**DEPRESSION SCALER**

CODE FOR DEVELOPING THE BEST FIT MACHINE LEARNING MODEL

**Google Colab File 1: Consists code for data pre- processing and EDA.**

#Importing required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

#Reading the dataset

data=pd.read\_csv("/content/data.csv",sep="\t")

data.head()

#Increasing the visibility of columns

pd.set\_option("display.max\_columns", None)

data.head()

Data pre- processing and EDA

#Dim of dataset

data.shape

Removing the below mentioned features as they are extra information gathered during the survey and are irrelevant to the target variable of the dataset.

**Qx{I} columns(Eg: Q1I)**

**Qx{E} columns (Eg: Q1E)**

**introelapse**

**testelapse**

**surveyelapse**

**engnat**

**hand**

**screensize**

**uniquenetworklocation**

**source**

**VCLx columns (Eg: VCL1)**

#Removing irrelevant features

removeCols=[f'Q{i}E' for i in range(1,43)]

removeCols.extend([f'Q{i}I' for i in range (1,43)])

removeCols.extend([f'VCL{i}' for i in range(1,17)])

removeCols.extend(['introelapse','testelapse','surveyelapse','engnat','hand','screensize','uniquenetworklocation','source'])

data1=data.drop(removeCols, axis=1)

data1.head()

data1.shape

#Handling null values

plt.figure(figsize=(15,5))

sns.heatmap(data1.isnull())

data1.isnull().sum()

#Removing null values

data1['major'].fillna('No Degree', inplace=True)

data1.isnull().sum()

'''Cleaning the 'major' col for different spellings of same major and

answers of type- &#1593 to be classified as 'No Degree'

'''

def rectifyMajor(title):

    if 'busin' in str(title).lower() or 'manage' in str(title).lower() or 'Buss' in str(title) or 'Bisness' in str(title) or 'Manag' in str(title) or 'buis' in str(title) or 'Entrepreneur' in str(title) or 'entrepr' in str(title).lower() or 'managment' in str(title).lower() or 'Buis' in str(title) or 'Busni' in str(title) or 'Mana' in str(title) or 'buss' in str(title).lower() or 'Bi' in str(title) or 'Mgt' in str(title) or 'MBA' in str(title) or 'Mgmt' in str(title) or 'MD' in str(title):

        return 'Business/Management'

    elif 'information technology' in str(title).lower() or 'IT' in str(title) or 'it' in str(title):

        return 'I.T'

    elif 'math' in str(title).lower() or 'LOGISTICS' in str(title) or 'st' in str(title).lower() or 'marh' in str(title).lower() or 'Mate' in str(title):

        return 'Mathematics'

    elif 'computer' in str(title).lower():

        return 'I.T'

    elif 'bio' in str(title).lower() or 'Plant' in str(title) or 'plant' in str(title).lower() or 'Micro' in str(title):

        return 'Biology'

    elif 'tesl' in str(title).lower() or 'TES' in str(title) or 'Teso' in str(title) or 'Enhlish' in str(title):

        return 'English'

    elif 'account' in str(title).lower() or 'Accoun' in str(title) or 'Acc' in str(title) or 'acc' in str(title).lower() or 'Acouunt' in str(title) or 'Acvount' in str(title) or 'Count' in str(title):

        return 'Accountacy'

    elif 'CA' in str(title):

        return 'CA'

    elif 'none' in str(title).lower() or '0' in str(title) or  '\_' in str(title) or '.' in str(title) or 'Nine' in str(title) or '19' in str(title):

        return 'No Degree'

    elif 'nurs' in str(title).lower() or 'BSN' in str(title):

        return 'Nursing'

    elif '-' in str(title).lower() or 'NIL' in str(title):

        return 'No Degree'

    elif 'teach' in str(title).lower() or 'Lect' in str(title) or 'eet' in str(title).lower():

        return 'Teaching'

    elif 'pharma' in str(title).lower() or 'medic' in str(title).lower() or 'med' in str(title).lower() or 'hospi' in str(title).lower() or 'Mwdicine' in str(title) or 'Farmacy' in str(title) or 'Pharacology' in str(title) or 'farmasi' in str(title).lower() or 'Farmasy' in str(title):

        return 'Pharmacy/Medical'

    elif 'doctor' in str(title).lower() or  'MBBS' in str(title) or 'Mbbs' in str(title) or 'Surge' in str(title) or 'surge' in str(title) or 'mbbs' in str(title).lower()or 'dermat' in str(title).lower() or 'Podiat' in str(title) :

        return 'Doctor'

    elif 'no' in str(title).lower() or 'Undec' in str(title) or 'Idk' in str(title) or 'idk' in str(title).lower() or 'Hahaha' in str(title) or 'never' in str(title).lower() or 'T' in str(title) or 'Good' in str(title):

        return 'No Degree'

    elif 'film' in str(title).lower() or 'Cinema' in str(title) or 'fil' in str(title).lower() or 'Adver' in str(title) or 'adver' in str(title) or 'Act' in str(title) or 'Enter' in str(title) or 'digital' in str(title).lower() or 'cinema' in str(title).lower() or 'Video' in str(title) or 'Direct' in str(title) or 'Theat' in str(title) or 'Radio' in str(title) or 'theat' in str(title).lower() or 'drama' in str(title).lower():

        return 'Media'

    elif 'international' in str(title).lower() or 'Internatianal' in str(title):

        return 'International Relations'

    elif 'human' in str(title).lower() or 'hr' in str(title).lower() or 'Hs' in str(title) or 'Hm' in str(title) or 'Humam' in str(title):

        return 'Human Resources'

    elif 'art' in str(title).lower() or 'Painting' in str(title) or 'Drawing' in str(title) or 'ba' in str(title) or 'Printing' in str(title) or 'las' in str(title).lower() or 'Ma' in str(title) or 'paint' in str(title).lower() or 'creative' in str(title).lower() or 'AA' in str(title) or 'BA' in str(title):

        return 'Arts'

    elif 'islam' in str(title).lower() or 'Muamalat' in str(title) or 'Quran' in str(title) or 'Halal' in str(title) or 'Usul' in str(title) or 'Zakat' in str(title) or 'usul' in str(title).lower():

        return 'Islamic Studies'

    elif 'physio' in str(title).lower() or 'fis' in str(title).lower():

        return 'Physiotherapy'

    elif 'socio' in str(title).lower() or 'social' in str(title).lower() or 'soical' in str(title).lower() or 'Sis' in str(title) or 'Sosio' in str(title) or 'Sicio' in str(title) or 'sosiality' in str(title).lower():

        return 'Sociology'

    elif 'bank' in str(title).lower():

        return 'Banking'

    elif 'agri' in str(title).lower():

        return 'Agriculture'

    elif 'Market' in str(title) or 'Finan' in str(title) or 'finance' in str(title).lower() or 'MARKETING' in str(title) or 'market' in str(title).lower() or 'retail' in str(title).lower() or 'CMP' in str(title) or 'Merket' in str(title):

        return 'Marketing/Finance'

    elif 'counsel' in str(title).lower() or 'cauns' in str(title) or 'Kaunseling' in str(title) or 'kaunseling' in str(title) or 'Caunsel' in str(title):

        return 'Counselling'

    elif 'programming' in str(title).lower() or 'coding' in str(title).lower() or 'Ibm' in str(title) or 'ceit' in str(title) or 'Hacking' in str(title):

        return 'I.T'

    elif 'civil' in str(title).lower() or 'comp' in str(title).lower() or 'Mechanical' in str(title) or 'Electrical' in str(title) or 'Mechatronics' in str(title) or 'Eee' in str(title) or 'cs' in str(title).lower() or 'mecha' in str(title) or 'Chemical' in str(title) or 'chemical' in str(title) or 'tech' in str(title) or 'ec' in str(title).lower() or 'egineering' in str(title).lower() or 'manufacturing' in str(title).lower():

        return 'Engineering'

    elif 'ict' in str(title).lower() or 'developer' in str(title).lower() or 'I.T' in str(title) or 'CAE&D' in str(title) or 'It' in str(title):

        return 'I.T'

    elif 'commu' in str(title).lower() or 'comm' in str(title).lower() or 'com' in str(title).lower() or 'Conmunication' in str(title):

        return 'Communications'

    elif 'administration' in str(title).lower() or 'admin' in str(title).lower():

        return 'Administration'

    elif 'psycho' in str(title).lower() or 'psy' in str(title).lower() or 'Clinical osychology' in str(title) or 'hschology' in str(title) or 'Pysch' in str(title) or 'pys' in str(title).lower() or 'Pych' in str(title) or 'pscy' in str(title) or 'payc' in str(title).lower() or 'Phyc' in str(title) or 'psicologia' in str(title) or 'Phsychology' in str(title) or 'Phichology' in str(title) or 'psuchology' in str(title) or 'Pschology' in str(title) or 'psikologi' in str(title).lower():

        return 'Psychology'

    elif 'english' in str(title).lower() or 'Elglish' in str(title) or 'esl' in str(title).lower() or 'Emg' in str(title) or 'emglisj' in str(title).lower():

        return 'English'

    elif 'law' in str(title).lower() or 'BBA' in str(title) or 'llb' in str(title) or 'lew' in str(title).lower() or 'kaw' in str(title).lower() or 'enforcement' in str(title).lower() or 'Kaw' in str(title):

        return 'Law'

    elif 'engineering' in str(title).lower() or 'engi' in str(title).lower() or 'eng' in str(title).lower() or 'Software' in str(title) or 'soft' in str(title).lower() or 'mechanical' in str(title).lower() or 'Egineeering' in  str(title) or 'electronic' in str(title).lower() or 'CE' in str(title) or 'mech' in str(title).lower() or 'Ciclvil' in str(title) or 'Eggineering' in str(title) or 'Tech' in str(title) or 'Teol' in str(title) or 'EEE' in str(title) or 'PE' in str(title):

        return 'Engineering'

    elif 'architecture' in str(title).lower() or 'aechitecture' in str(title).lower() or 'archirecture' in str(title).lower() or 'architect' in str(title).lower() or 'Arsitechture' in str(title) or 'Building' in str(title) or 'building' in str(title).lower() or 'Arc' in str(title):

        return 'Architecture'

    elif 'design' in str(title).lower() or 'Desig' in str(title) or 'Dssign' in str(title):

        return 'Designer'

    elif 'science' in str(title).lower() or 'Sceince' in str(title) or 'Sci' in str(title) or 'sciene' in str(title) or 'BS' in str(title):

        return 'Pure Sciences'

    elif 'physics' in str(title).lower() or 'Phsyics' in str(title) or 'EMC' in str(title) or 'Physic' in str(title) or 'physi' in str(title):

        return 'Physics'

    elif 'chemistry' in str(title).lower() or 'CIS' in str(title) or 'Chem' in str(title):

        return 'Chemistry'

    elif 'french' in str(title).lower() or 'Fr' in str(title):

        return 'French'

    elif 'religi' in str(title).lower() or 'Relegion' in str(title) or 'Rel' in str(title) or 'Hukum' in str(title) or 'Sains' in str(title):

        return 'Religious Studies'

    elif title=='&#1593;&#1604;&#1605; &#1606;&#1601;&#1587;' or title=='&#22810;&#23186;&#39636;&#35373;&#35336;' or title=='nil' or title=='drop out' or title=='&#1055;&#1089;&#1080;&#1093;&#1086;&#1083;&#1' or title=='75' or title=='Secondary education' or title=='Thiê&#769;t kê&#769; &#273;ô&#768; ho&#803;a' or title=='18' or title=='ongoing' or title=='&#28888;&#22521;' or title=='lol' or title=='In college currently' or title=='secondary education' or title=='Dropped out' or title=='na' or title=='didnt attend' or title=='im going on the next year. ' or title=='&#304;lahiyat' or title=='lmfao, im 15' or title=='Elem Ed' or title=='yes' or title=='N/a' or title=='/' or title=='???' or title=='cocaine 101' or title=='doesnt matter' or title== 'oooo' or title=='G' or title=='Yes' or title=='Na' or title=='Na 'or title=='Want sure':

        return 'No Degree'

    elif 'Music' in str(title) or 'Dance' in str(title) or 'danc' in str(title).lower() or 'Vocational' in str(title) or 'Muisc' in str(title) or 'music' in str(title).lower() or 'Performance' in str(title):

        return 'Music/Dance'

    elif 'pol' in str(title).lower() or 'Govern' in str(title) or 'Right' in str(title):

        return 'Politics'

    elif 'photo' in str(title).lower() or 'Foto' in str(title) or 'Photo' in str(title):

        return 'Photography'

    elif 'Television' in str(title) or 'telev' in str(title).lower():

        return 'Television'

    elif 'bahasa' in str(title).lower() or 'Bahasa' in str(title) or 'Malay' in str(title) or 'malay' in str(title).lower():

        return 'Malaysian languages'

    elif 'Urban' in str(title) or 'Town' in str(title) or 'town' in str(title).lower() or 'planning' in str(title) or 'Plann' in str(title) or 'development' in str(title):

        return 'Economic Developments'

    elif 'Public' in str(title) or 'public' in str(title).lower():

        return 'Public Relations'

    elif 'Writing' in str(title) or 'writing' in str(title).lower() or 'Screenwritinf' in str(title) or 'Author' in str(title):

        return 'Writing/Author'

    elif 'philosophy' in str(title).lower() or 'Phil' in str(title) or 'philos' in str(title).lower() or 'Filo' in str(title) or 'Phylosophy' in str(title):

        return 'Philosophy'

    elif 'Actua' in str(title):

        return 'Acturial Studies'

    elif 'DENTALWORKS' in str(title) or 'dental' in str(title) or 'Dental' in str(title) or 'Odont' in str(title):

        return 'Dentist'

    elif 'beaut' in str(title).lower() or 'Fashion' in str(title) or 'make' in str(title) or 'fashion' in str(title).lower() or 'hair' in str(title).lower() or 'cosmet' in str(title).lower():

        return 'Fashion'

    elif 'Health' in str(title) or 'health' in str(title).lower() or 'wellness' in str(title).lower() or 'Healtcare' in str(title):

        return 'Healthcare'

    elif 'Language' in str(title) or 'lang' in str(title).lower() or 'Laq' in str(title):

        return 'Languages'

    elif 'cook' in str(title).lower() or 'bakery' in str(title).lower() or 'Bak' in str(title) or 'CULINARY' in str(title) or 'Food' in str(title) or 'food' in str(title) or 'chef' in str(title).lower() or 'Cul' in str(title) or 'Patiss' in str(title) or 'culi' in str(title).lower():

        return 'Cookings'

    elif 'Hotel' in str(title) or 'hotel' in str(title).lower() or 'food service' in str(title) or 'cater' in str(title).lower():

        return 'Hotel Management'

    elif 'therapy' in str(title).lower() or 'ot' in str(title).lower() or 'theraphy' in str(title):

        return 'Therapeutical Studies'

    elif 'veter' in str(title).lower() or 'Veter' in str(title) or 'Vet' in str(title):

        return 'Veterinary'

    elif 'Survey' in str(title) or 'survey' in str(title) or 'serveyors' in str(title).lower() or 'Qs' in str(title) or 'SURVEYING' in str(title) or 'QS' in str(title) or 'Surver' in str(title):

        return 'Surveyour Studies'

    elif 'Aircraft' in str(title) or 'aircraft' in str(title).lower() or 'aircr' in str(title).lower() or 'aviation' in str(title).lower() or 'Aero' in str(title) or 'navigation' in str(title).lower():

        return 'Aircrafts'

    elif 'environment' in str(title).lower() or 'Environment' in str(title) or 'envi' in str(title).lower():

        return 'Environmental Educations'

    elif 'Syariah' in str(title) or 'syariah' in str(title):

        return 'Syrian Languages'

    elif 'judicial' in str(title).lower() or 'juri' in str(title).lower() or 'legal' in str(title).lower():

        return 'Judicial Studies'

    elif 'Liter' in str(title) or 'literature' in str(title) or 'litt' in str(title).lower():

        return 'Literature'

    elif 'child' in str(title).lower() or 'Child' in str(title) or 'Preschool' in str(title):

        return 'Child Educations'

    elif 'Tour' in str(title) or 'tour'  in str(title).lower():

        return 'Tourisms'

    elif 'Gam' in str(title) or 'game' in str(title).lower():

        return 'Gaming'

    elif 'education' in str(title).lower() or 'Education' in str(title) or 'ed' in str(title).lower() or 'acad' in str(title) or 'Dploma' in str(title):

        return 'B.Ed or M.Ed'

    elif 'Sport' in str(title) or 'sport' in str(title).lower():

        return 'Sports'

    elif 'Petro' in str(title):

        return 'Petroleum'

    elif 'Journ' in str(title) or 'jour' in str(title).lower() or 'Joun' in str(title) or 'Jurn' in str(title):

        return 'Journalism'

    elif 'Mandarin' in str(title):

        return 'Chinese/Mandarin Languages'

    elif 'Electrician' in str(title):

        return 'Electrician'

    elif 'Network' in str(title) or 'network' in str(title).lower():

        return 'Networking'

    elif 'geo' in str(title).lower() or 'GEO' in str(title):

        return 'Geography'

    elif 'Librarian' in str(title) or 'lib' in str(title).lower():

        return 'Librarian'

    elif 'Mission' in str(title) or 'mission' in str(title).lower():

        return 'Missionary Studies'

    elif 'Forensic' in str(title) or 'foren' in str(title).lower() or 'Crime' in str(title) or 'crim' in str(title).lower():

        return 'Forensic/Criminal studies'

    elif 'Animation' in str(title) or 'animation' in str(title).lower() or 'imag' in str(title) or 'graphic' in str(title) or 'Graphic' in str(title):

        return 'Animations'

    elif 'aqua' in str(title).lower() or 'Aqu' in str(title):

        return 'Aquaculture'

    elif 'soldier' in str(title).lower() or 'lwa' in str(title).lower() or 'defence' in str(title):

        return 'Army'

    elif 'Kinesi' in str(title) or 'kines' in str(title).lower() or 'hod' in str(title):

        return 'Human Kinetics'

    elif 'Horti' in str(title) or 'horti' in str(title) or 'Landscape' in str(title):

        return 'Horticulture'

    elif 'commerce' in str(title).lower() or 'Coome' in str(title):

        return 'Commerce'

    elif 'Speech' in str(title) or 'speech' in str(title).lower():

        return 'Speech Pathology'

    elif 'SECRET' in str(title) or 'secret' in str(title).lower():

        return 'Secretary'

    elif 'Animals' in str(title) or 'animal' in str(title).lower() or 'Pet' in str(title):

        return 'Animal Care'

    elif 'Organisation' in str(title) or 'organi' in str(title).lower():

        return 'Organizational Behaviour'

    elif 'event' in str(title).lower() or 'Event' in str(title):

        return 'Event Managment'

    elif 'radiology' in str(title).lower() or 'Radiography' in str(title) or 'radiograpghy' in str(title).lower() or 'Radiation' in str(title) or 'radiography' in str(title):

        return 'Radiography'

    elif 'nutrition' in str(title).lower() or 'Nutrition' in str(title):

        return 'Nutritionist'

    elif 'Audit' in str(title) or 'audit' in str(title).lower():

        return 'Auditing'

    elif 'Neuro' in str(title) or 'neuroligy' in str(title).lower():

        return 'Neurology'

    elif 'Anato' in str(title) or 'anat' in str(title).lower():

        return 'Anatomy'

    elif 'trade' in str(title).lower():

        return 'Trading'

    elif 'Interpre' in str(title) or 'translation' in str(title).lower():

        return 'Interpreter'

    elif 'audio' in str(title).lower() or 'Audio' in str(title):

        return 'Audiology'

    elif 'insurance' in str(title).lower() or 'Insurance' in str(title):

        return 'Insurances'

    elif 'archaeology' in str(title).lower() or 'archaeology' in str(title).lower() or 'archeology' in str(title).lower() or 'treasury' in str(title):

        return 'Archeology'

    elif 'SERV'in str(title) or 'service' in str(title).lower():

        return 'Service Training'

    elif 'GERMAN' in str(title) or 'german' in str(title).lower():

        return 'German'

    elif 'KOREAN' in str(title) or 'Korea' in str(title):

        return 'Korean'

    elif 'valuat' in str(title).lower() or 'valuer' in str(title).lower():

        return 'Registered Valuer'

    elif 'skil' in str(title).lower() or 'Skill' in str(title) or 'Professional' in str(title) or 'practical' in str(title).lower():

        return 'Skilled Labour'

    elif 'virology' in str(title):

        return 'Virology'

    elif 'lab' in str(title).lower() or 'Lab' in str(title) or 'MLT' in str(title):

        return 'Laboratory Worker'

    elif 'GENERAL' in str(title) or 'General' in str(title):

        return 'General'

    elif 'Opto' in str(title) or 'opto' in str(title).lower():

        return 'Optometry'

    elif 'Zoo' in str(title) or 'zoo' in str(title).lower():

        return 'Zoology'

    elif 'office' in str(title).lower() or 'Office' in str(title):

        return 'Office Skills'

    elif 'found' in str(title).lower() or 'Found' in str(title):

        return 'Foundation Education'

    elif 'general' in str(title).lower() or 'General' in str(title):

        return 'General Education'

    elif 'real estate' in str(title).lower() or 'property' in str(title).lower():

        return 'Realtor'

    elif 'Meteorology' in str(title) or 'Metrology' in str(title):

        return 'Meterology'

    elif 'operations' in str(title).lower() or 'Operation' in str(title):

        return 'Operational Managment'

    elif 'Merchandising' in str(title) or 'merchand' in str(title).lower():

        return 'Merchandising'

    elif 'Spanish' in str(title):

        return 'Spanish'

    elif 'Nature' in str(title) or 'natur' in str(title).lower():

        return 'Nature Conservation/Resources'

    elif title=='a level ' or title==' ':

        return 'No Degree'

    elif 'Corporate' in str(title) or 'corporate' in str(title).lower():

        return 'Corporate'

    elif 'greek' in str(title).lower() or 'Greek' in str(title):

        return 'Greek'

    elif 'Behaviour' in str(title) or 'Behavior' in str(title) or 'Organizational Behaviour' in str(title):

        return 'Behaviour Analysis'

    elif 'publish' in str(title).lower():

        return 'Publishing'

    elif 'Safety' in str(title) or 'safety' in str(title).lower():

        return 'Safety Training'

    elif 'genetic' in str(title).lower() or 'Genetic' in str(title):

        return 'Genetics'

    elif 'Dietetic' in str(title):

        return 'Dietician'

    elif 'Production' in str(title) or 'manufacturing' in str(title).lower():

        return 'Production And Manufacturing'

    elif 'Welding' in str(title):

        return 'Welding'

    elif 'Geron' in str(title):

        return 'Gerontology'

    elif 'Research' in str(title) or 'Ph D' in str(title):

        return 'Ph.D'

    elif 'arabic' in str(title).lower() or 'Arabic' in str(title):

        return 'Arabic'

    else:

        return title

data1['major'] = data1['major'].apply(rectifyMajor)

data1=data1.replace([np.inf, -np.inf], np.nan)

data1=data1.dropna()

Investigating the categorical variables.

1. What is the 'major' of people who participated in the survey?

data1.head()

data1['major'].value\_counts()

plt.figure(figsize=(10,5))

data1['major'].value\_counts()[:20].plot(kind='barh',color='blue')

plt.ylabel('Majors')

plt.xlabel('Count')

plt.title('Top 20 majors of people participated in the survey')

data1.drop('major', inplace=True, axis=1)

data1.head()

1. Nationalities of people who participated in the survey.

plt.figure(figsize=(10,5))

data1['country'].value\_counts()[:20].plot(kind='barh',color='green')

plt.ylabel('Country Initials')

plt.xlabel('Count')

plt.title('Top 20 countries from where the people participated in the survey')

data1.drop('country', inplace=True, axis=1)

data1.head()

**Calculating the depression score and extracting the depression dataset:**

The dataset used consists of survey answer for DAS- 42 questions => it consists of survey results for depression, anxiety, and stress scaler. We are working on a depression scaler and hence, have to extract questions only for depression.

**Questions class for each thing in DAS-**

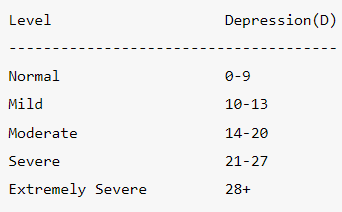
Each of the 42 questions is scored on a 4-point scale ranging from 0 (“Did not apply to me at all”) to 3 (“Applied to me very much, or most of the time”). Scores for Depression, Anxiety, and Stress are calculated by summing the scores for the relevant items:

**Depression: 3, 5, 10, 13, 16, 17, 21, 24, 26, 31, 34, 37, 38, 42**

Anxiety: 2, 4, 7, 9, 15, 19, 20, 23, 25, 28, 30, 36, 40, 41

Stress: 1, 6, 8, 11, 12, 14, 18, 22, 27, 29, 32, 33, 35, 39

**Scoring-** The severity labels are used to describe the full range of scores in the population, so ‘mild’ for example means that the person is above the population mean but probably still below the typical severity of someone seeking help (i.e. it does not mean a mild level of disorder).



#Separating DAS-42 question columns and the rest

data2=data1.filter(regex='Q\d{1,2}A')

data2.head()

other\_data=data1.iloc[:,42:]

other\_data=other\_data.astype(int)

other\_data.head()

#Subtracting one from the responses to the DAS-42 questions as scoring for each question ranges from 0-3 but the answers are from 1-4

def subOne(data):

    return data2.subtract(1,axis=1)

data2=subOne(data2)

data2.head()

depressionKey={'Depression':[3, 5, 10, 13, 16, 17, 21, 24, 26, 31, 34, 37, 38, 42]}

dep=[]

for i in depressionKey["Depression"]:

    dep.append('Q'+str(i)+'A')

depression\_data= data2.filter(dep)

depression\_data.head()

#Calculation depression score

def score(data):

    col=list(data)

    data['totalScore']=data[col].sum(axis=1)

    return data

depression\_data=score(depression\_data)

depression\_data.head()

#merging the datasets and producing the depression dataset

depression=pd.merge(depression\_data,other\_data,how='left',left\_index=True,right\_index=True)

depression.head()

def level(x):

    if x<=9:

        return 'Normal'

    if  10<=x<=13:

        return 'Mild'

    if 14<=x<=20:

        return 'Moderate'

    if 21<=x<=27:

        return 'Severe'

    if x>28:

        return 'Extremely Severe'

depression['Level']=depression['totalScore'].apply(level)

depression.head()

depression.drop('totalScore',inplace=True,axis=1)

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

ax = sns.countplot(x='Level', data=depression,order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

for p in ax.patches:

        ax.annotate('{:.0f}'.format(p.get\_height()), (p.get\_x()+0.25, p.get\_height()+150))

plt.title('Plot of people across the depression levels')

plt.show()

#Age feature

depression['age'].value\_counts().sort\_index()

plt.figure(figsize=(12,5))

ax=sns.boxenplot(x=depression['age'])

print('Depression size before dropping outliers in "age":',depression.shape[0])

indexes=depression[depression['age']>75].index

depression = depression.drop(indexes, axis=0)

print('Depression size after dropping outliers in "age":',depression.shape[0])

plt.figure(figsize=(12,5))

ax=sns.boxenplot(x=depression['age'])

#Grouping age values into different age groups and dropping the age feature.

def ageGroups(x):

    if x<=10:

        return 'Under 10'

    if  10<=x<=16:

        return 'Primary Children'

    if 17<=x<=21:

        return 'Secondary Children'

    if 21<=x<=35:

        return 'Adults'

    if 36<=x<=48:

        return 'Elder Adults'

    if x>=49:

        return 'Older People'

age=depression['age'].apply(ageGroups)

plt.figure(figsize=(15, 5))

sns.countplot(x=depression['age'], hue=age)

depression['ageGroups']=depression['age'].apply(ageGroups)

depression.drop('age',axis=1,inplace=True)

depression.head()

plt.figure(figsize=(12,6))

sns.countplot(depression.ageGroups,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

Converting categorical variable- ageGroups to labelled class.

**1-** Under 10

**2-** Primary Children

**3-** Secondary Children

**4-** Adults

**5-** Elder Adults

**6-** Older People

def ageClass(value):

    if value =='Under 10':

        return 1

    if  value=='Primary Children':

        return 2

    if value=='Secondary Children':

        return 3

    if value=='Adults':

        return 4

    if value=='Elder Adults':

        return 5

    if value =='Older People':

        return 6

depression['ageGroups']=depression['ageGroups'].apply(ageClass)

depression.head()

#Family size feature

depression['familysize'].value\_counts().sort\_index()

plt.figure(figsize=(12,5))

ax=sns.boxenplot(x=depression['familysize'])

print('Depression size before dropping outliers in "familysize":',depression.shape[0])

indexes=depression[depression['familysize']>13].index

depression = depression.drop(indexes, axis=0)

print('Depression size after dropping outliers in "age":',depression.shape[0])

plt.figure(figsize=(12,5))

ax=sns.boxenplot(x=depression['familysize'])

plt.figure(figsize=(12, 6))

sns.countplot(x=depression['familysize'], hue=depression['familysize'])

plt.figure(figsize=(15,6))

sns.countplot(depression.familysize,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

Comparing different levels of depression scale with other features like gender, marriage, education, races etc.

**Education level:**

**1**- Less than high school

**2**- High school

**3**- University degree

**4**- Graduate degree

depression['education'].value\_counts().sort\_index()

#Since there are 0's as answers in this question, we combine 0 and 1 as 'Less than high school'

depression['education'] = depression['education'].map({ 0: 1,  1: 1, 2: 2, 3: 3, 4: 4 })

def label(title) -> str:

  if title == 0 or title == 1:

        return 'Less than high school'

  if title == 2:

        return 'High school'

  if title == 3:

        return 'University degree'

  if title == 4:

        return 'Graduate degree'

  return title

educationLevel = depression['education'].apply(label)

plt.figure(figsize=(10,5))

sns.countplot(x=depression['education'],hue=educationLevel)

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

sns.countplot(depression.education,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Urban feature values -**

**1-** Rural (country side)

**2-** Suburban

**3-** Urban (town, city)

depression['urban'].value\_counts()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume them to be from urban area as most of the records are from urban area.'''

depression['urban']=depression['urban'].map({ 0: 3,  1: 1, 2: 2, 3: 3, 4: 4 })

def label(title) -> str:

  if title == 0 or title == 3:

        return 'Urban'

  if title == 1:

        return 'Rural'

  if title == 2:

        return 'Suburban'

  return title

areaType = depression['urban'].apply(label)

plt.figure(figsize=(10,5))

sns.countplot(x=depression['urban'],hue=areaType)

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

sns.countplot(depression.urban,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Gender feature values -**

**1-** Male

**2-** Female

**3-** Other

depression['gender'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume them to be female as most of the records show female participation in the survey.'''

depression['gender']=depression['gender'].map({0: 2, 1: 1, 2: 2, 3: 3})

def label(title) -> str:

  if title == 0 or title == 2:

        return 'Female'

  if title == 1:

        return 'Male'

  if title == 3:

        return 'Other'

  return title

genderType = depression['gender'].apply(label)

plt.figure(figsize=(10,5))

sns.countplot(x=depression['gender'],hue=genderType)

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

sns.countplot(depression.gender,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Religion feature values -**

**1-** Agnostic

**2-** Atheist

**3-** Buddhist

**4-** Christian (Catholic)

**5-** Christian (Mormon)

**6-** Christian (Protestant)

**7-** Christian (Other)

**8-** Hindu

**9-** Jewish

**10-** Muslim

**11-** Sikh

**12-** Other

depression['religion'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume them to be Muslim as most of the records show Muslim participation in the survey.'''

depression['religion']=depression['religion'].map({0: 10, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 8: 8, 9: 9, 10: 10, 11: 11, 12: 12})

def label(title) -> str:

  if title == 0 or title == 10:

        return 'Muslim'

  if title == 1:

        return 'Agnostic'

  if title == 2:

        return 'Atheist'

  if title == 3:

        return 'Buddhist'

  if title == 4:

        return 'Christian (Catholic)'

  if title == 5:

        return 'Christian (Mormon)'

  if title == 6:

        return 'Christian (Protestant)'

  if title == 7:

        return 'Christian (Other)'

  if title == 8:

        return 'Hindu'

  if title == 9:

        return 'Jewish'

  if title == 11:

        return 'Sikh'

  if title == 12:

        return 'Other'

  return title

religionType = depression['religion'].apply(label)

plt.figure(figsize=(13,6))

sns.countplot(x=depression['religion'],hue=religionType)

plt.figure(figsize=(15,6))

sns.countplot(depression.religion,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Sexual Orientation feature values -**

**1-** Heterosexual

**2-** Bisexual

**3-** Homosexual

**4-** Asexual

**5-** Other

depression['orientation'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume them to be Heterosexual as most of the records show Heterosexual orientation people participation in the survey.'''

depression['orientation']=depression['orientation'].map({0: 1, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5})

def label(title) -> str:

  if title == 0 or title == 1:

        return 'Heterosexual'

  if title == 2:

        return 'Bisexual'

  if title == 3:

        return 'Homosexual'

  if title == 4:

        return 'Asexual'

  if title == 5:

        return 'Other'

  return title

orientationType = depression['orientation'].apply(label)

plt.figure(figsize=(10,5))

sns.countplot(x=depression['orientation'],hue=orientationType)

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

sns.countplot(depression.orientation,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Race feature value -**

**10-** Asian

**20-** Arab

**30-** Black

**40-** Indigenous Australian

**50-** Native American

**60-** White

**70-** Other

depression['race'].value\_counts().sort\_index()

depression['race'] = depression['race'].apply(lambda x: x/10)

def label(title) -> str:

  if title == 1:

        return 'Asian'

  if title == 2:

        return 'Arab'

  if title == 3:

        return 'Black'

  if title == 4:

        return 'Indigenous Australian'

  if title == 5:

        return 'Native American'

  if title == 6:

        return 'White'

  if title == 7:

        return 'Other'

  return title

depression['race']=depression['race'].astype(int)

raceType = depression['race'].apply(label)

plt.figure(figsize=(12,6))

sns.countplot(x=depression['race'],hue=raceType)

plt.figure(figsize=(12,6))

sns.countplot(depression.race,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Voted feature values -**

**1-** Yes

**2-** No

depression['voted'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume that they did not vote as most of the records show people in the survey have not voted.'''

depression['voted']=depression['voted'].map({0: 2, 1: 1, 2: 2})

def label(title) -> str:

  if title == 1:

        return 'Yes'

  if title == 2 or title == 0:

        return 'No'

  return title

ifVoted = depression['voted'].apply(label)

plt.figure(figsize=(8,5))

sns.countplot(x=depression['voted'],hue=ifVoted)

plt.figure(figsize=(8,5))

sns.countplot(depression.voted,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Married feature values -**

**1-** Never married

**2-** Currently married

**3-** Previously married

depression['married'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume that they are not married as most of the records of survey show unmarried peopple participation.'''

depression['married']=depression['married'].map({0: 1, 1: 1, 2: 2, 3: 3})

def label(title) -> str:

  if title == 1 or title == 0:

        return 'Never Married'

  if title == 2:

        return 'Currently Married'

  if title == 3:

        return 'Previously Married'

  return title

maritalStatus= depression['married'].apply(label)

plt.figure(figsize=(8,5))

sns.countplot(x=depression['married'],hue=maritalStatus)

plt.figure(figsize=(10,5))

sns.countplot(depression.married,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

**Exploring how the 'Ten Item Personality Inventory' and depression scale vary with each item.**

The values each item in TIPI -

**1-** Disagree strongly

**2-** Disagree moderately

**3-** Disagree a little

**4-** Neither agree nor disagree

**5-** Agree a little

**6-** Agree moderately

**7-** Agree strongly

1. **TIPI1:** Extraverted, enthusiastic -

depression['TIPI1'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 1.'''

depression['TIPI1']=depression['TIPI1'].map({0: 1, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

def label(title) -> str:

  if title == 0 or title ==1:

    return 'Disagree strongly'

  if title == 2:

     return 'Disagree moderately'

  if title == 3:

    return 'Disagree a little'

  if title == 4:

    return 'Neither disagree nor agree'

  if title == 5:

    return 'Agree a little'

  if title == 6:

    return 'Agree moderately'

  if title == 7:

    return 'Agree strongly'

  return title

tipi1=depression['TIPI1'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI1,hue=tipi1)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI1,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI2:** Critical, quarrelsome –

depression['TIPI2'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 5.'''

depression['TIPI2']=depression['TIPI2'].map({0: 5, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi2=depression['TIPI2'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI2,hue=tipi2)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI2,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI3:** Dependable, self-disciplined –

depression['TIPI3'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 6.'''

depression['TIPI3']=depression['TIPI3'].map({0: 6, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi3=depression['TIPI3'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI3,hue=tipi3)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI3,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI4:** Anxious, easily upset –

depression['TIPI4'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 7.'''

depression['TIPI4']=depression['TIPI4'].map({0: 7, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi4=depression['TIPI4'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI4,hue=tipi4)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI4,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI5:** Open to new experiences, complex –

depression['TIPI5'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 6.'''

depression['TIPI5']=depression['TIPI5'].map({0: 6, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi5=depression['TIPI5'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI5,hue=tipi5)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI5,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI6:** Reserved, quiet –

depression['TIPI6'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 7.'''

depression['TIPI6']=depression['TIPI6'].map({0: 7, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi6=depression['TIPI6'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI6,hue=tipi6

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI6,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI7:** Sympathetic, warm –

depression['TIPI7'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 6.'''

depression['TIPI7']=depression['TIPI7'].map({0: 6, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi7=depression['TIPI7'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI7,hue=tipi7)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI7,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI8:** Disorganized, careless –

depression['TIPI8'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 6.'''

depression['TIPI8']=depression['TIPI8'].map({0: 6, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi8=depression['TIPI8'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI8,hue=tipi8)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI8,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI9:** Calm, emotionally stable –

depression['TIPI9'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 4.'''

depression['TIPI9']=depression['TIPI9'].map({0: 4, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi9=depression['TIPI9'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI9,hue=tipi9)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI9,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

1. **TIPI10:** Conventional, uncreative –

depression['TIPI10'].value\_counts().sort\_index()

'''Since there are 0's as answers in this question implying the user has not entered the data,

we assume the maximum response as the answer and combine 0 and 4.'''

depression['TIPI10']=depression['TIPI10'].map({0: 4, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7})

tipi10=depression['TIPI10'].apply(label)

plt.figure(figsize=(12,5))

sns.countplot(x=depression.TIPI10,hue=tipi10)

plt.figure(figsize=(14,6))

sns.countplot(depression.TIPI10,hue=depression['Level'],palette="blend:#CC0000,#EDA",hue\_order=['Extremely Severe','Severe','Moderate','Mild','Normal'])

last\_col=depression.pop('Level')

depression.insert(34,'Level',last\_col)

#Saving the cleaned and modified dataset as a .csv file

from google.colab import files

depression.to\_csv('depression.csv')

files.download('depression.csv')

**Google Colab File 2- Contains code to address imbalance of the dataset and machine learning implementation**.

#Importing required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

depression=pd.read\_csv('/content/depression.csv',header=None)

depression.head()

depression.shape

depression=depression.replace([np.inf, -np.inf], np.nan)

depression=depression.dropna()

data=depression.values

from imblearn.over\_sampling import SMOTE

from sklearn.preprocessing import LabelEncoder

from collections import Counter

#Splitting the dataset into input features and target variable

X, y = data[:, :-1], data[:, -1]

#Label encoding the target variable

Ly=LabelEncoder()

y = Ly.fit\_transform(y)

Ly.classes\_

from sklearn.model\_selection import train\_test\_split,cross\_val\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, stratify=y,random\_state = 42)

#Check the shape of X\_train and X\_test

X\_train.shape, X\_test.shape

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

X\_train = pd.DataFrame(X\_train)

X\_test = pd.DataFrame(X\_test)

Before applying SMOTE –

#Summarizing the distribution

counter = Counter(y\_train)

for keys,values in counter.items():

  percent = values/ len(y) \* 100

  print('Class=%d, n=%d (%.3f%%)' % (keys, values, percent))

#Plotting the distribution

plt.bar(counter.keys(), counter.values())

Applying SMOTE on training dataset-

#Transforming the training dataset

oversample = SMOTE(random\_state=42)

X\_train, y\_train = oversample.fit\_resample(X\_train, y\_train)

#Summarizing the distribution

counter = Counter(y\_train)

for keys,values in counter.items():

  percent = values/ len(y) \* 100

  print('Class=%d, n=%d (%.3f%%)' % (keys, values, percent))

#Plotting the distribution

plt.bar(counter.keys(), counter.values())

Model Creation

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report

Random Forest

#Random Forest

rfc = RandomForestClassifier(n\_estimators=150)

rfc.fit(X\_train, y\_train)

rfc\_pred = rfc.predict(X\_test)

print(classification\_report(y\_test,rfc\_pred))

print('Cross Validation Score:',round(np.mean(cross\_val\_score(rfc, X\_train, y\_train, cv = 10)),3))

x=rfc\_pred

y=y\_test

sns.distplot(x, hist = False, color = 'r', label = 'Predicted Values')

sns.distplot(y, hist = False, color = 'g', label = 'Actual Values')

plt.title('Random Forest')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.legend()

K- Nearest Neighbours

from sklearn import metrics as mt

#Finding optimum k value

from sklearn.neighbors import KNeighborsClassifier

optK=[]

for i in range(1,112):

  knn=KNeighborsClassifier(n\_neighbors=i)

  knn.fit(X\_train,y\_train)

  y\_pred=knn.predict(X\_test)

  optK.append(mt.accuracy\_score(y\_test,y\_pred))

maxval=max(optK)

kval=optK.index(maxval)+1

print(kval) #adding one as the indexing start from zero.

#kNN Classifier

knn=KNeighborsClassifier(n\_neighbors=kval)

knn.fit(X\_train, y\_train)

knn\_pred=knn.predict(X\_test)

print(classification\_report(y\_test,knn\_pred))

print('Cross Validation Score:',round(np.mean(cross\_val\_score(knn, X\_train, y\_train, cv = 10)),3))

x=knn\_pred

y=y\_test

sns.distplot(x, hist = False, color = 'r', label = 'Predicted Values')

sns.distplot(y, hist = False, color = 'g', label = 'Actual Values')

plt.title('kNN Classifier')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.legend()

Naive Bayes

#Naive Bayes

from sklearn.naive\_bayes import MultinomialNB

mnb = MultinomialNB()

mnb.fit(X\_train, y\_train)

nb\_pred = mnb.predict(X\_test)

print(classification\_report(y\_test,nb\_pred))

print('Cross Validation Score:',round(np.mean(cross\_val\_score(mnb, X\_train, y\_train, cv = 10)),3))

x=nb\_pred

y=y\_test

sns.distplot(x, hist = False, color = 'r', label = 'Predicted Values')

sns.distplot(y, hist = False, color = 'g', label = 'Actual Values')

plt.title('Naive Bayes')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.legend()

eXtreme Gradient Boosting

#XGBoost

from xgboost import XGBClassifier

xgb=XGBClassifier()

xgb.fit(X\_train, y\_train)

xgb\_pred = xgb.predict(X\_test)

print(classification\_report(y\_test,xgb\_pred))

print('Cross Validation Score:',round(np.mean(cross\_val\_score(xgb, X\_train, y\_train, cv = 10)),3))

x=xgb\_pred

y=y\_test

sns.distplot(x, hist = False, color = 'r', label = 'Predicted Values')

sns.distplot(y, hist = False, color = 'g', label = 'Actual Values')

plt.title('eXtreme Gradient Boosting')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.legend()

Saving the model with maximum efficiency and generating random test cases for prediction:

def testcases():

    # get random answers for the Depression 14 questions

    qaAnswers = []

    for i in range(1, 15):

        qaAnswers.append(np.random.randint(0, 4)) # random values from 0 to 3

    # get random answers for TIPI1 to TIPI10 questions

    tipiAnswers = []

    for i in range(1, 11):

        tipiAnswers.append(np.random.randint(1, 8)) # random values from 1 to 7

    education = np.random.randint(1, 5) # random values from 1 to 4

    urban = np.random.randint(1, 4) # random values from 1 to 3

    gender = np.random.randint(1, 4) # random values from 1 to 3

    religion = np.random.randint(1, 13) # random values from 1 to 12

    orientation=np.random.randint(1, 6) # random values from 1 to 5

    race = np.random.randint(1, 8) # random values from 1 to 7

    voted = np.random.randint(1, 3) # random values from 1 to 2

    married = np.random.randint(1, 4) # random values from 1 to 3

    familysize = np.random.randint(1, 21) # random values from 1 to 20

    age\_group = np.random.randint(1, 7) # random values from 1 to 6

    return np.array([\*qaAnswers, \*tipiAnswers, education, urban, gender, religion, orientation, race, voted, married, familysize, age\_group])

for i in range(1, 10):

    values = testcases()

    values\_scaled = scaler.transform([values])

    print(f'\*\*\*\*\*\*\*\*\*\*\* Iteration {i} \*\*\*\*\*\*\*\*\*\*\*\*\*')

    print(f'Test Values: {values}')

    print(f'Prediction ===> {Ly.inverse\_transform([rfc.predict(values\_scaled)])}')

    print(f'-------------------------------------------------------------\n')

#saving the trained model

import pickle

data={"model": rfc,"labels":Ly}

with open('saved\_model.pkl','wb') as file:

  pickle.dump(data,file)