Code:

Importing libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px

import warnings

warnings.filterwarnings('ignore')

**READING DATASETS**

df1 = pd.read\_csv('mental-and-substance-use-as-share-of-disease.csv')

df2=pd.read\_csv("prevalence-by-mental-and-substance-use-disorder.csv")

**SHOW DATA SET**

df1.head()

df2.head()

**MERGING TWO DATASETS**

data = pd.merge(df1, df2)

data.head()

**DATA CLEANING**

data.isnull().sum()

data.drop('Code', axis=1, inplace=True)

data.size,data.shape

**RENAMED COLUMNS**

data.set\_axis(['Country','Year','Schizophrenia', 'Bipolar\_disorder', 'Eating\_disorder','Anxiety','drug\_usage','depression','alcohol','mental\_fitness'], axis='columns', inplace=True)

data.head()

**EXPLORATORY ANALYSIS**

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

sns.heatmap(data.corr(),annot=True,cmap='Greens')

plt.plot()

sns.jointplot(data,x="Schizophrenia",y="mental\_fitness",kind="reg",color="m")

plt.show()

sns.jointplot(data,x='Bipolar\_disorder',y='mental\_fitness',kind='reg',color='blue')

plt.show()

sns.pairplot(data,corner=True)

plt.show()

mean = data['mental\_fitness'].mean()

mean

fig = px.pie(data, values='mental\_fitness', names='Year')

fig.show()

fig=px.bar(data.head(10),x='Year',y='mental\_fitness',color='Year',template='ggplot2')

fig.show()

**YEARWISE VARIATIONS IN MENTAL FITNESS OF DIFFERENT COUNTRIES**

fig = px.line(data, x="Year", y="mental\_fitness", color='Country',markers=True,color\_discrete\_sequence=['red','blue'],template='plotly\_dark')

fig.show()

df=data.copy()

df.head()

df.info()

from sklearn.preprocessing import LabelEncoder

l=LabelEncoder()

for i in df.columns:

if df[i].dtype == 'object':

df[i]=l.fit\_transform(df[i])

X = df.drop('mental\_fitness',axis=1)

y = df['mental\_fitness']

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

xtrain, xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.2, random\_state=2)

X = df.drop('mental\_fitness',axis=1)

y = df['mental\_fitness']

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

xtrain, xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.2, random\_state=2)

print("xtrain: ", xtrain.shape)

print("xtest: ", xtest.shape)

print("ytrain: ", ytrain.shape)

print("ytest: ", ytest.shape)

**LINEAR REGRESSION**

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

lr = LinearRegression()

lr.fit(xtrain,ytrain)

# model evaluation for training set

ytrain\_pred = lr.predict(xtrain)

mse = mean\_squared\_error(ytrain, ytrain\_pred)

rmse = (np.sqrt(mean\_squared\_error(ytrain, ytrain\_pred)))

r2 = r2\_score(ytrain, ytrain\_pred)

print("The model performance for training set")

print("--------------------------------------")

print('MSE is {}'.format(mse))

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# model evaluation for testing set

ytest\_pred = lr.predict(xtest)

mse = mean\_squared\_error(ytest, ytest\_pred)

rmse = (np.sqrt(mean\_squared\_error(ytest, ytest\_pred)))

r2 = r2\_score(ytest, ytest\_pred)

print("The model performance for testing set")

print("--------------------------------------")

print('MSE is {}'.format(mse))

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

**RANDOM FOREST REGRESSOR**

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor()

rf.fit(xtrain, ytrain)

# model evaluation for training set

ytrain\_pred = rf.predict(xtrain)

mse = mean\_squared\_error(ytrain, ytrain\_pred)

rmse = (np.sqrt(mean\_squared\_error(ytrain, ytrain\_pred)))

r2 = r2\_score(ytrain, ytrain\_pred)

print("The model performance for training set")

print("--------------------------------------")

print('MSE is {}'.format(mse))

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# model evaluation for testing set

ytest\_pred = rf.predict(xtest)

mse = mean\_squared\_error(ytest, ytest\_pred)

rmse = (np.sqrt(mean\_squared\_error(ytest, ytest\_pred)))

r2 = r2\_score(ytest, ytest\_pred)

print("The model performance for testing set")

print("--------------------------------------")

print('MSE is {}'.format(mse))

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

**KNEIGHBORS REGRESSION**

from sklearn.neighbors import KNeighborsRegressor

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

knn=KNeighborsRegressor(n\_neighbors=10)

knn.fit(xtrain,ytrain)

ytrain\_pred = knn.predict(xtrain)

mse = mean\_squared\_error(ytrain, ytrain\_pred)

rmse = (np.sqrt(mean\_squared\_error(ytrain, ytrain\_pred)))

r2 = r2\_score(ytrain, ytrain\_pred)

print("The model performance for training set")

print("--------------------------------------")

print('MSE is {}'.format(mse))

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

# model evaluation for testing set

ytest\_pred = knn.predict(xtest)

mse = mean\_squared\_error(ytest, ytest\_pred)

rmse = (np.sqrt(mean\_squared\_error(ytest, ytest\_pred)))

r2 = r2\_score(ytest, ytest\_pred)

print("The model performance for testing set")

print("--------------------------------------")

print('MSE is {}'.format(mse))

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))