

min\_2017\_install[1] = dict\_2017[cat]

min\_2017\_install[0] = cat

if min\_2018\_install[1] > dict\_2018[cat]:

min\_2018\_install[1] = dict\_2018[cat]

min\_2018\_install[0] = cat

#print(max\_2016\_install)

#print(max\_2017\_install)

#print(max\_2018\_install)

#print(min\_2016\_install)

#print(min\_2017\_install)

#print(min\_2018\_install)

max\_install = [max\_2016\_install[1],max\_2017\_install[1],max\_2018\_install[1]]

min\_install = [min\_2016\_install[1],min\_2017\_install[1],min\_2018\_install[1]]

Years = ['2016','2017','2018']

pos = np.arange(len(Years))

bar\_width = 0.3

figure2 = plt.Figure(figsize=(8,4), dpi=85)

chart = figure2.add\_subplot(111)

Max\_bar = chart.bar(Years,max\_install,bar\_width,color='blue',edgecolor='black')

Min\_bar = chart.bar(pos+bar\_width,min\_install,bar\_width,color='pink',edgecolor='black')

chart.set\_ylabel("Download")

chart.set\_xlabel('Years')

figure2.suptitle('Max and Min download across 2016-17-18 years for a category',fontsize=18)

plt.legend(['max','min'],loc=10)

max\_month = [max\_2016\_install[0],max\_2017\_install[0],max\_2018\_install[0]]

min\_month = [min\_2016\_install[0],min\_2017\_install[0],min\_2018\_install[0]]

for idx,rect in enumerate(Max\_bar):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,max\_month[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(Min\_bar):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,min\_month[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=5,y=10)

String = """

Blue bar shows the maximum download

From the Analysis Installation of Game Category is increasing over the year

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=40,y=350)

root.mainloop()

"""Ques 6 END==========================================================================================================="""

"""Ques 6-part2 START==========================================================================================================="""

def function\_q6\_part2(event):

global screen

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='700',height=450,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=720

h=550

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Last Update",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

df = pd.read\_csv("DATA SET-2.csv")

#print(df.head(5))

#df.drop(9148,axis=0, inplace=True)

#df.drop(10472,axis=0,inplace=True)

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

#print((df['year'][5]))

#6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the

dict\_years = {}

for year in df['year'].unique():

dict\_years[year]=0

for index in range(len(df)):

dict\_years[df['year'][index]] += df['Installs'][index]

Years = []

list\_install = []

for year in dict\_years:

if year==2016 or year==2017 or year==2018:

Years.append(str(year))

list\_install.append(dict\_years[year])

pos = np.arange(len(Years))

bar\_width = 0.3

Years.reverse()

list\_install.reverse()

figure2 = plt.Figure(figsize=(8,4), dpi=85)

chart = figure2.add\_subplot(111)

chart.bar(Years,list\_install,bar\_width,color='blue',edgecolor='black')

#Min\_bar = chart.bar(pos+bar\_width,min\_install,bar\_width,color='pink',edgecolor='black')

chart.set\_ylabel("Years")

chart.set\_xlabel('Installs')

figure2.suptitle('Barchart on Installs on each Year ',fontsize=18)

#plt.legend(Month,loc=10)

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=5,y=10)

String = """

From the Analysis Installation of apps is Positively increasing from 2016-2018

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=40,y=360)

root.mainloop()

"""Ques 6-part2 END==========================================================================================================="""

"""Ques 16 START==========================================================================================================="""

def function\_q16(event):

global screen

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='800',height=450,bd=4,relief=RIDGE)

big\_frame.place(x=10,y=60)

w=820

h=550

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Last Update",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

df = pd.read\_csv("DATA SET-2.csv")

#print(df.head(0))

#print(df['Last Updated'].head(5))

d = pd.DatetimeIndex(df['Last Updated'])

df['year'] = d.year

df['month'] = d.month

# Data cleaning for "Installs" column

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

#print(df['Installs'].head(5))

df['Installs'] = pd.to\_numeric(df['Installs'])

#print(df['year'].unique())

#print(df['month'].unique())

dict\_year = {}

for year in df['year'].unique():

dict\_year[year] = {}

for month in df['month'].unique():

dict\_year[year][month] = 0

#print(dict\_year)

for index in range(len(df)):

if dict\_year[df['year'][index]][df['month'][index]] < df['Installs'][index]:

dict\_year[df['year'][index]][df['month'][index]] = df['Installs'][index]

else:

continue

#print(dict\_year)

count,label,text = [],[],[]

dict\_month = {1:"Jan",2:"Feb",3:"March",4:"April",5:"May",6:"June",7:"July",8:"Aug",9:"Sept",10:"Oct",11:"Nov",12:"Dec"}

for year in dict\_year:

max\_month = ""

max\_install = 0

for month in dict\_year[year]:

if max\_install < dict\_year[year][month]:

max\_install = dict\_year[year][month]

max\_month = month

#print("For {} best install is in {} month with {} installs".format(year,max\_month,max\_install))

count.append(max\_install)

label.append(str(year))

text.append(str(dict\_month[max\_month]))

"""

bar\_plot = plt.bar(label,count,tick\_label=label)

for idx,rect in enumerate(bar\_plot):

height = rect.get\_height()

plt.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,text[idx],ha='center', va='bottom', rotation=0)

"""

color = cm.rainbow(np.linspace(0, 1, 10))

figure2 = plt.Figure(figsize=(7,4), dpi=100)

chart = figure2.add\_subplot(111)

#x=['Dis-Liked Apps' , 'Liked Apps']

#y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

bar\_plot = chart.bar(label,count,color=color)

chart.set\_ylabel("Installs")

chart.set\_xlabel("Year")

figure2.suptitle("Month Indicating best downloads")

chart.legend()

for idx,rect in enumerate(bar\_plot):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,text[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=0,y=10)

String="The Google Play Store has 9 Year data out of which Maximum Download is seen in year 2018"

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=410)

plt.show()

"""Ques16 END==========================================================================================================="""

"""Ques11 START========================================================================================================="""

def install():

global Installs,sample

Installs=[]

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

return Installs

def dates\_str\_to\_int():

global sample

dates=sample['Last Updated']

year=[]

counter=0

for i in dates:

year.append([int(i[-8:-6]),month(i[:-9]),int(i[-4:])])

counter=counter+1

return year

def month(x):

if x[0:3]=='Jan':

return 1

elif x[0:3]=='Feb':

return 2

elif x[0:3]=='Mar':

return 3

elif x[0:3]=='Apr':

return 4

elif x[0:3]=='Ma' or x[0:3]=='May':

return 5

elif x[0:3]=='Jun':

return 6

elif x[0:3]=='Jul':

return 7

elif x[0:3]=='Aug':

return 8

elif x[0:3]=='Sep':

return 9

elif x[0:3]=='Oct':

return 10

elif x[0:3]=='Nov':

return 11

elif x[0:3]=='Dec':

return 12

def quat():

global sample

counter=0

temp1=[0,0,0,0,0,0,0,0,0,0,0,0]

temp2=[0,0,0,0,0,0,0,0,0,0,0,0]

temp3=[0,0,0,0,0,0,0,0,0,0,0,0]

temp4=[0,0,0,0,0,0,0,0,0,0,0,0]

temp5=[0,0,0,0,0,0,0,0,0,0,0,0]

temp6=[0,0,0,0,0,0,0,0,0,0,0,0]

temp7=[0,0,0,0,0,0,0,0,0,0,0,0]

temp8=[0,0,0,0,0,0,0,0,0,0,0,0]

temp9=[0,0,0,0,0,0,0,0,0,0,0,0]

ins=install()

counter=0

year=dates\_str\_to\_int()

for i in year:

if i[2]==2010:

if i[1]==1:

temp1[0]=temp1[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp1[1]=temp1[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp1[2]=temp1[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp1[3]=temp1[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp1[4]=temp1[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp1[5]=temp1[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp1[6]=temp1[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp1[7]=temp1[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp1[8]=temp1[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp1[9]=temp1[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp1[10]=temp1[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp1[11]=temp1[11]+ins[counter]

counter=counter+1

elif i[2]==2011:

if i[1]==1:

temp2[0]=temp2[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp2[1]=temp2[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp2[2]=temp2[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp2[3]=temp2[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp2[4]=temp2[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp2[5]=temp2[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp2[6]=temp2[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp2[7]=temp2[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp2[8]=temp2[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp2[9]=temp2[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp2[10]=temp2[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp2[11]=temp2[11]+ins[counter]

counter=counter+1

elif i[2]==2012:

if i[1]==1:

temp3[0]=temp3[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp3[1]=temp3[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp3[2]=temp3[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp3[3]=temp3[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp3[4]=temp3[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp3[5]=temp3[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp3[6]=temp3[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp3[7]=temp3[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp3[8]=temp3[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp3[9]=temp3[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp3[10]=temp3[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp3[11]=temp3[11]+ins[counter]

counter=counter+1

elif i[2]==2013:

if i[1]==1:

temp4[0]=temp4[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp4[1]=temp4[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp4[2]=temp4[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp4[3]=temp4[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp4[4]=temp4[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp4[5]=temp4[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp4[6]=temp4[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp4[7]=temp4[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp4[8]=temp4[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp4[9]=temp4[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp4[10]=temp4[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp4[11]=temp4[11]+ins[counter]

counter=counter+1

elif i[2]==2014:

if i[1]==1:

temp5[0]=temp5[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp5[1]=temp5[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp5[2]=temp5[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp5[3]=temp5[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp5[4]=temp5[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp5[5]=temp5[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp5[6]=temp5[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp5[7]=temp5[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp5[8]=temp5[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp5[9]=temp5[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp5[10]=temp5[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp5[11]=temp5[11]+ins[counter]

counter=counter+1

elif i[2]==2015:

if i[1]==1:

temp6[0]=temp6[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp6[1]=temp6[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp6[2]=temp6[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp6[3]=temp6[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp6[4]=temp6[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp6[5]=temp6[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp6[6]=temp6[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp6[7]=temp6[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp6[8]=temp6[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp6[9]=temp6[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp6[10]=temp6[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp6[11]=temp6[11]+ins[counter]

counter=counter+1

elif i[2]==2016:

if i[1]==1:

temp7[0]=temp7[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp7[1]=temp7[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp7[2]=temp7[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp7[3]=temp7[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp7[4]=temp7[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp7[5]=temp7[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp7[6]=temp7[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp7[7]=temp7[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp7[8]=temp7[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp7[9]=temp7[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp7[10]=temp7[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp7[11]=temp7[11]+ins[counter]

counter=counter+1

elif i[2]==2017:

if i[1]==1:

temp8[0]=temp8[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp8[1]=temp8[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp8[2]=temp8[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp8[3]=temp8[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp8[4]=temp8[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp8[5]=temp8[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp8[6]=temp8[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp8[7]=temp8[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp8[8]=temp8[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp8[9]=temp8[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp8[10]=temp8[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp8[11]=temp8[11]+ins[counter]

counter=counter+1

elif i[2]==2018:

if i[1]==1:

temp9[0]=temp9[0]+ins[counter]

counter=counter+1

elif i[1]==2:

temp9[1]=temp9[1]+ins[counter]

counter=counter+1

elif i[1]==3:

temp9[2]=temp9[2]+ins[counter]

counter=counter+1

elif i[1]==4:

temp9[3]=temp9[3]+ins[counter]

counter=counter+1

elif i[1]==5:

temp9[4]=temp9[4]+ins[counter]

counter=counter+1

elif i[1]==6:

temp9[5]=temp9[5]+ins[counter]

counter=counter+1

elif i[1]==7:

temp9[6]=temp9[6]+ins[counter]

counter=counter+1

elif i[1]==8:

temp9[7]=temp9[7]+ins[counter]

counter=counter+1

elif i[1]==9:

temp9[8]=temp9[8]+ins[counter]

counter=counter+1

elif i[1]==10:

temp9[9]=temp9[9]+ins[counter]

counter=counter+1

elif i[1]==11:

temp9[10]=temp9[10]+ins[counter]

counter=counter+1

elif i[1]==12:

temp9[11]=temp9[11]+ins[counter]

counter=counter+1

return temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8,temp9

def plotMyGraph(big\_frame,dict,limit1,limit2):

Quatmonth\_list = [[],[],[]]

Month1,Month2,Month3 = [],[],[]

for year in range(limit1,limit2+1):

#print(year)

Quatmonth\_list[0].append(dict[year][0][1])

Quatmonth\_list[1].append(dict[year][1][1])

Quatmonth\_list[2].append(dict[year][2][1])

Month1.append(dict[year][0][0])

Month2.append(dict[year][1][0])

Month3.append(dict[year][2][0])

Years = []

for i in range(limit1,limit2+1):

Years.append(str(i))

#Years=["2010","2011","2012","2013","2014","2015","2016","2017","2018"]

pos = np.arange(len(Years))

bar\_width = 0.3

#color = cm.rainbow(np.linspace(0, 1, 3))

figure2 = plt.Figure(figsize=(10,4), dpi=100)

chart = figure2.add\_subplot(111)

bar1 = chart.bar(Years,Quatmonth\_list[0],bar\_width,color='blue',edgecolor='black')

bar2 = chart.bar(pos+bar\_width,Quatmonth\_list[1],bar\_width,color='pink',edgecolor='black')

bar3 = chart.bar(pos+bar\_width\*2,Quatmonth\_list[2],bar\_width,color='red',edgecolor='black')

chart.set\_ylabel("Installs")

chart.set\_xlabel('Years')

figure2.suptitle('Group Barchart - Quater Month across the year',fontsize=18)

#plt.legend(Month,loc=10)

for idx,rect in enumerate(bar1):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month1[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(bar2):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month2[idx],ha='center', va='bottom', rotation=0)

for idx,rect in enumerate(bar3):

height = rect.get\_height()

chart.text(rect.get\_x() + rect.get\_width()/2., 1.05\*height,Month3[idx],ha='center', va='bottom', rotation=0)

canvas = FigureCanvasTkAgg(figure2, master=big\_frame)

canvas.get\_tk\_widget().place(x=0,y=100)

#Which quarter of which year has generated the highest number of install for each app used in the study?

String="In the above Graph Quarter of each Year with their Higher Installs are plotted"

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=520)

def function\_q11(event):

global screen

global sample,Installs

Installs = []

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='1050',height=580,bd=4,relief=RIDGE)

big\_frame.place(x=5,y=60)

tk.Label(big\_frame,text="For Better Visulization please select the range",font=("Calibri",15,'bold'),fg='#75acff',bg='white').place(x=350,y=0)

tk.Label(big\_frame,text="Starting Year:",font=("Calibri",13,'italic'),fg='#75acff',bg='white').place(x=150+100,y=45)

limit1 = tk.ttk.Combobox(big\_frame,values=["2010","2011","2012","2013","2014","2015","2016","2017","2018"])

limit1.place(x=240+100+25,y=49)

tk.Label(big\_frame,text="Ending Year:",font=("Calibri",13,'italic'),fg='#75acff',bg='white').place(x=400+100+50,y=45)

limit2 = tk.ttk.Combobox(big\_frame,values=["2010","2011","2012","2013","2014","2015","2016","2017","2018"])

limit2.place(x=490+100+50+25,y=49)

w=1070

h=670

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Last Update",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

sample=pd.read\_csv('DATA SET-2.csv')#reading data for the data set

sample=sample.replace(np.NaN,0)

sample.drop(index=[10472],inplace=True)

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

years = quat()

dict = []

for i in range(2019):

dict.append(i)

i=2010

for year in years:

indi = [0,0,0]

quat1 = year[0:3]

quat2 = year[3:6]

quat3 = year[6:9]

quat4 = year[9:12]

#print(quat1)

#print(quat2)

#print(quat3)

#print(quat4)

if sum(indi)<sum(quat1):

dict[i]=[]

indi = quat1

dict[i] = [["Jan",quat1[0]],["Feb",quat1[1]],["March",quat1[2]]]

if sum(indi)<sum(quat2):

print('')

dict[i]=[]

indi = quat2

dict[i] = [["April",quat2[0]],["May",quat2[1]],["June",quat2[2]]]

if sum(indi)<sum(quat3):

dict[i]=[]

indi = quat3

dict[i] = [["July",quat3[0]],["Aug",quat3[1]],["Sept",quat3[2]]]

if sum(indi)<sum(quat4):

dict[i]=[]

indi = quat4

dict[i] = [["Oct",quat4[0]],["Nov",quat4[1]],["Dec",quat4[2]]]

i+=1

plotMyGraph(big\_frame,dict,2010,2018)

tk.Button(big\_frame,text="SET GRAPH",bd=10,font=("Calibri",13,'italic'),fg='#75acff',bg='white',

command = lambda:plotMyGraph(big\_frame,dict,int(limit1.get()),int(limit2.get())) if (limit1.get()!=limit2.get() and limit1.get()!="" and limit2.get()!="" and int(limit1.get())<int(limit2.get()) ) else tk.messagebox.showerror("ERROR","PLEASE SELECT THE VALID RANGE")).place(x=490+100+50+25+165,y=40)

"""Ques11 END======================================================================================"""

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def overviewClick(event):

global screen

import InsightsDesign as over

over.startingScreen(screen)

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def sizeClick(event):

print('')

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(ws,hs))

window.configure(background='white')

def machineClick(event):

global screen

import InsightsDesignForMachineLearningModels as mac

mac.startingScreen(screen)

def startingScreen(root):

global screen,df

df=pd.read\_csv("DATA SET-2.csv")

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="LAST UPDATE",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_over = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_over.bind("<Button-1>",overviewClick)

lbl\_over.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame = tk.Frame(screen,bg='#F8E0E0',width='1520',height='730',bd=4,relief=RIDGE)

big\_frame.place(x=3,y=100)

"""

#11) Which quarter of which year has generated the highest number of install for each app used in the study?

#16) Which month(s) of the year , is the best indicator to the average downloads that an app will generate over the entire year?

6) For the years 2016,2017,2018 what are the category of apps that have got the most and the least downloads. What is the percentage increase or decrease that the

10) Across all the years ,which month has seen the maximum downloads fr each of the category. What is the ratio of downloads for the app that qualifies as teen versus mature17+

"""

#frame1 = tk.Frame(big\_frame,bg='white',width='800',height='400',bd=4,relief=RIDGE)

#frame1.place(x=3,y=3)

#tk.Label(frame1,text=,font=("Calibri",11,'bold'),fg='#910030',bg='white')

ques = """Which month of the year , is the best indicator to the average

downloads that an app will generate over the entire year?"""

q16 = tk.Label(big\_frame,text = ques,width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q16.bind("<Button-1>", function\_q16)

q16.place(x=50,y=10)

q11 = tk.Label(big\_frame,text = '''Which quarter of which year has generated the highest number of install for each

app used in the study?''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q11.bind("<Button-1>", function\_q11)

q11.place(x=750,y=10)

#function\_q11(frame1)

q6 = tk.Label(big\_frame,text = '''For the years 2016,2017,2018 what are the category of apps that have got the most

and the least downloads.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q6.bind("<Button-1>", function\_q6)

q6.place(x=50,y=250)

q6\_part2 = tk.Label(big\_frame,text = ''' What is the percentage increase or decrease that the

apps have got over the period of three years.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q6\_part2.bind("<Button-1>", function\_q6\_part2)

q6\_part2.place(x=50,y=490)

q10 = tk.Label(big\_frame,text = '''Across all the years ,which month has seen the maximum downloads fr each of the

category. What is the ratio of downloads for the app that qualifies as teen versus

mature17+''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q10.bind("<Button-1>", function\_q10)

q10.place(x=750,y=250)

screen.mainloop()

#startingScreen(tk.Tk())

Insights design to study review

# -\*- coding: utf-8 -\*-

"""

Created on Wed Jul 3 17:57:13 2019

@author: Dharmik joshi

"""

import pandas as pd

from matplotlib import pyplot as plt

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

import statsmodels.api as sm

import sys

import tkinter as tk

from tkinter import ttk

from tkinter import Listbox

from collections import OrderedDict

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.cm as cm

def list\_return(x,y,positive,negative,neutral,scroll1,scroll2,scroll3):

global data

positive.delete(0,'end')

negative.delete(0,'end')

neutral.delete(0,'end')

for i in range(len(data['Sentiment'])):

if data['App'][i][0:10]==y.get():

if data['Sentiment'][i]=='Positive':

positive.insert('end',data['Translated\_Review'][i])

elif data['Sentiment'][i]=='Negative':

negative.insert('end',data['Translated\_Review'][i])

elif data['Sentiment'][i]=='Neutral':

neutral.insert('end',data['Translated\_Review'][i])

scroll1.pack(side='right', fill='y' )

scroll2.pack(side='right', fill='y' )

scroll3.pack( side='right', fill='y' )

"""

positive.pack( side = 'left', fill = 'both' )

negative.pack( side = 'left', fill = 'both' )

neutral.pack( side = 'left', fill = 'both' )

"""

scroll1.config( command = positive.yview )

scroll2.config( command = negative.yview )

scroll3.config( command = neutral.yview )

def Sentiment\_list(positive,negative,neutral,list\_positive\_app,list\_negative\_app,list\_average\_app,scroll1,scroll2,scroll3):

positive.delete(0,'end')

negative.delete(0,'end')

neutral.delete(0,'end')

for i in range(40):

negative.insert('end',list\_negative\_app[i])

for i in range(50):

neutral.insert('end',list\_average\_app[i])

for i in range((50)):

positive.insert('end',list\_positive\_app[i][0:15])

scroll1.pack(side='right', fill='y' )

scroll2.pack(side='right', fill='y' )

scroll3.pack( side='right', fill='y' )

scroll1.config( command = positive.yview )

scroll2.config( command = negative.yview )

scroll3.config( command = neutral.yview )

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 2 21:46:07 2019

@author: Dharmik joshi

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

#import Tkinter as tk

#import ttk

def overviewClick(event):

global screen

import InsightsDesign as over

over.startingScreen(screen)

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def sizeClick(event):

print('')

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(1180,800))

window.resizable(False,False)

window.configure(background='white')

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def machineClick(event):

global screen

import InsightsDesignForMachineLearningModels as mac

mac.startingScreen(screen)

def startingScreen(root):

global screen,data

root.destroy()

data=pd.read\_csv('DATA SET-1.csv')

data=data.replace(np.nan,'Not Available')

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="REVIEW",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_over = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_over.bind("<Button-1>",overviewClick)

lbl\_over.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame = tk.Frame(screen,bg='#003b6b',width='1400',height='730',bd=4,relief=RIDGE)

big\_frame.place(x=3,y=100)

Apps=list(OrderedDict.fromkeys(data['App']))

canvas=[]

filtered=[]

for i in Apps:

filtered.append(i[0:10])

for i in range(4):

can=tk.Canvas(big\_frame,width=320,height=720,bg='#003b6b')

canvas.append(can)

can.grid(row=0,column=i)

combo=ttk.Combobox(big\_frame,values=filtered,state="readonly")

combo.place(x=650,y=20)

"""

for i in range(0,4):

can=tk.Canvas(canvas[i],width=280,height=600,bg='#EFEFFB')

canvas.append(can)

can.place(x=0,y=110)

"""

#labels for interface

l=tk.Label(big\_frame,text='Select App',bg='#003b6b')

l.config(font=("Calibri", 16,"bold"),fg="orange")

l.place(x=540,y=14)

l=tk.Label(big\_frame,text='Positive',bg='#003b6b')

l.config(font=("Calibri", 16,'bold'),fg='yellow')

l.place(x=430,y=45)

l=tk.Label(big\_frame,text='Neutral',bg='#003b6b')

l.config(font=("Calibri", 16,'bold'),fg='yellow')

l.place(x=700,y=45)

l=tk.Label(big\_frame,text='Negative',bg='#003b6b')

l.config(font=("Calibri", 16,'bold'),fg='yellow')

l.place(x=950,y=45)

scroll1=tk.Scrollbar(canvas[1])

scroll2=tk.Scrollbar(canvas[2])

scroll3=tk.Scrollbar(canvas[3])

positive=tk.Listbox(canvas[1],yscrollcommand = scroll1.set,height=35,width=45,bg='#F5A9BC')

negative=tk.Listbox(canvas[2],yscrollcommand = scroll2.set,height=35,width=43,bg='#F7F8E0')

neutral=tk.Listbox(canvas[3],yscrollcommand = scroll3.set,height=35,width=45,bg='#F8E0F1')

positive.pack(side = 'left', fill = 'both')

negative.pack( side = 'left', fill = 'both' )

neutral.pack( side = 'left', fill = 'both' )

pos\_box = positive

neg\_box = negative

neu\_box = neutral

tk.Button(big\_frame,text='Review',bd=5,width=12,bg='#c90000',font=("Calibri",12,'italic'),

command=lambda:list\_return(canvas,combo,pos\_box,neg\_box,neu\_box,scroll1,scroll2,scroll3)).place(x=810,y=10)

#Sentiment Analysis

list\_of\_apps\_most\_positive\_sentiments = []

list\_of\_apps\_most\_negative\_sentiments = []

list\_of\_apps\_most\_average\_sentiments = []

list\_of\_apps\_most\_zero\_sentiments = []

Sample\_Data = pd.read\_csv("DATA SET-1.csv")

dict\_app = {}

for app in Sample\_Data['App']:

if app in dict\_app:

dict\_app[app]+=1

else:

dict\_app[app]=1

app = []

for i in dict\_app.keys():

app.append(i)

dict\_sentiment = {}

for i in app:

dict\_sentiment[i]=[0,0]

for a in range(len(Sample\_Data['App'])):

app\_name = Sample\_Data['App'][a]

app\_sentiment = Sample\_Data['Sentiment'][a]

if app\_name in dict\_sentiment:

if app\_sentiment == "Positive":

dict\_sentiment[app\_name][0]+=1

elif app\_sentiment == "Negative":

dict\_sentiment[app\_name][1]+=1

else:

if app\_sentiment == "Positive":

dict\_sentiment[app\_name][0]=1

elif app\_sentiment == "Negative":

dict\_sentiment[app\_name][1]=1

for app\_name in dict\_sentiment:

app = app\_name

positive = dict\_sentiment[app\_name][0]

negative = dict\_sentiment[app\_name][1]

ratio1 = positive / dict\_app[app]

ratio2 = negative / dict\_app[app]

if(positive-negative<3 and positive-negative>-3)and(positive!=0):

#avg\_sentiment[app\_name]=positive

list\_of\_apps\_most\_average\_sentiments.append(app\_name)

elif(ratio1>ratio2):

list\_of\_apps\_most\_positive\_sentiments.append(app\_name)

#print("pos ",positive)

#print("neg ",negative)

elif(ratio1<ratio2):

list\_of\_apps\_most\_negative\_sentiments.append(app\_name)

else:

list\_of\_apps\_most\_zero\_sentiments.append(app\_name)

y\_count=[len(list\_of\_apps\_most\_positive\_sentiments),len(list\_of\_apps\_most\_negative\_sentiments),len(list\_of\_apps\_most\_average\_sentiments)]#,len(list\_of\_apps\_most\_zero\_sentiments)]

x\_label=['Positive','Negative','Average']#,'NONE']

text = """--SENTIMENT ANALYSIS--"""

tk.Label(canvas[0],text=text,font=("Calibri",20,"bold"),fg="yellow",bg='#003b6b').place(x=10,y=100)

text = """COUNT OF SENTIMENTS"""

tk.Label(canvas[0],text=text,font=("Calibri",20,"bold"),fg="yellow",bg='#003b6b').place(x=10,y=150)

tk.Label(canvas[0],text="Positive Sentiment : "+str(len(list\_of\_apps\_most\_positive\_sentiments)),font=("Calibri",15,"bold"),fg="yellow",bg='#003b6b').place(x=10,y=100+100)

tk.Label(canvas[0],text="Negative Sentiment : "+str(len(list\_of\_apps\_most\_negative\_sentiments)),font=("Calibri",15,"bold"),fg="yellow",bg='#003b6b').place(x=10,y=100+150)

tk.Label(canvas[0],text="Neutral Sentiment : "+str(len(list\_of\_apps\_most\_average\_sentiments)),font=("Calibri",15,"bold"),fg="yellow",bg='#003b6b').place(x=10,y=100+200)

tk.Button(canvas[0],text="CLICK TO GET APPS",bd=10,font=("Calibri",15,"bold"),fg="yellow",bg='#003b6b',command = lambda: Sentiment\_list(pos\_box,neg\_box,neu\_box,list\_of\_apps\_most\_positive\_sentiments,list\_of\_apps\_most\_negative\_sentiments,list\_of\_apps\_most\_average\_sentiments,scroll1,scroll2,scroll3)).place(x=10,y=100+200+50)

figure1 = plt.Figure(figsize=(5,4), dpi=50)

color = cm.rainbow(np.linspace(0, 1, 3))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(y\_count, labels=x\_label,colors = color, autopct='%1.1f%%', startangle=200)

ax3.set\_title("Pie chart on Category")

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, canvas[0])

pie\_plot.get\_tk\_widget().place(x=20,y=450)

#ax3.legend(loc=0)

screen.mainloop()

#startingScreen(tk.Tk())

Insights design for study of ML model

# -\*- coding: utf-8 -\*-

"""

Created on Wed Jul 3 20:47:35 2019

@author: Dharmik joshi

"""

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 2 21:46:07 2019

@author: Dharmik joshi

"""

from sklearn.impute import SimpleImputer

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

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns #statistical analysis

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

from sklearn import metrics #used to find the accuracy

from sklearn import tree

from sklearn.impute import SimpleImputer

import math

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

#import Tkinter as tk

#import ttk

import pandas as pd

import sys

import seaborn as sns

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from sklearn.impute import SimpleImputer

from PIL import Image,ImageTk

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

from sklearn.linear\_model import LinearRegression #for all the formulas of linear regression formuls

from sklearn.metrics import r2\_score

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression #for all the formulas of linear regression formuls

from sklearn.metrics import r2\_score

import statsmodels.api as sm

"""Ques 13 START============================================================================================="""

"""

Study and find out the relation between the Sentiment-polarity and sentiment-subjectivity of all the apps.

"""

def newRelation1(app,x,y):

global dict\_app\_relation

for i in x:

if i==-999:

x.remove(i)

y.remove(i)

if x==[] or y==[]:

return

data = pd.DataFrame({'Sentiment\_pol':y , 'Sentiment\_sub': x})

val = data['Sentiment\_pol'].corr(data['Sentiment\_sub'])

dict\_app\_relation[app] = val

def function\_q13(event):

global screen,df,dict\_app\_relation

dict\_app\_relation={}

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='700',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=50,y=60)

w=800

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df = pd.read\_csv("DATA SET-1.csv")

df=df.replace(np.NaN,-999)

dict\_app\_index\_count={}

for index in range(len(df['App'])):

app = df['App'][index]

if app in dict\_app\_index\_count:

dict\_app\_index\_count[app][1]+=1

else:

dict\_app\_index\_count[app]=[index,1]

# after this for loop dict\_app\_index\_count will hold the app name as key and it's first index in data set and total count in data set as item

for app in dict\_app\_index\_count:

index = dict\_app\_index\_count[app][0]

count = dict\_app\_index\_count[app][1]

sub,pol=[],[]

for i in range(count):

i = index+i

sub.append(df['Sentiment\_Subjectivity'][i])

pol.append(df['Sentiment\_Polarity'][i])

newRelation1(app,sub,pol)

app\_no = np.arange(len(dict\_app\_relation.keys()))

relation = []

for i in dict\_app\_relation:

relation.append(dict\_app\_relation[i])

figure3 = plt.Figure(figsize=(6,4), dpi=100)

ax3 = figure3.add\_subplot(111)

ax3.scatter(app\_no,relation, color = 'g')

scatter3 = FigureCanvasTkAgg(figure3, big\_frame)

scatter3.get\_tk\_widget().place(x=50,y=45)

ax3.legend()

ax3.set\_xlabel("Applications in sequence")

ax3.set\_ylabel("Correlation")

ax3.set\_title("correlation between Polarity and Subjectivity for all apps")

String = """

In this Scatter plot each point represent the correlation

between sentiment polarity and sentiment subjectivity for

all Application present in Data set from the Sentiment Analysis

Most of apps have positive relation with between sentiment polarity and subjectivity

"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=420)

root.mainloop()

"""Ques 13 END============================================================================================="""

"""Ques9 START============================================================================================="""

""""

All those apps who habve managed to get over 1,00,000 downloads, have they managed

to get an average rating of 4.1 and above?

An we conclude something in co-relation to the number of downloads and the ratings received.

"""

def getResult(rating,installs,big\_frame):

df = pd.read\_csv("DATA SET-2.csv")

# Replace "NaN" with mean

imputer = SimpleImputer()

df['Rating'] = imputer.fit\_transform(df[['Rating']])

temp = []

for index in range(len(df['Rating'])):

if df['Rating'][index] >= rating:

temp.append(1)

else:

temp.append(0)

cat\_rating= pd.DataFrame(zip(temp,temp),columns=["cat\_Ratings","ignore"])

df = pd.concat([df,cat\_rating],axis=1)

df.drop("ignore",axis=1,inplace=True)

df.drop(df.index[9148], inplace=True)

# Data cleaning for "Installs" column

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

rating\_sum = 0

rate=[]

#1169

for index in range(len(df)):

try:

if df['Installs'][index]>=installs:

if df['Rating'][index]>=rating:

rate.append(1)

else:

rate.append(0)

rating\_sum+=df['Rating'][index]

except:

#print(index)

continue

#print(len(rate))

avg\_rating = (rating\_sum/len(rate))

#print(df['Installs'].corr(df['Rating']))

val = "Yes" if (rating\_sum/len(rate))>=rating else "No"

rel = "Greater than" if val == "Yes" else "Lesser than"

fig, ax = plt.subplots(figsize=(10, 10))

l1 ='>={}'.format(rating)

l2 ='<{}'.format(rating)

size=[rate.count(1),rate.count(0)]

label = [l1,l2]

title = 'Count of {}'.format(rating)

figure1 = plt.Figure(figsize=(5,5), dpi=70)

#color = cm.rainbow(np.linspace(0, 1, 10))

#fig1, ax1 = plt.subplots()

ax3 = figure1.add\_subplot(111)

ax3.pie(size, labels=label,colors = ['red','blue'], autopct='%1.1f%%', startangle=200)

ax3.set\_title(title)

#ax3.xlim(0,3.0)

pie\_plot = FigureCanvasTkAgg(figure1, big\_frame)

pie\_plot.get\_tk\_widget().place(x=80,y=190)

tk.Label(big\_frame,text="--Results--",font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=220,y=470)

String = "Average rating of all the apps who managed to get over {} download is {:.1f}".format(installs,avg\_rating)

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=500)

String ="""{}! All those apps who have managed to get over {} downloads ,

they have to get an average rating of {:.1f} which is {} than {} """.format(val,installs,avg\_rating,rel,rating)

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=0,y=530)

#ax3.legend(loc=0)

def function\_q9(event):

global screen,df

df = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='600',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=50,y=60)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

String=""" find the avearge rating of given installs is greater than or equal

to given minimum rating or not?"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=0)

tk.Label(big\_frame,text="Rating : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=0,y=70)

rating = tk.StringVar()

tk.Entry(big\_frame,textvar=rating,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=110,y=65)

tk.Label(big\_frame,text="Installs : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=280,y=70)

installs = tk.StringVar()

tk.Entry(big\_frame,textvar=installs,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=400,y=65)

tk.Button(big\_frame,text="RESULT",width=13,bd=10,font=("Calibri",16,'bold'),fg='#75acff',bg='white',command = lambda : getResult(float(rating.get()),int(installs.get()),big\_frame) if(rating.get()!="" and installs.get()!="") else tk.messagebox.showerror("ERROR","PLEASE FILL ALL THE DETAILS")).place(x=220,y=130)

root.mainloop()

"""Ques9 END============================================================================================="""

"""Ques8 START============================================================================================="""

#try Aroil model for prediction

def value(x):

global And,val

val+=1

And[str(x)]=val

return val

def getPrediction(big\_frame):

global rating,size,installs,price,type,android,df

df = pd.read\_csv("DATA SET-2.csv")

category = {'SPORTS':0,'ENTERTAINMENT':1,'SOCIAL':2,'NEWS\_AND\_MAGAZINES':3,'EVENTS':4,'TRAVEL\_AND\_LOCAL':5,'GAME':6}

for index in range(len(df['Category'])):

if df['Category'][index] in category:

continue

else:

df.drop(index,axis=0,inplace=True)

df['Category'] = df['Category'].map(lambda x : category[x] if(x in category) else -1)

dict\_content\_rating = {"Adults only 18+": 0, "Everyone": 1, "Everyone 10+": 2, "Mature 17+": 3, "Teen": 4}

df['Content Rating NUM'] = df['Content Rating'].map(lambda x : dict\_content\_rating[x] if(x in dict\_content\_rating) else -1 )

# Data cleaning for "Size" column

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

df['Price'] = df['Price'].map(lambda x: x if x==0 else x.lstrip('$').rstrip())

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

# Change datatype

df['Reviews'] = pd.to\_numeric(df['Reviews'])

df['Installs'] = pd.to\_numeric(df['Installs'])

df['Price'] = pd.to\_numeric(df['Price'])

# Replace "NaN" with mean

imputer = SimpleImputer()

df['Rating'] = imputer.fit\_transform(df[['Rating']])

# Rounding the mean value to 1 decimal place

df['Rating'].round(1)

df.dropna(axis=0, inplace=True)

#sns.heatmap(df.isnull())

df['Type'] = df['Type'].map(lambda x : 1 if(x=="Free") else 0)

global And,val

And = {}

val = -1

df['Android Ver'] = df['Android Ver'].map(lambda x : And[str(x)] if(str(x) in And) else value(x))

# Features selection

features = ['Rating', 'Size', 'Installs', 'Price', 'Type','Android Ver']

#Spliting the datat fro training and testing

train,test=train\_test\_split(df,test\_size = 0.3)

#creating a response and target variable

#taking the training data input

train\_x = train[features] #multiple indepent variable

train\_y = train['Category'] #only one dependent variable

##print(list(train.columns))

train,test=train\_test\_split(df,test\_size = 0.3)

#train,test=train\_test\_split(df,test\_size = 0.2)

#taking the testing data input

test\_x = test[features]

test\_y = test['Category']

##print(list(test.columns))

"""

#Creating a decision tree model based on the training data

model = tree.DecisionTreeClassifier()

model.fit(train\_x,train\_y)

#now prediction using the trained model

prediction = model.predict(test\_x)

#now displaying the predicted vs actual values

#dataframe = pd.DataFrame(prediction,test\_y)

"""

#idea of random forest to improve efficiency #will create a small different trees

""" RANDOM FOREST """

model = RandomForestClassifier(n\_estimators = 100) #this will create the group of 100 data

model.fit(train\_x,train\_y)

prediction = model.predict(test\_x)

#now displaying the predicted vs actual values

#print(metrics.accuracy\_score(prediction,test\_y))

#print(classification\_report(test\_y , prediction))

rating\_app = float(rating.get())

size\_app = float(size.get())

installs\_app = int(installs.get())

if type == "Free":

price\_app = 0

else:

price\_app = int(price.get())

if type=="Free":

type\_app = 1

else:

type\_app=0

android\_app = int(And[android.get()])

prediction = model.predict(np.array([rating\_app,size\_app,installs\_app,price\_app,type\_app,android\_app]).reshape(1,-1))

#print(prediction)

#print(model.score(test\_x,test\_y))

#print(category)

val = ""

for val in category:

if category[val] == prediction:

#print(val)

break

tk.Label(big\_frame,text="-----RESULT-----",height='2',font=("Calibri",19,'bold'),fg='#ad023e',bg='white').place(x=250,y=400)

#print(val)

string = "With help of parameters {} category is most likely to be downloaded in comming years".format(val)

tk.Label(big\_frame,text=string,height='2',font=("Calibri",10,'italic'),fg='#ad023e',bg='white').place(x=0,y=450)

string = "Accuracy score for this model is {:.2f}%".format(model.score(test\_x,test\_y)\*100)

tk.Label(big\_frame,text=string,height='2',font=("Calibri",11,'italic'),fg='#ad023e',bg='white').place(x=0,y=500)

def prediction(big\_frame):

global rating,size,installs,price,type,android

if rating.get()==" " or size.get()==" " or installs.get()==" " or price.get()==" " or type.get()=="--Select Type--" or android.get()=="--Select Android Ver--":

#print("1")

tk.messagebox.showerror("ERROR","PLEASE FILL ALL THE ENTRY FIELDS")

else:

if 0.0>=float(rating.get())>=5.0:

tk.messagebox.showerror("ERROR","RATING IS NOT APPROPRIATE")

else:

try:

getPrediction(big\_frame)

except:

tk.messagebox.showerror("ERROR","PLEASE OPEN NEW WINDOW")

"""Amongst sports, entertainment,social media,news,events,travel and games,

which is the category of app that is most likely to be downloaded in the

coming years, kindly make a prediction and back it with suitable findings."""

def function\_q8(event):

global screen,df

root = Toplevel(screen)

#root.wm\_attributes('-alpha', 0.7)

big\_frame = tk.Frame(root,bg='white',width='610',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=40,y=60)

#background = ImageTk.PhotoImage(Image.open("background.jpeg"))

#back = tk.Label(big\_frame,image=background)

#back.place(x=0,y=0)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

""""PARAMETERS are['Rating', 'Size', 'Installs', 'Price', 'Content Rating NUM','Type','Android Ver']

Validations are rating must be a float value and in range of 0.0 - 5.0

Size must be a int value

Installs provide a

"""

global rating,size,installs,price,type,android

tk.Label(root,text="Random",width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

tk.Label(big\_frame,text="IN ORDER TO PREDICT THE CATEGORY PLEASE PROVIDE THE FOLLOWING PARAMETER VALUES",

height=2,font=("Helvetica",9,'italic'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").place(x=10,y=0)

tk.Label(big\_frame,text="Rating : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=0,y=70)

rating = tk.StringVar()

tk.Entry(big\_frame,textvar=rating,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=110,y=65)

tk.Label(big\_frame,text="Size : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=300,y=70)

size = tk.StringVar()

tk.Entry(big\_frame,textvar=size,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=400,y=65)

tk.Label(big\_frame,text="Installs : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=0,y=140)

installs = tk.StringVar()

tk.Entry(big\_frame,textvar=installs,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=110,y=135)

tk.Label(big\_frame,text="Type : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=300,y=140)

type = tk.ttk.Combobox(big\_frame,values=["Free","Paid"],state="readonly",width=15,height=25,font=("Calibri",13,'italic'))#,width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

type.set("--Select Type--")

type.place(x=400,y=135)

tk.Label(big\_frame,text="Price : ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=0,y=210)

price = tk.StringVar()

tk.Entry(big\_frame,textvar=price,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=110,y=210)

annd = []

for i in df['Android Ver'].unique():

annd.append(i)

tk.Label(big\_frame,text="Android Ver: ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=275,y=210)

android = tk.ttk.Combobox(big\_frame,values=annd,state="readonly",width=15,height=25,font=("Calibri",13,'italic'))#,width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

android.set("--Select Android Ver--")

android.place(x=410,y=210)

"""

tk.Label(big\_frame,text="Android Ver: ",width=13,height='1',font=("Calibri",16,'italic'),fg='#05b8ff',bg='white').place(x=275,y=210)

android = tk.StringVar()

tk.Entry(big\_frame,textvar=rating,width=13,bd=10,font=("Calibri",16,'italic'),fg='#75acff',bg='white').place(x=400,y=205)

"""

tk.Button(big\_frame,text="Prediction",width=13,bd=10,font=("Calibri",16,'bold'),fg='#75acff',bg='white',command = lambda:prediction(big\_frame)).place(x=240,y=280)

tk.Label(big\_frame,text="This Prediction will take some time kindly wait!!",height='2',font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=150,y=350)

root.mainloop()

"""Ques8 END============================================================================================="""

"""Ques17 START============================================================================================="""

#Does the size of the App influence the number of installs that it gets ? if,yes the trend is positive or negative with the increase in the app size.

def function\_q17(event):

global screen,df

df = pd.read\_csv("DATA SET-2.csv")

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='600',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=50,y=60)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

df.drop(9148,axis=0, inplace=True)

df.drop(10472,axis=0,inplace=True)

#print(df['Installs'].head(5))

df['Installs'] = df['Installs'].map(lambda x: x.rstrip('+'))

df['Installs'] = df['Installs'].map(lambda x: ''.join(x.split(',')))

df['Installs'] = pd.to\_numeric(df['Installs'])

# Data cleaning for "Size" column

df['Size'] = df['Size'].map(lambda x: x.rstrip('M'))

df['Size'] = df['Size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)

df['Size'] = df['Size'].map(lambda x: np.nan if x.startswith('Varies') else x)

df['Size'] = pd.to\_numeric(df['Size'])

# Replace "NaN" with mean

imputer = SimpleImputer()

df['Size'] = imputer.fit\_transform(df[['Size']])

df['Installs'] = imputer.fit\_transform(df[['Installs']])

#now creating linear approximation

x = df['Size'].values.reshape(-1,1) # this reshape wil converts the data into the specific format in which fit function is required

y = df['Installs'].values.reshape(-1,1)

reg=LinearRegression()

reg.fit(x,y)

#reg.coef\_calculates slope , reg.intercept\_calculates 'C'

#print(reg.coef\_)

#print(reg.score(x,y))

#now creating prediction

prediction = reg.predict(x)

#now assesing efficiency using R-squared model

x = df['Size']

y = df['Installs']

x2 = sm.add\_constant(x) #sci-kit is used to eliminate the value of x because x is indipndent variable

#Ordinary least squares is the simplest and most common estimator in which the two \(\beta\)s are chosen to minimize the square of the distance between

est = sm.OLS(y,x2)

est2 = est.fit()

#print( est2.summary())

figure3 = plt.Figure(figsize=(5,4), dpi=100)

ax3 = figure3.add\_subplot(111)

ax3.scatter(df['Size'],df['Installs'], color = 'y')

ax3.plot(df['Size'],prediction,color='r')

scatter\_plot = FigureCanvasTkAgg(figure3, big\_frame)

scatter\_plot.get\_tk\_widget().place(x=50,y=20)

ax3.legend()

ax3.set\_xlabel("Size of the App")

ax3.set\_ylabel("Installs")

ax3.set\_title("Trend of Install")

String = """ Conclusion : -

Here we have applied Linear Regression to find the Trend

As we can observe from above graph There is a Positive Trend

From the trend as increase in the size of App influence the

number of installs"""

tk.Label(big\_frame,text=String,font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=450)

root.mainloop()

"""Ques17 END============================================================================================="""

"""Ques15 START============================================================================================="""

#For Ques 15

def setAppSentiment(app,sentiment):

global dict\_app\_sent

p=sentiment.count(1)

n=sentiment.count(0)

if(p>n):

dict\_app\_sent[str(app)]=1

else:

dict\_app\_sent[str(app)]=0

#Ques 15

def result(app,root):

global dict\_app\_sent,df

for a in df['App'].unique():

if (str(a)[0:10]+"...") == app:

if dict\_app\_sent[str(a).lower()] == 1:

sentiment = "LIKE"

else:

sentiment = "DISLIKE"

break

#print(dict\_app\_sent)

try:

String = "Name of choosen app is '{}' User will {} this App".format(str(a),str(sentiment))

except:

String = "App Name is can not be identified"

tk.Label(root,text=String,height='2',font=("Calibri",13,'italic'),fg='#ad023e',bg='white').place(x=10,y=560)

def function\_q15(event):

global screen

root = Toplevel(screen)

big\_frame = tk.Frame(root,bg='white',width='600',height='630',bd=4,relief=RIDGE)

big\_frame.place(x=50,y=60)

w=700

h=700

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

x=(ws/2)-(w/2)

y=(hs/2)-(h/2)

root.geometry("%dx%d+%d+%d"%(w,h,x,y))

root.configure(background='white')

tk.Label(root,text="",bg='white').pack()

tk.Label(root,text="Is it advisable to launch an app like ’10 Best foods for you’? Do the users like these apps?",

width=500,height=2,font=("Helvetica",11,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

"""For this Question will use the Sentiment Analysis provided in data set"""

"""======================================================================"""

global dict\_app\_sent

dict\_app\_sent={}

df = pd.read\_csv("DATA SET-1.csv")

#print(df.isnull().sum())

df['Sentiment'] = df['Sentiment'].map(lambda x : 1 if x=='Positive' else 0)

dict\_app\_index\_count={}

for index in range(len(df['App'])):

app = df['App'][index]

if app in dict\_app\_index\_count:

dict\_app\_index\_count[str(app)][1]+=1

else:

dict\_app\_index\_count[str(app)]=[index,1]

for app in dict\_app\_index\_count:

index = dict\_app\_index\_count[str(app)][0]

count = dict\_app\_index\_count[str(app)][1]

app = app.lower()

pol = []

for i in range(count):

i = index+i

pol.append(df['Sentiment'][index])

setAppSentiment(str(app),pol)

"""======================================================================"""

"""in this label i have to plot a count plot"""

like\_dislike\_list = []

for i in dict\_app\_sent:

like\_dislike\_list.append(dict\_app\_sent[i])

"""THIS CODE WILL PRINT THE GRAPH"""

figure = plt.Figure(figsize=(5,5), dpi=100)

chart = figure.add\_subplot(111)

x=['Dis-Liked Apps' , 'Liked Apps']

y=[like\_dislike\_list.count(0),like\_dislike\_list.count(1)]

chart.bar(x,y, color=['red','green'])

chart.set\_ylabel("Count App's")

figure.suptitle("Count of Liked and Disliked apps")

chart.legend()

canvas = FigureCanvasTkAgg(figure, master=big\_frame)

canvas.get\_tk\_widget().place(x=50,y=45)

canvas.draw()

tk.Label(big\_frame,text="Select an App : ",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='white').place(x=40,y=515)

app\_var = tk.StringVar()

App\_name = []

for app in df['App'].unique():

App\_name.append(str(app)[0:10]+"...")

combo = tk.ttk.Combobox(big\_frame,values=App\_name,state="readonly",width=25,height=25,font=("Calibri",13,'italic'))#,width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

combo.set("--Select App--")

combo.place(x=180,y=525)

tk.Button(big\_frame,text="CHECK",width=10,height=1,bd=4,font=("Calibri",12,'bold'),fg='#75acff',bg='#e1e5eb',command = lambda:result(combo.get(),big\_frame) if combo.get()!="--Select App--" else tk.messagebox.showerror("ERROR","APP NAME IS NOT FOUND")).place(x=440,y=520)

root.mainloop()

"""Ques15 END============================================================================================="""

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def overviewClick(event):

global screen

import InsightsDesign as over

over.startingScreen(screen)

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def sizeClick(event):

print('')

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(ws,hs))

window.configure(background='white')

def startingScreen(root):

global screen,df

df=pd.read\_csv('DATA SET-2.csv')

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="LEARNING MODELS",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_over = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_over.bind("<Button-1>",overviewClick)

lbl\_over.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

#lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame = tk.Frame(screen,bg='#F8E0E0',width='1520',height='730',bd=4,relief=RIDGE)

big\_frame.place(x=3,y=100)

q15 = tk.Label(big\_frame,text = '''Is it advisable to launch an app like ’10 Best foods for you’? Do the users like these

apps?''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q15.bind("<Button-1>", function\_q15)

q15.place(x=50,y=10)

q17 = tk.Label(big\_frame,text = '''Does the size of the App influence the number of installs that it gets ? if,yes the

trend is positive or negative with the increase in the app size.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q17.bind("<Button-1>", function\_q17)

q17.place(x=750,y=10)

q8 = tk.Label(big\_frame,text = '''Amongst sports, entertainment,social media,news,events,travel and games,which

is the category of app that is most likely to be downloaded in the coming years,

kindly make a prediction and back it with suitable findings.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q8.bind("<Button-1>", function\_q8)

q8.place(x=50,y=250)

q13 = tk.Label(big\_frame,text = '''Study and find out the relation between the Sentiment-polarity and

sentimentsubjectivity of all the apps.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q13.bind("<Button-1>", function\_q13)

q13.place(x=750,y=250)

q9 = tk.Label(big\_frame,text = '''All those apps who habve managed to get over 1,00,000 downloads, have they

managed to get an average rating of 4.1 and above? An we conclude something in

co-relation to the number of downloads and the ratings received.''',width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE)

q9.bind("<Button-1>", function\_q9)

q9.place(x=50,y=500)

#tk.Label(big\_frame,text = "Feature 5",width=70,height=8,font=("Calibri",13,'bold'),bd=10,fg='#75acff',bg='powder blue',relief=RIDGE).place(x=750,y=500)

screen.mainloop()

#startingScreen(tk.Tk())

Insights design for Search app

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jul 2 21:46:07 2019

@author: Dharmik joshi

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#import pypdfocr.pypdfocr\_gs as pdfImg

from PIL import Image, ImageTk

#import Tkinter as tk

#import ttk

import pandas as pd

from matplotlib import pyplot as plt

import numpy as np

import sys

from collections import OrderedDict

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from PIL import Image, ImageTk

def month(x):

if x[0:3]=='Jan':

return 1

elif x[0:3]=='Feb':

return 2

elif x[0:3]=='Mar':

return 3

elif x[0:3]=='Apr':

return 4

elif x[0:3]=='Ma' or x[0:3]=='May':

return 5

elif x[0:3]=='Jun':

return 6

elif x[0:3]=='Jul':

return 7

elif x[0:3]=='Aug':

return 8

elif x[0:3]=='Sep':

return 9

elif x[0:3]=='Oct':

return 10

elif x[0:3]=='Nov':

return 11

elif x[0:3]=='Dec':

return 12

def install():

global sample

Installs=[]

for i in sample['Installs']: #converting string based installs into integer based

if i=='Free':

Installs.append(0)

else:

Installs.append(int(i.replace('+','').replace(',','')))

return Installs

def dates\_str\_to\_int():

global sample

dates=sample['Last Updated']

year=[]

counter=0

for i in dates:

year.append([int(i[-8:-6]),month(i[:-9]),int(i[-4:])])

counter=counter+1

return year

def display(x,y,z):

for i in x:

for j in set(i):

y.insert('end',j)

def filtering(value,canvas\_listbox):

global sample

installs=install()

year=dates\_str\_to\_int()

rating=sample['Rating']

genre=sample['Genres'].unique()

ans=[]

for i in genre:

ans.append([])

for i in range(len(installs)):

if i!=10472 and installs[i]==value[0]:

if rating[i]>=value[1]:

if year[i][2]==value[2]:

for j in range(len(genre)):

if genre[j]==sample['Genres'][i]:

ans[j].append(sample['App'][i])

canvas\_listbox.delete(0,'end')

display(ans,canvas\_listbox,genre)

def getting(install,rating,year,canvas\_listbox):

if install.get().strip()!='' and rating.get().strip()!='' and year.get().strip()!='':

value=[int(install.get().replace(',','').replace('+','')),float(rating.get()),int(year.get())]

filtering(value,canvas\_listbox)

else:

tk.messagebox.showerror('Error','Please select values')

def reviewClick(event):

global screen

import InsightsDesignToStudyReview as rev

rev.startingScreen(screen)

def overviewClick(event):

global screen

import InsightsDesign as over

over.startingScreen(screen)

def categoryClick(event):

global screen

import InsightsDesignForCategory as cat

cat.startingScreen(screen)

def installClick(event):

global screen

import InsightsDesignForInstall as inst

inst.startingScreen(screen)

def searchAppClick(event):

global screen

import InsightsDesignForSearchApp as app

app.startingScreen(screen)

def sizeClick(event):

print('')

def lastupdateClick(event):

global screen

import InsightsDesignForLastUpdate as up

up.startingScreen(screen)

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(ws,hs))

window.configure(background='white')

def machineClick(event):

global screen

import InsightsDesignForMachineLearningModels as mac

mac.startingScreen(screen)

def startingScreen(root):

global screen,sample

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="RATING",width=1000,height=1,font=("Helvetica",15,'bold'),fg='#2864ad',bg='#e3efff', borderwidth=2, relief="groove").pack()

lbl\_over = tk.Label(screen,text = "Overview",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_over.bind("<Button-1>",overviewClick)

lbl\_over.place(x=5,y=65)

lbl\_category = tk.Label(screen,text = "Category",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_category.bind("<Button-1>", categoryClick)

lbl\_category.place(x=130,y=65)

lbl\_Installs = tk.Label(screen,text = "Installs",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_Installs.bind("<Button-1>", installClick)

lbl\_Installs.place(x=255,y=65)

lbl\_searchapp = tk.Label(screen,text = "Search App",width=13,height='2',font=("Calibri",13,'bold'),fg='#75acff',bg='white')

lbl\_searchapp.bind("<Button-1>", searchAppClick)

lbl\_searchapp.place(x=255+125,y=65)

lbl\_machine = tk.Label(screen,text = "Learning Models",width=25,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_machine.bind("<Button-1>", machineClick)

lbl\_machine.place(x=255+125+125,y=65)

lbl\_review = tk.Label(screen,text = "Reviews",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_review.bind("<Button-1>", reviewClick)

lbl\_review.place(x=255+125+125+125+108,y=65)

lbl\_lastupdate = tk.Label(screen,text = "Last Updated",width=13,height='2',font=("Calibri",13,'italic'),fg='#75acff',bg='#e1e5eb')

lbl\_lastupdate.bind("<Button-1>", lastupdateClick)

lbl\_lastupdate.place(x=255+125+125+125+125+109,y=65)

big\_frame = tk.Frame(screen,bg='#F8E0E0',width='1520',height='730',bd=4,relief=RIDGE)

big\_frame.place(x=3,y=100)

sample=pd.read\_csv('DATA SET-2.csv')

sample.drop(index=[10472],inplace=True)

sample=sample.replace(np.NaN,0)

year=[2010,2011,2012,2013,2014,2015,2016,2017,2018]

rating=[]

for i in range(5):

for j in range(10):

rating.append(i+(j/10))

rating.append(5.0)

tk.Label(big\_frame,text='Installs',width=10,height=1,font=("Helvetica",15,'bold'),fg='white',bg='#2864ad', borderwidth=2, relief="groove").place(x=150,y=60)

tk.Label(big\_frame,text='Rating',width=10,height=1,font=("Helvetica",15,'bold'),fg='white',bg='#2864ad', borderwidth=2, relief="groove").place(x=450,y=60)

tk.Label(big\_frame,text='Year',width=10,height=1,font=("Helvetica",15,'bold'),fg='white',bg='#2864ad', borderwidth=2, relief="groove").place(x=750,y=60)

combo\_install=ttk.Combobox(big\_frame,width=17,values=['0','10+','100+','1,000+','10,000+','1,00,000+','10,00,000+','1,00,00,000+'],state="readonly")

combo\_install.place(x=150,y=110)

combo\_rating=ttk.Combobox(big\_frame,width=17,values=rating,state="readonly")

combo\_rating.place(x=450,y=110)

combo\_year=ttk.Combobox(big\_frame,width=17,values=year,state="readonly")

combo\_year.place(x=750,y=110)

canvas=tk.Canvas(big\_frame,width=970,height=450,bg='pink')

canvas.place(x=150,y=150)

scroll1=tk.Scrollbar(canvas)

canvas\_listbox=tk.Listbox(canvas,yscrollcommand = scroll1.set,height=20,width=96,bg='#A9D0F5',font=('Calibri',14,'bold'))

canvas\_listbox.pack( side = 'left', fill = 'both' )

scroll1.pack(side='right', fill='y' )

btn\_search=tk.Button(big\_frame,text='Search',bd=12,width=10,bg="powder blue",command=lambda:getting(combo\_install,combo\_rating,combo\_year,canvas\_listbox))

btn\_search.place(x=1020,y=85)

screen.mainloop()

#startingScreen(tk.Tk())

Design to insert data

import pandas as pd

from matplotlib import pyplot as plt

import numpy as np

import sys

from collections import OrderedDict

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from PIL import Image, ImageTk

import InsightsDesign as ove

data=pd.read\_csv('DATA SET-2.csv')

data=data.replace(np.NaN,0)

data.drop(index=[10472],inplace=True)

sample=pd.read\_csv('DATA SET-1.csv')

def saveing(x,y,z,p):

global data

value=[]

if z=='DATA SET-2.csv':

date1=p[0].get()

month=p[1].get()

year=p[2].get()

date=month+' '+date1+','+' '+year

dd=data.columns.tolist()

elif z=='DATA SET-1.csv':

dd=sample.columns.tolist()

for i in x:

value.append(i.get())

if z=='DATA SET-2.csv':

value.insert(10,date)

#print(value)

value[5]=str(value[5])+'+'

value[7]='$'+str(value[7])

#print(value)

#print(dd)

dp=pd.DataFrame([value],columns=dd)

dat=data.append(dp)

elif z=='DATA SET-1.csv':

dp=pd.DataFrame([value],columns=dd)

dat=sample.append(dp)

tk.messagebox.showinfo('Success','Data Successfully Written')

dat.to\_csv(z,index=False)

y.config(state='disabled')

def check1(x):

for i in x:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

try:

if(isinstance(float(x[3].get()), float) and isinstance(float(x[4].get()), float)):

if(float(x[3].get())<=1 and float(x[3].get())>=-1):

if(float(x[4].get())<=1 and float(x[4].get())>=-1):

return False

else:

tk.messagebox.showwarning('Wrong Value','Please provide value in range -1 to 1')

return True

else:

tk.messagebox.showwarning('Wrong Value','Please provide value in range -1 to 1')

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in rating column')

return True

def check(x,z):

d=[]

for i in x:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

for i in z:

if i.get()=='':

tk.messagebox.showwarning('Fields empty','Please provide all the fields')

return True

try:

if(isinstance(float(x[2].get()), float)):# code for checking the user entered a valid rating in the entry field

if(float(x[2].get())<=5 and float(x[2].get())>=0):

d.append(False)

else:

tk.messagebox.showerror('Out of range','Rating should be between 0 to 5 only')

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in rating column')

return True

try:

if(isinstance(int(x[3].get()), int)):

d.append(False)

except:

tk.messagebox.showwarning('Wrong Value','Please provide a integer value in Reviews')

return True

try:

if(isinstance(float(x[4].get()[:-1]), float)):

if(x[4].get()[-1]=='k' or x[4].get()[-1]=='M'):

d.append(False)

else:

tk.messagebox.showerror('Size',"Size should end with 'k' or 'M'")

return True

except:

tk.messagebox.showwarning('Wrong Value','Please provide a integer value followed in size column')

return True

try:

if(isinstance(float(x[5].get()), float)):

d.append(False)

except:

tk.messagebox.showwarning('Wrong Value','Please provide a integer value in Installs')

return True

try:

if(isinstance(float(x[7].get()), float)):

d.append(False)

except:

tk.messagebox.showwarning('Wrong Value','Please provide a float value in Price')

return True

if set(d)==False:

return False

def validate2(x,y):

App=x[0].get()

d=0

ap=sample['App'].unique()

for i in ap:

if i.strip()==App.strip():

msg='App named '+App+' is already present'

tk.messagebox.showerror("Error",msg)

d=1

if(check1(x)):

d=1

if d==0:

y.config(state='normal')

def validate(x,y,z):

App=x[0].get()

d=0

ap=data['App']

for i in ap:

if i.strip()==App.strip():

msg='App named '+App+' is already present'

tk.messagebox.showerror("Error",msg)

d=1

break

if check(x,z):

d=1

if d==0:

y.config(state='normal')

def adjustWindow(window):

global screen

ws=screen.winfo\_screenwidth()

hs=screen.winfo\_screenheight()

window.geometry("%dx%s"%(1300,hs))

window.configure(background='white')

def startingScreen(root):

global screen,df,data

dates=[]

month=['January', 'February', 'March', 'April','May','June','July','August','September', 'October', 'November','December']

years=[]

for i in range(1,32):

dates.append(i)

for i in range(2010,2020):

years.append(i)

root.destroy()

screen = tk.Tk()

adjustWindow(screen)

screen.title("Insights of Google App's")

tk.Label(screen,text="",bg="white").pack()

tk.Label(screen,text="INSERT VALUES",width=1000,height=1,font=("Helvetica",15,'bold'),fg='white',bg='#2864ad', borderwidth=2, relief="groove").pack()

insertition\_frame\_1 = tk.Frame(screen,bg='#d9f5fa',width = 500,height = 640,bd=4,relief=RIDGE)

insertition\_frame\_1.place(x=100,y=80)

header=data.columns.tolist()

category= list(OrderedDict.fromkeys(data['Category']))

content=list(OrderedDict.fromkeys(data['Content Rating']))

genre=list(OrderedDict.fromkeys(data['Genres']))

header2=sample.columns.tolist()

screen.title('Data Modifying')

txt=[]

datecombo=[]

for i in range(1,14):

tk.Label(insertition\_frame\_1,text=header[i-1],width=11,font=("Calibri",11,'italic'),fg='#ab3059',bg='#d9f5fa').place(x=50,y=40\*i)

for i in range(1,14):

if i!=2 and i!=10 and i!=9 and i!=7 and i!=11 and i!=13:

txtfield=tk.Entry(insertition\_frame\_1,bd=10,insertwidth=4,bg="white")

txt.append(txtfield)

txtfield.place(x=150,y=40\*i)

elif i==2:

combo=ttk.Combobox(insertition\_frame\_1,values=category)

txt.append(combo)

combo.place(x=150,y=40\*i)

elif i==9:

combo=ttk.Combobox(insertition\_frame\_1,values=content,state="readonly")

txt.append(combo)

combo.place(x=150,y=40\*i)

elif i==10:

combo=ttk.Combobox(insertition\_frame\_1,values=genre,state="readonly")

txt.append(combo)

combo.place(x=150,y=40\*i)

elif i==7:

combo=ttk.Combobox(insertition\_frame\_1,values=['Free','Paid'],state="readonly")

txt.append(combo)

combo.place(x=150,y=40\*i)

elif i==11:

combo=ttk.Combobox(insertition\_frame\_1,values=dates,width=2,state="readonly")

datecombo.append(combo)

combo.place(x=150,y=40\*i)

combo=ttk.Combobox(insertition\_frame\_1,values=month,width=10,state="readonly")

datecombo.append(combo)

combo.place(x=190,y=40\*i)

combo=ttk.Combobox(insertition\_frame\_1,values=years,width=5,state="readonly")

datecombo.append(combo)

combo.place(x=278,y=40\*i)

elif i==13:

combo=ttk.Combobox(insertition\_frame\_1,values=list(data['Android Ver'].unique()),state="readonly")

txt.append(combo)

combo.place(x=150,y=40\*i)

btn\_save=tk.Button(insertition\_frame\_1,text='Save',state="disabled",bd=12,width=10,bg="powder blue",command=lambda:saveing(txt,btn\_save,'DATA SET-2.csv',datecombo))

btn\_validate=tk.Button(insertition\_frame\_1,text='Validate',bd=12,width=10,bg="powder blue",command=lambda:validate(txt,btn\_save,datecombo))

btn\_validate.place(x=100,y=580)

btn\_save.place(x=250,y=580)

insertition\_frame\_2 = tk.Frame(screen,bg='#d9f5fa',width = 500,height = 640,bd=4,relief=RIDGE)

insertition\_frame\_2.place(x=700,y=80)

txt2=[]

for i in range(1,6):

tk.Label(insertition\_frame\_2,text=header2[i-1],width=17,font=("Calibri",11,'italic'),fg='#ab3059',bg='#d9f5fa').place(x=50,y=40\*i)

for i in range(1,6):

if i!=3:

txtfield=tk.Entry(insertition\_frame\_2,bd=10,insertwidth=4,bg="white")

txt2.append(txtfield)

txtfield.place(x=250,y=40\*i)

elif i==3:

combo=ttk.Combobox(insertition\_frame\_2,values=['Positive','Negative','Neutral'],state="readonly")

txt2.append(combo)

combo.place(x=250,y=40\*i)

btn\_save1=tk.Button(insertition\_frame\_2,text='Save',state="disabled",bd=12,width=10,bg="powder blue",command=lambda:saveing(txt2,btn\_save1,'DATA SET-1.csv',''))

btn\_validate1=tk.Button(insertition\_frame\_2,text='Validate',bd=12,width=10,bg="powder blue",command=lambda:validate2(txt2,btn\_save1))

btn\_validate1.place(x=100,y=580)

btn\_save1.place(x=250,y=580)

tk.Button(screen,text="BACK TO OVERVIEW",bd=10,font=("Calibri",13,'italic'),fg='#75acff',bg='white',command = lambda: ove.startingScreen(screen) ).place(x=550,y=730)

screen.mainloop()

#startingScreen(tk.Tk())