Importing Required Modules

```
import warnings
warnings.filterwarnings('ignore')
```

```
import numpy as np
import pandas as pd
```

```
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from xgboost import XGBRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
```

Mounting Google Drive

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

```
Mounted at /content/drive
```

```
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

Reading data from csv files

Note: If does not works, one download the same datasets (**csv files**) manually from the links:

- Link 1 - **Kaggle**
- Link 2 - **Google Drive**
- Link 3 - **Our World India**

```
df1=pd.read_csv("/content/prevalence-by-mental-and-substance-use-disorder _AI.csv")
df2=pd.read_csv("/content/mental-and-substance-use-as-share-of-disease -AI.csv")
```

Data Exploration

```
df1.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)	Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)
0	Afghanistan	AFG	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036
1	Afghanistan	AFG	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250
2	Afghanistan	AFG	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501
3	Afghanistan	AFG	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958
4	Afghanistan	AFG	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779

```
df2.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
4	Afghanistan	AFG	1994	1.712986

Merging the two datasets into one

```
data=pd.merge(df1,df2)
data.head(10)
```

	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)	Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)
0	Afghanistan	AFG	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036
1	Afghanistan	AFG	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250

Filtering out the null data or unlabelled data

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

```
Entity                                                                               0
Code                                                                               690
Year                                                                                 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
DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)      0
dtype: int64
```

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

Filtered data

```
1 data.head(10)
```

	Entity	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)	Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Depressive disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)	(Disa A Life - diso Sex: A (P
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036	1
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250	1
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501	1
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958	1
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779	1
5	Afghanistan	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5.098495	0.445422	1
6	Afghanistan	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5.100580	0.444837	1
7	Afghanistan	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5.105474	0.443938	1
8	Afghanistan	1998	0.221129	0.716139	0.100343	4.777377	0.422491	5.113707	0.442665	1
9	Afghanistan	1999	0.220065	0.716323	0.097946	4.782067	0.421215	5.120480	0.441428	1

Filtered dataset size

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

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

Renaming columns as per our convenience

```
1 data.set_axis(["Country","Year","Schizophrenia","Bipolar_disorder","Eating_disorder","Anxiety","drug_usage","depression","alcohol","mental_fitness"], axis="columns", inplace=True)
```

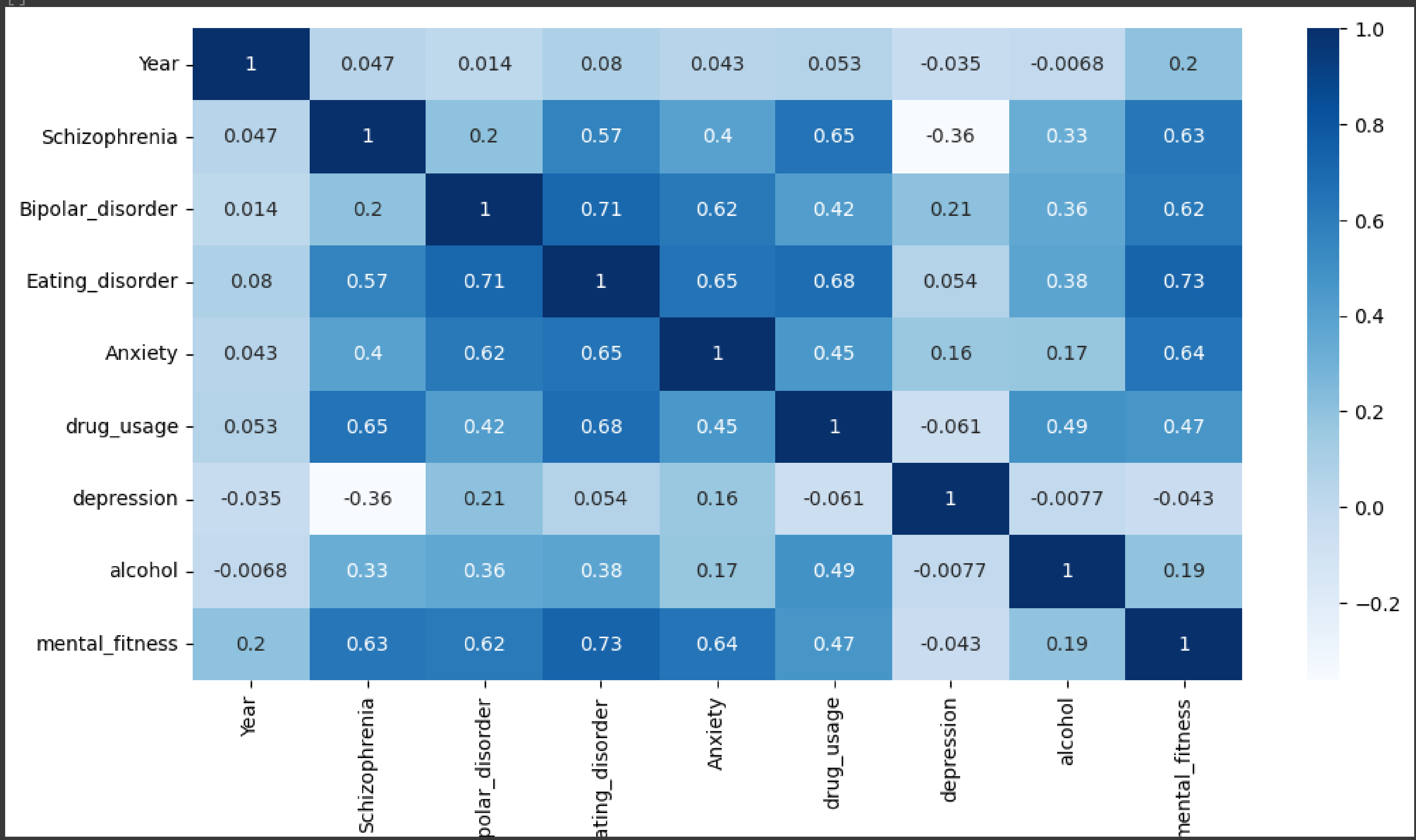
```
1 data.head()
```

	Country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	Anxiety	drug_usage	depression	alcohol	mental_fitn
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036	1.696
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250	1.734
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501	1.791
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958	1.776
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779	1.712

Visual Analysis

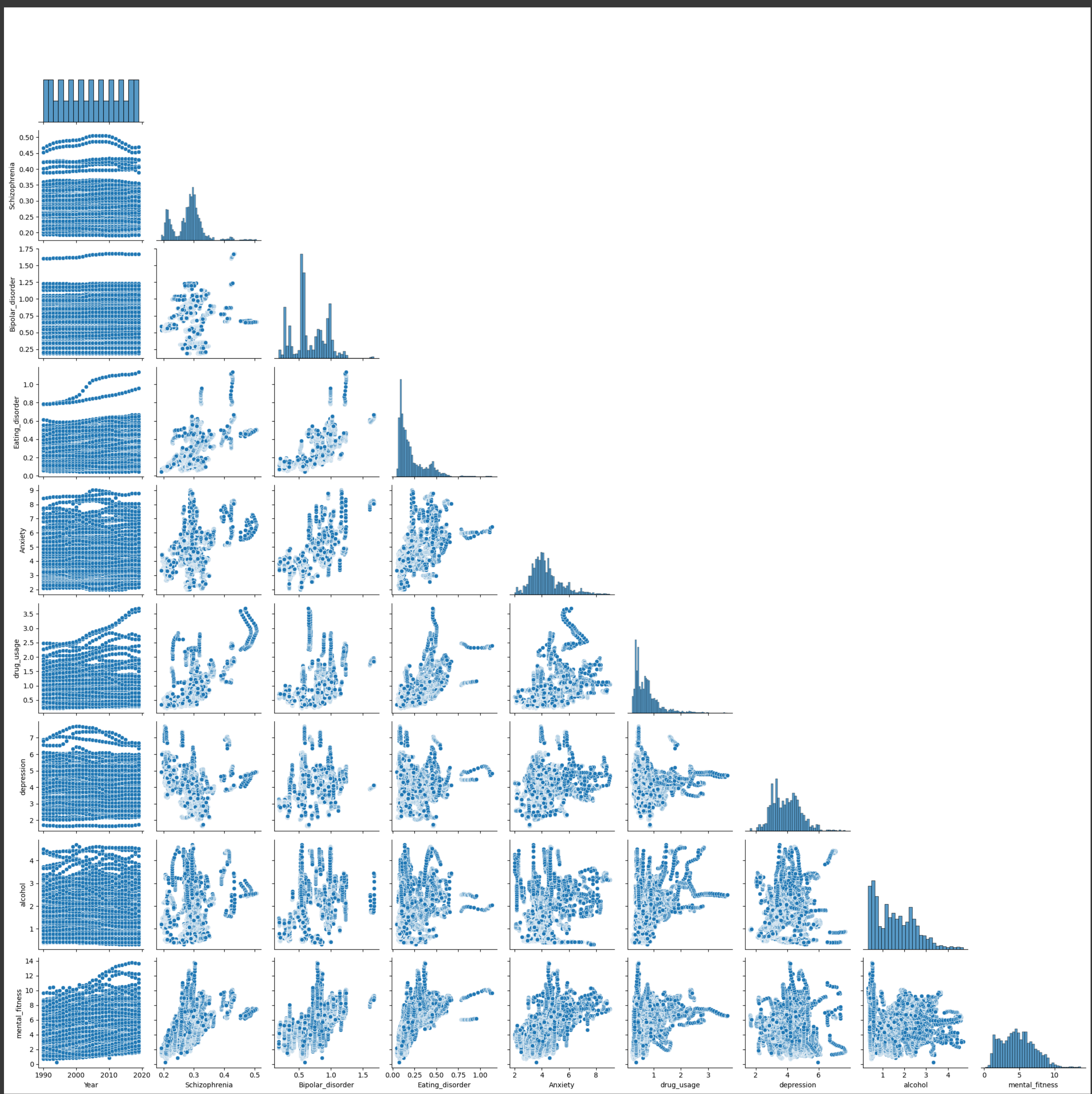
The correlation heatmap of diseases and mental fitness

```
1 plt.figure(figsize=(12,6))
2 sns.heatmap(data.corr(),annot=True,cmap='Blues')
3 plt.plot()
```



Pairplot of each feature

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



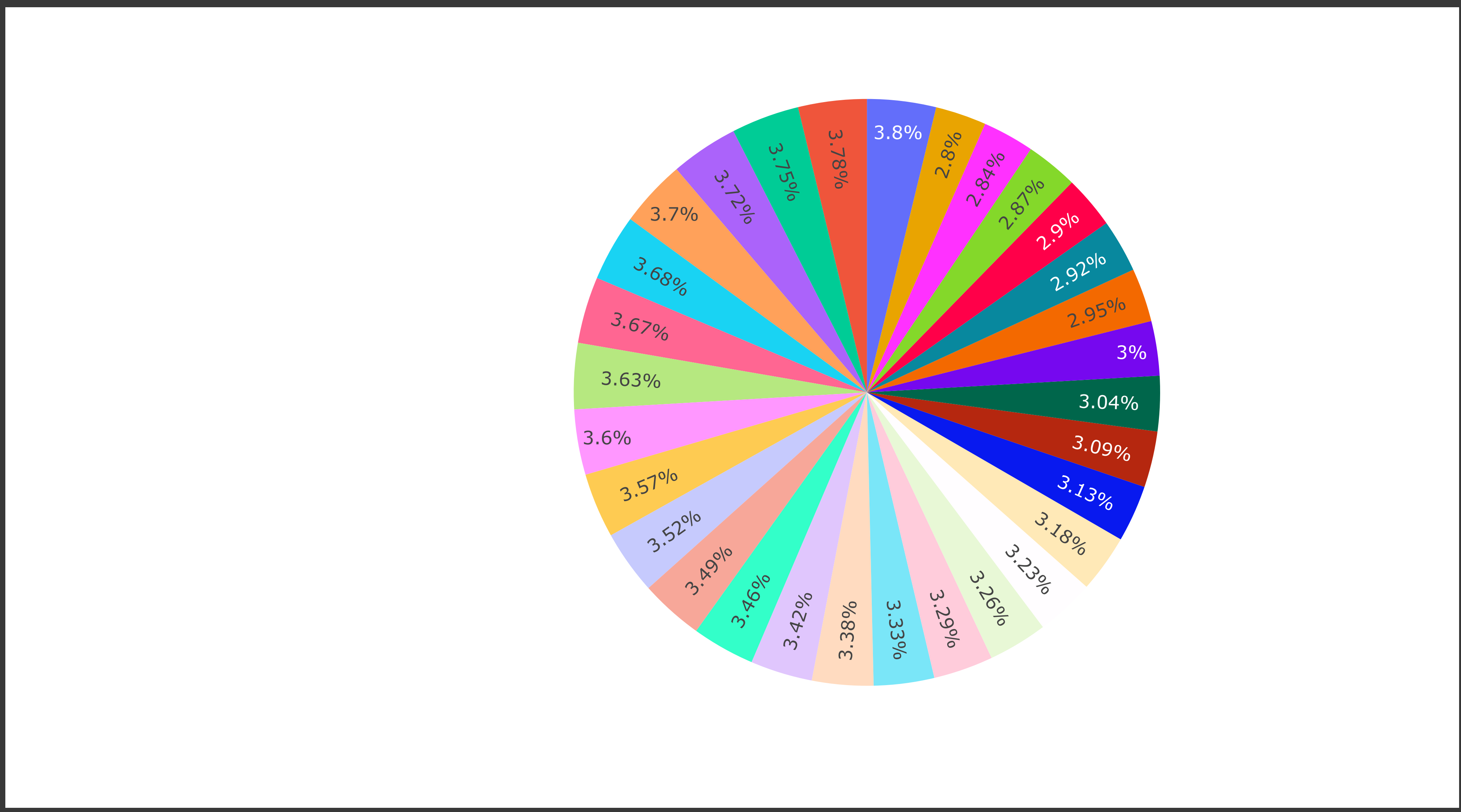
Mean of mental fitness column of the data dataframe

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

```
4.8180618117506135
```

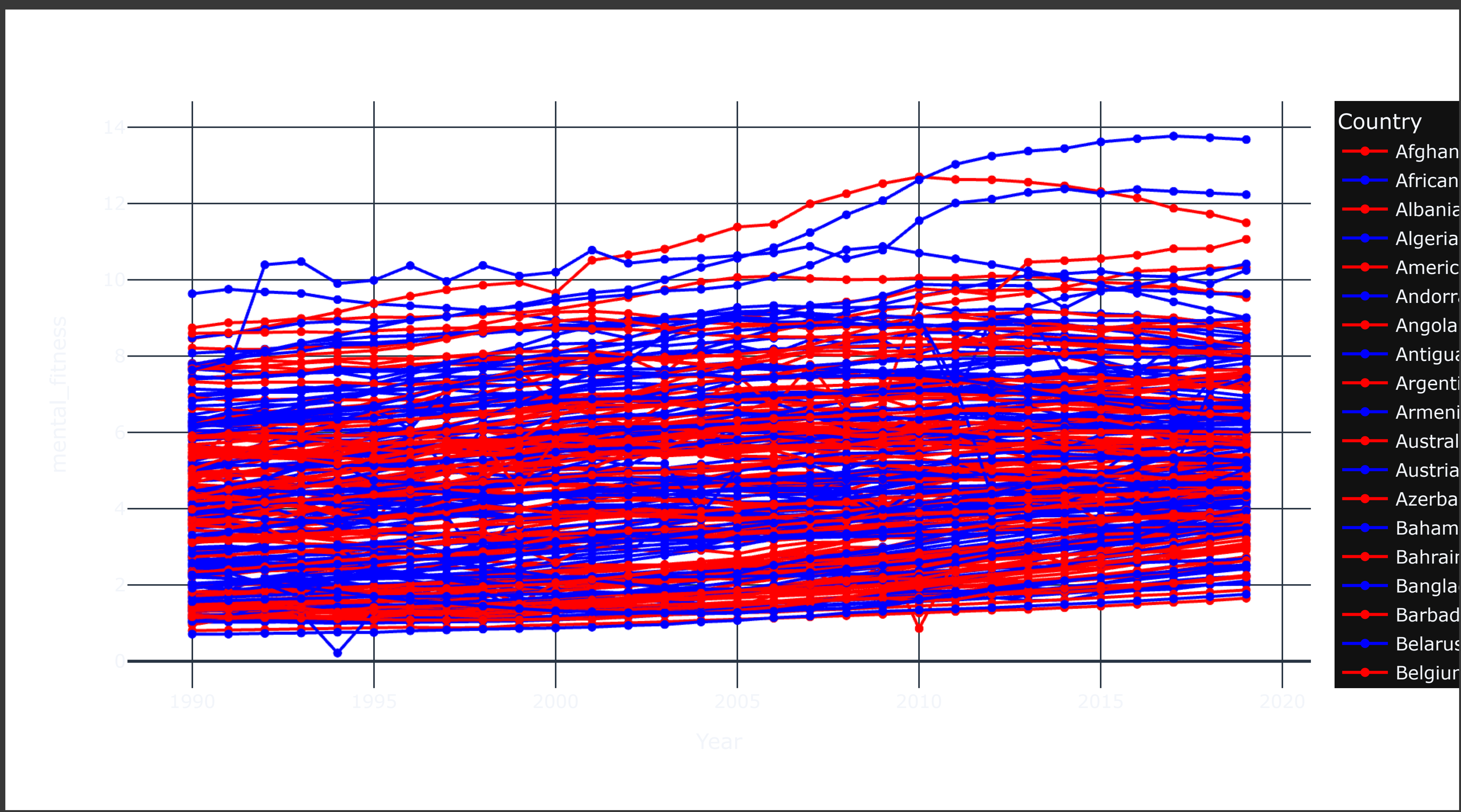
Pie Chart

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



Year wise variation in mental fitness in 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()
```



Describing our data i.e. dataframe object

```
df=data
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
```

Fitting the data for further processing

```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
for i in df.columns:
    if df[i].dtype == 'object':
        df[i]=l.fit_transform(df[i])
```

```
df.shape
```

```
(6840, 10)
```

Splitting our dataset into two halves i.e. testing and training set

```
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 :", xtest.shape)
```

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

Using Random Forest Algorithm

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
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(mse))
r2 = r2_score(ytrain, ytrain_pred)
print("The model performance for training set")
print("--------------------------------------")
print(f'MSE is {mse}')
print(f'RMSE is {rmse}')
print(f'R2 score is {r2}')
print()

# 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(f'MSE is {mse}')
print(f'RMSE is {rmse}')
print(f'R2 score is {r2}')
```

```
The model performance for training set
--------------------------------------
MSE is 0.005259863595446224
RMSE is 0.07252491706611063
R2 score is 0.9990211243170465

The model performance for testing set
--------------------------------------
MSE is 0.03115675220203062
RMSE is 0.17651275365262029
R2 score is 0.9935230811549247
```

Using Linear Regression Algorithm

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet, LinearRegression, BayesianRidge
model = LinearRegression()
model.fit(xtrain, ytrain)
```

```
▾ LinearRegression
LinearRegression()
```

Using pre-trained model

```
y_pred = model.predict(xtest)
```

1. Ridge Regression

```
model = Ridge(alpha=0.5)
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 1.1393226139229882
RMSE is 1.0673905629726113
R2 score is 0.7631556697280758
```

2. Lasso Regression

```
model = Lasso(alpha=0.5)
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
```

```
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 2.7702717436599804
RMSE is 1.6644133331777837
R2 score is 0.4241111799412294
```

3. Elastic Net Regression

```
model = ElasticNet(alpha=0.5, l1_ratio=0.5)
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 2.740266404991704
RMSE is 1.6553750043394106
R2 score is 0.4303487409749741
```

4. Decision Tree Regression

```
model =  DecisionTreeRegressor()
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 0.08630921976031734
RMSE is 0.2937843082268305
R2 score is 0.9820578920310921
```

5. Random Forest Regression

```
  model = RandomForestRegressor()
  model.fit(xtrain, ytrain)
  y_pred = model.predict(xtest)
  mse = mean_squared_error(ytest, y_pred)
  r2 = r2_score(ytest, y_pred)
  rmse = np.sqrt(mse)
  print("The model Performance for testing set")
  print("-------------------------------------")
  print(f"MSE is {mse}")
  print(f"RMSE is {rmse}")
  print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 0.030616600459818923
RMSE is 0.17497599966800853
R2 score is 0.9936353688213556
```

6. Polynomial Regression

```
features = PolynomialFeatures(degree=2)
xpoly = features.fit_transform(xtrain)
model = LinearRegression()
model.fit(xpoly, ytrain)
xtest_poly = features.transform(xtest)
y_pred = model.predict(xtest_poly)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 0.5365987525787108
RMSE is 0.7325290114246061
R2 score is 0.8884509351204317
```

7. Support Vector Regression

```
model = SVR()
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
```

```
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 4.7911713917240055
RMSE is 2.1888744577348436
R2 score is 0.0040031105995593785
```

8. XGBoost Regression

```
model = XGBRegressor()
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 0.04225689361124382
RMSE is 0.2055648160830151
R2 score is 0.9912155648062968
```

9. K - Nearest Neighbour Regression

```
model = KNeighborsRegressor()
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 1.0049803438106724
RMSE is 1.0024870791240514
R2 score is 0.7910829702162167
```

10. Bayesian Ridge Regression

```
model = BayesianRidge()
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 1.135667232782995
RMSE is 1.0656768894852675
R2 score is 0.7639155566006879
```

11. Neural Network Regression

```
model = MLPRegressor(max_iter=1000)
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 2.0379544186653145
RMSE is 1.42756940940373
R2 score is 0.5763465558262661
```

12. Gradient Boosting Regression

```
model = GradientBoostingRegressor()
model.fit(xtrain, ytrain)
y_pred = model.predict(xtest)
mse = mean_squared_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
rmse = np.sqrt(mse)
print("The model Performance for testing set")
print("-------------------------------------")
print(f"MSE is {mse}")
print(f"RMSE is {rmse}")
print(f"R2 score is {r2}")
```

```
The model Performance for testing set
-------------------------------------
MSE is 0.24536599739628345
```

```
RMSE is 0.49534432205919493
R2 score is 0.9489928975211638
```

Results Visualization

```python
# dictionary which stores model performance like <model-name: accuracy>
models_performance = {"Random Forests": 0.9938472950788096,
                      "Elastic Net": 0.4303487409749741,
                      "Polynomail": 0.8884509351204317,
                      "Decision Trees": 0.984631488977419,
                      "Lasso Regression": 0.4241111799412294,
                      "Support Vector Machine": 0.0040031105995593785,
                      "XGBoost Regression": 0.9912155648062968,
                      "K - Nearest Neighbour": 0.7910829702162167,
                      "Bayesian Ridge Regression": 0.7639155566006879,
                      "Neural Networks": 0.7281529631080045,
                      "Gradient Boosting": 0.9490628375125112}

for key, value in models_performance.items():
    models_performance[key] = round(value * 100, 3)
```

```python
print(models_performance)
```

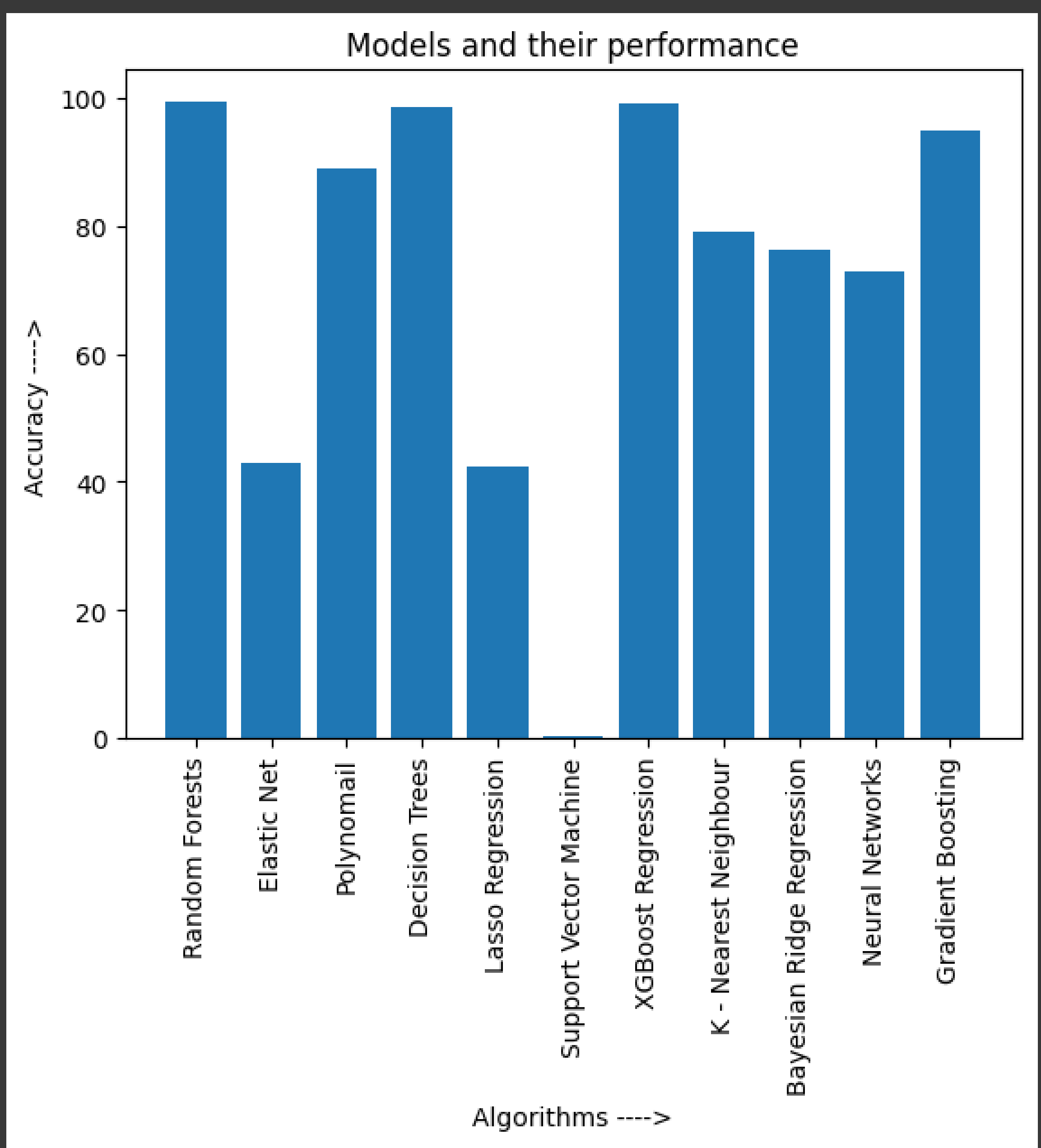
```
{'Random Forests': 99.385, 'Elastic Net': 43.035, 'Polynomail': 88.845, 'Decision Trees': 98.463, 'Lasso Regression': 42.411, 'Support Vector Machine': 0.4, 'XGBoost Regression': 99.122, '
```

```python
def get_key_with_max_and_min_value(dict_data):
    max_value = min_value = dict_data[list(dict_data.keys())[0]]
    max_key = min_key = list(dict_data.keys())[0]

    for key, value in dict_data.items():
        if value > max_value:
            max_value = value
            max_key = key
        elif value < min_value:
            min_value = value
            min_key = key

    return max_key, min_key


max_key, min_key = get_key_with_max_and_min_value(models_performance)
print("The algorithm with the maximum accuracy is:", max_key)
print("The key with the minimum accuracy is:", min_key)
```

```
The algorithm with the maximum accuracy is: Random Forests
The key with the minimum accuracy is: Support Vector Machine
```

Bar graph

```python
import matplotlib.pyplot as plt
plt.bar(models_performance.keys(), models_performance.values())
plt.title("Models and their performance")
plt.xticks(rotation=90)
plt.xlabel("Algorithms ---->")
plt.ylabel("Accuracy ---->")
plt.show()
```



Summary

Algorithm	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R^2
Random Forests	0.031286265113436774	0.1768792387857794	99.385%
Elastic Net Regression	2.740266404991704	1.6553750043394106	43.035%
PLoynomail Regression	0.5365987525787108	0.7325290114246061	88.845%
Decision Trees	0.07392911677576598	0.27189909300283804	98.463%
Lasso Regression	2.7702717436599804	1.6644133331777837	42.411%
Support Vector Machine (SVM)	4.7911713917240055	2.1888744577348436	0.4%
XGBoost Regression	0.04225689361124382	0.2055648160830151	99.122%

Algorithm	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R^2
K-Nearest Neighbour (KNN)	1.0049803438106724	1.0024870791240514	79.108%
Bayesian Ridge Regression	1.135667232782995	1.0656768894852675	76.392%
Neural Networks	1.30770061627995	1.1435473826125222	72.815%
Gradient Boosting	0.24502955609887844	0.49500460209868596	94.906%

```
np.random.seed(range(0,100))
print("Hi!\nWelcome to Mental Fitness Tracker!\nFill the detail to check your mental fitness!")
country=l.fit_transform([input("Enter Your country Name : ")])
year=int(input("Enter the Year : "))
schi=(float(input("Enter your Schizophrenia rate in % (it not enter 0) : ")))
bipo_dis=(float(input("Enter your Bipolar disorder rate in % (it not enter 0) : ")))
eat_dis=(float(input("Enter your Eating disorder rate in % (it not enter 0) : ")))
anx=(float(input("Enter your Anxiety rate in % (it not enter 0) : ")))
drug_use=(float(input("Enter your Drug Usage rate in per year % (it not enter 0) : ")))
depr=(float(input("Enter your Depression rate in % (it not enter 0) : ")))
alch=(float(input("Enter your Alcohol Consuming rate per year in % (it not enter 0) : ")))

prediction=rf.predict([[country,year,schi,bipo_dis,eat_dis,anx,drug_use,depr,alch]])
print(f"Your Mental Fitness is {prediction[0]/10:.3%}")
```

```
Hi!
Welcome to Mental Fitness Tracker!
Fill the detail to check your mental fitness!
Enter Your country Name : India
Enter the Year : 2012
Enter your Schizophrenia rate in % (it not enter 0) : 3
Enter your Bipolar disorder rate in % (it not enter 0) : 23
Enter your Eating disorder rate in % (it not enter 0) : 54
Enter your Anxiety rate in % (it not enter 0) : 64
Enter your Drug Usage rate in per year % (it not enter 0) : 78
Enter your Depression rate in % (it not enter 0) : 65
Enter your Alcohol Consuming rate per year in % (it not enter 0) : 97
Your Mental Fitness is 96.746%
```

✓ 31s completed at 12:11 PM