Open In Colab

IMPORTING REQUIRED LIBRARIES

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force
```

```
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("/content/drive/MyDrive/mental-and-substance-use-as-share-of-disease.csv")
df2=pd.read_csv("/content/drive/MyDrive/prevalence-by-mental-and-substance-use-disorder.csv")
```

```
df1.head()
```

	Entity	Code	Year	DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)
0	Afghanistan	AFG	1990	1.696670
1	Afghanistan	AFG	1991	1.734281
2	Afghanistan	AFG	1992	1.791189
3	Afghanistan	AFG	1993	1.776779

```
df2.head()
```

	Entity	Code	Year	Prevalence - Schizophrenia - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Bipolar disorder - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Eating disorders - Sex: Both - Age: Age-standardized (Percent)	Pre di Se sta
0	Afghanistan	AFG	1990	0.228979	0.721207	0.131001	
1	Afghanistan	AFG	1991	0.228120	0.719952	0.126395	
2	Afghanistan	AFG	1992	0.227328	0.718418	0.121832	
3	Afghanistan	AFG	1993	0.226468	0.717452	0.117942	
4	Afghanistan	AFG	1994	0.225567	0.717012	0.114547	

MERGING TWO DATASETS

```
data = pd.merge(df1, df2)
data.head()
```

1 to 5 of 5 entries Filter

index	Entity	Code	Year	DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Perce
0	Afghanistan	AFG	1990	1.696670
1	Afghanistan	AFG	1991	1.734281
2	Afghanistan	AFG	1992	1.791189
3	Afghanistan	AFG	1993	1.776779
4	Afghanistan	AFG	1994	1.712985

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

DATA CLEANING

```
data.isnull().sum()
```

```
Entity                                                                                0
Code                                                                                690
Year                                                                                  0
DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)      0
Prevalence - Schizophrenia - Sex: Both - Age: Age-standardized (Percent)              0
Prevalence - Bipolar disorder - Sex: Both - Age: Age-standardized (Percent)           0
Prevalence - Eating disorders - Sex: Both - Age: Age-standardized (Percent)           0
Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)          0
Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)         0
Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)       0
Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)      0
dtype: int64
```

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

```
data.size,data.shape
```

```
(68400, (6840, 10))
```

```
data.set_axis(['Country','Year','Schizophrenia', 'Bipolar_disorder', 'Eating_disorder','Anxiety','drug_usage','depressi
```

```
data.head()
```

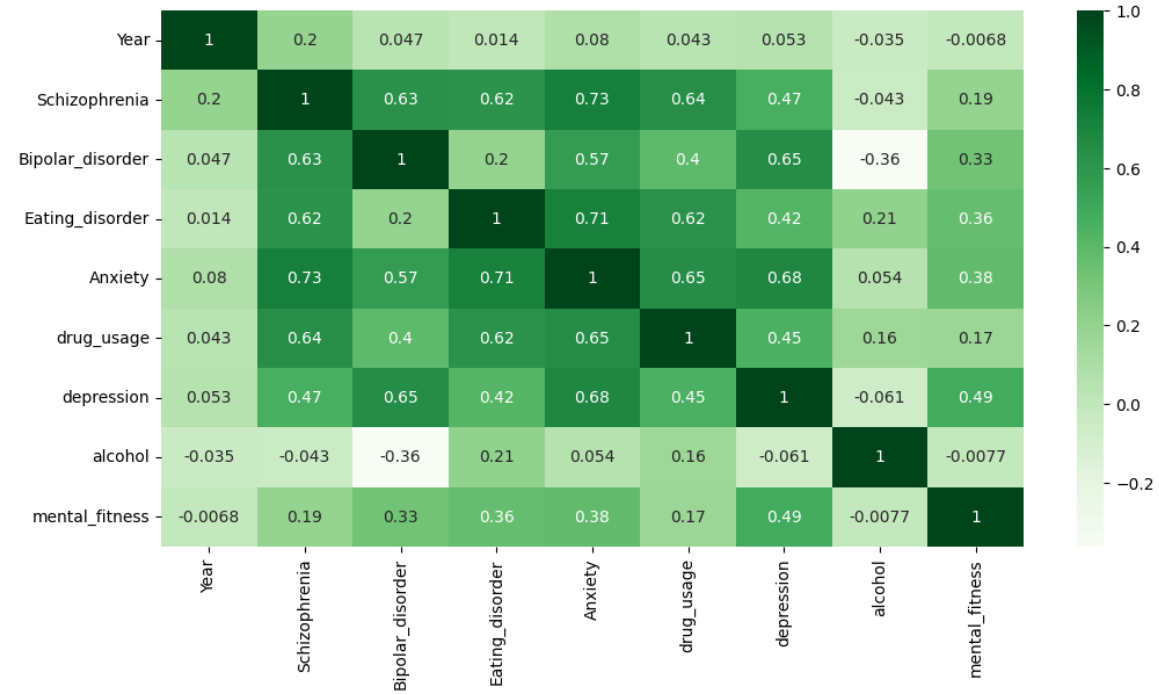
	Country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	An
0	Afghanistan	1990	1.696670	0.228979	0.721207	0.
1	Afghanistan	1991	1.734281	0.228120	0.719952	0.
2	Afghanistan	1992	1.791189	0.227328	0.718418	0.
3	Afghanistan	1993	1.776779	0.226468	0.717452	0.
4	Afghanistan	1994	1.712986	0.225567	0.717012	0.

EXPLORATORY ANALYSIS

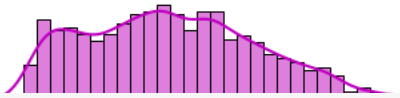
```
plt.figure(figsize=(12,6))
sns.heatmap(data.corr(),annot=True,cmap='Greens')
```

```
plt.plot()
```

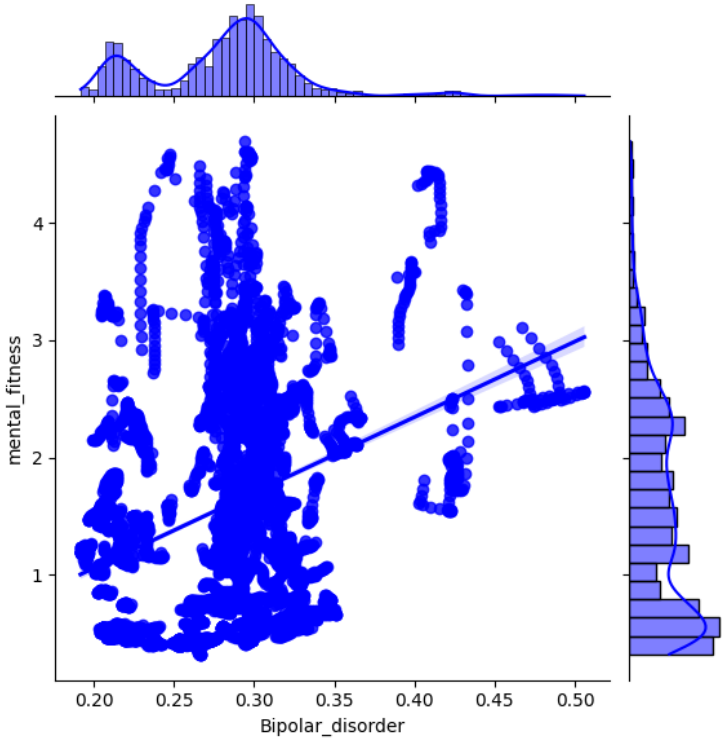
```
[]
```



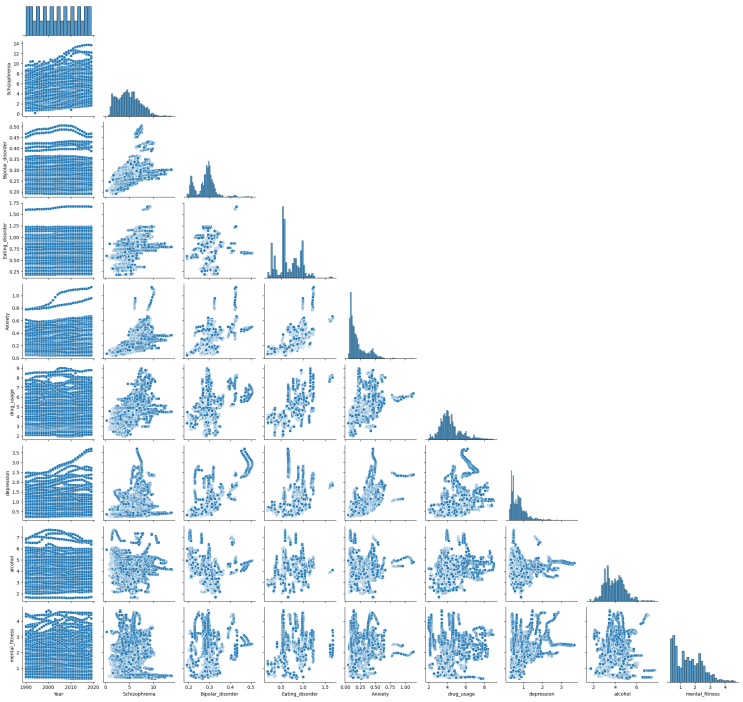

```
sns.jointplot(data,x="Schizophrenia",y="mental_fitness",kind="reg",color="m")
plt.show()
```



```
sns.jointplot(data,x='Bipolar_disorder',y='mental_fitness',kind='reg',color='blue')
plt.show()
```



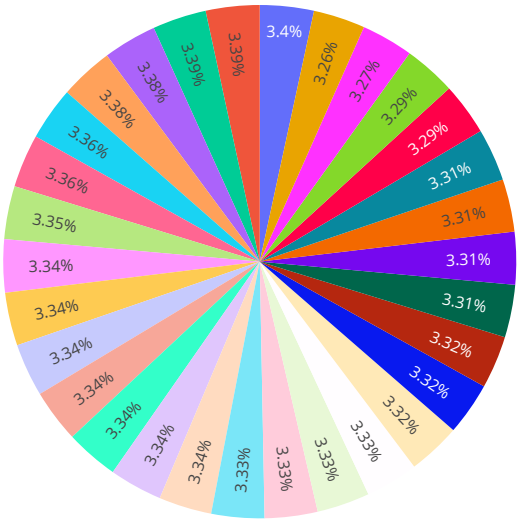

```
sns.pairplot(data,corner=True)
plt.show()
```


Year
Schizophrenia
Bipolar_disorder
Eating_disorder
Anxiety
drug_usage
depression
alcohol
mental_fitness

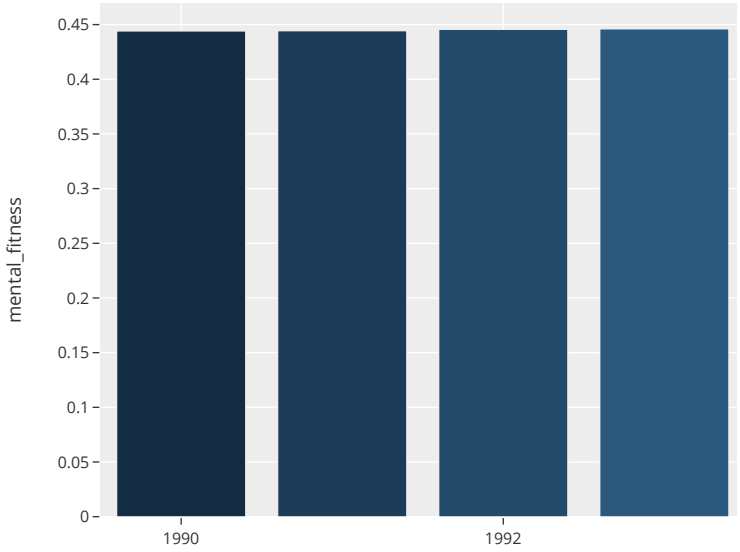
```
mean = data['mental_fitness'].mean()
mean
```

```
1.5788071625382236
```

```
fig = px.pie(data, values='mental_fitness', names='Year')
fig.show()
```

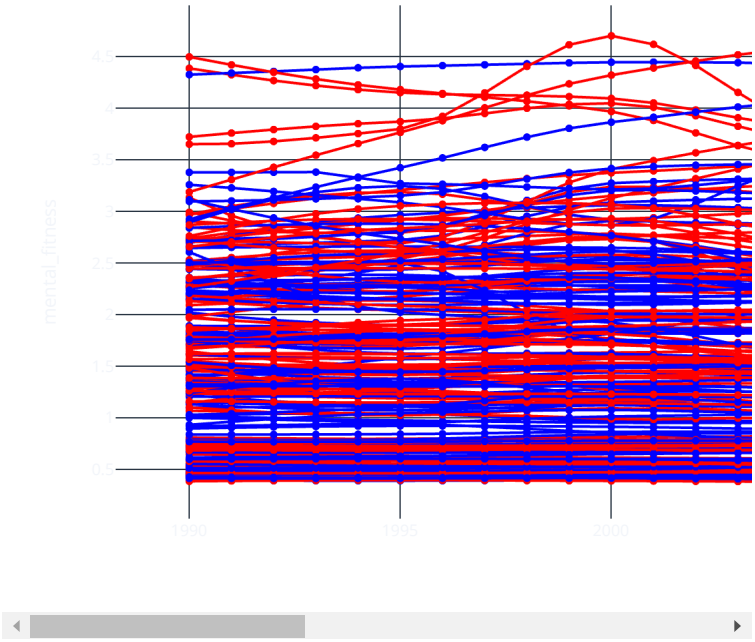



```
fig=px.bar(data.head(10),x='Year',y='mental_fitness',color='Year',template='ggplot2')
fig.show()
```



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'],t
fig.show()
```




```
df=data.copy()
df.head()
```

	Country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	An
0	Afghanistan	1990	1.696670	0.228979	0.721207	0.
1	Afghanistan	1991	1.734281	0.228120	0.719952	0.
2	Afghanistan	1992	1.791189	0.227328	0.718418	0.
3	Afghanistan	1993	1.776779	0.226468	0.717452	0.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6840 entries, 0 to 6839
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Country           6840 non-null   object 
 1   Year              6840 non-null   int64  
 2   Schizophrenia     6840 non-null   float64
 3   Bipolar_disorder  6840 non-null   float64
 4   Eating_disorder   6840 non-null   float64
 5   Anxiety           6840 non-null   float64
 6   drug_usage        6840 non-null   float64
 7   depression        6840 non-null   float64
 8   alcohol           6840 non-null   float64
 9   mental_fitness    6840 non-null   float64
dtypes: float64(8), int64(1), object(1)
memory usage: 587.8+ KB
```

```
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)
```

```
xtrain:  (5472, 9)
xtest:  (1368, 9)
ytrain:  (5472,)
ytest:  (1368,)
```

MULTIPLE LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Create and fit the Multiple Linear Regression model
lr = LinearRegression()
```

```
lr.fit(xtrain, ytrain)

# Model evaluation for the training set
ytrain_pred = lr.predict(xtrain)
mse_train = mean_squared_error(ytrain, ytrain_pred)
rmse_train = np.sqrt(mean_squared_error(ytrain, ytrain_pred))
r2_train = r2_score(ytrain, ytrain_pred)

print("The model performance for the training set")
print("-----------------------------------------")
print("MSE is {}".format(mse_train))
print("RMSE is {}".format(rmse_train))
print("R2 score is {}".format(r2_train))
print("\n")

# Model evaluation for the testing set
ytest_pred = lr.predict(xtest)
mse_test = mean_squared_error(ytest, ytest_pred)
rmse_test = np.sqrt(mean_squared_error(ytest, ytest_pred))
r2_test = r2_score(ytest, ytest_pred)

print("The model performance for the testing set")
print("-----------------------------------------")
print("MSE is {}".format(mse_test))
print("RMSE is {}".format(rmse_test))
print("R2 score is {}".format(r2_test))
```

```
The model performance for the training set
-----------------------------------------
MSE is 0.5768675399720521
RMSE is 0.7595179655360709
R2 score is 0.3358121167259167


The model performance for the testing set
-----------------------------------------
MSE is 0.5792230513574372
RMSE is 0.7610670478725493
R2 score is 0.35130869036834456
```

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))
```

```
The model performance for training set
-------------------------------------
MSE is 0.0006165398418637139
RMSE is 0.024830220334578466
R2 score is 0.9992901346251144


The model performance for testing set
-------------------------------------
MSE is 0.0034482704431171986
RMSE is 0.058721975810740554
R2 score is 0.9961381663515159
```

✓ 0s completed at 8:23 PM