```
# Step 1: Import the required libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Step 2: Load your dataset from the CSV file
data = pd.read_csv("prevalence-by-mental.csv")
```

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

```
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
```

Exploratory data Analysis

```
data2 = pd.read_csv("mental-disease -AI.csv")
```

```
data.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)	Preval Depr disor Sex: Age standa (Pe
0	Afghanistan	AFG	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5
1	Afghanistan	AFG	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5
2	Afghanistan	AFG	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5
3	Afghanistan	AFG	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5
4	Afghanistan	AFG	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5

```
data2.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 2 datasets

```
data1=pd.merge(data,data2)
```

```
data1.head()
```

	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 - Depressiv disorders - Sex: Both - Age: Age standardize (Percent
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.11630

Data *Cleaning*

```
data1.isnull().sum()
```

```
                 0
Country          0
Year             0
Schizophrenia    0
Bipolar          0
Eating           0
Anxiety          0
Drug use         0
Depressive       0
Alcohol        690
MentalFitness    0
dtype: int64
```

```
data1.head()
```

	Country	Year	Schizophrenia	Bipolar	Eating	Anxiety	Drug use	Depressive	Alcohol	MentalFitn
0	0	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036	1.696
1	0	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250	1.734
2	0	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501	1.791
3	0	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958	1.776
4	0	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779	1.712

```
data1.size, data1.shape
```

```
(75240, (6840, 11))
```

```
# colomn Set

data1.set_axis(['Country','Year', 'Schizophrenia','Bipolar','Eating','Anxiety','Drug use','Depressive','Alcohol','Code','MentalFitness'],

data1.head()
```

	Country	Year	Schizophrenia	Bipolar	Eating	Anxiety	Drug use	Depressive	Alcohol	Code	Me
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036	AFG	
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250	AFG	
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501	AFG	
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958	AFG	
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779	AFG	

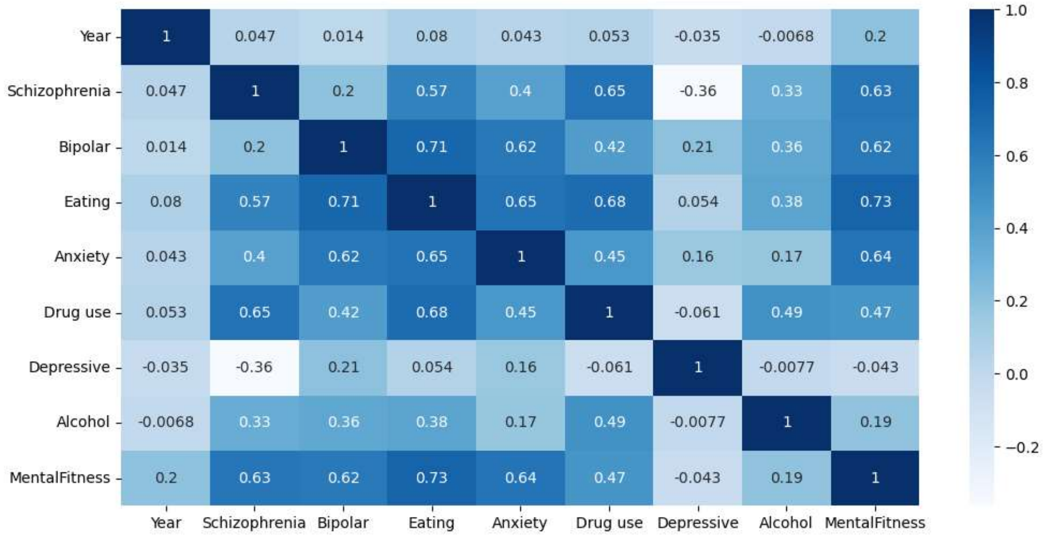
Values

Target colomn is mental fitness

data visualization

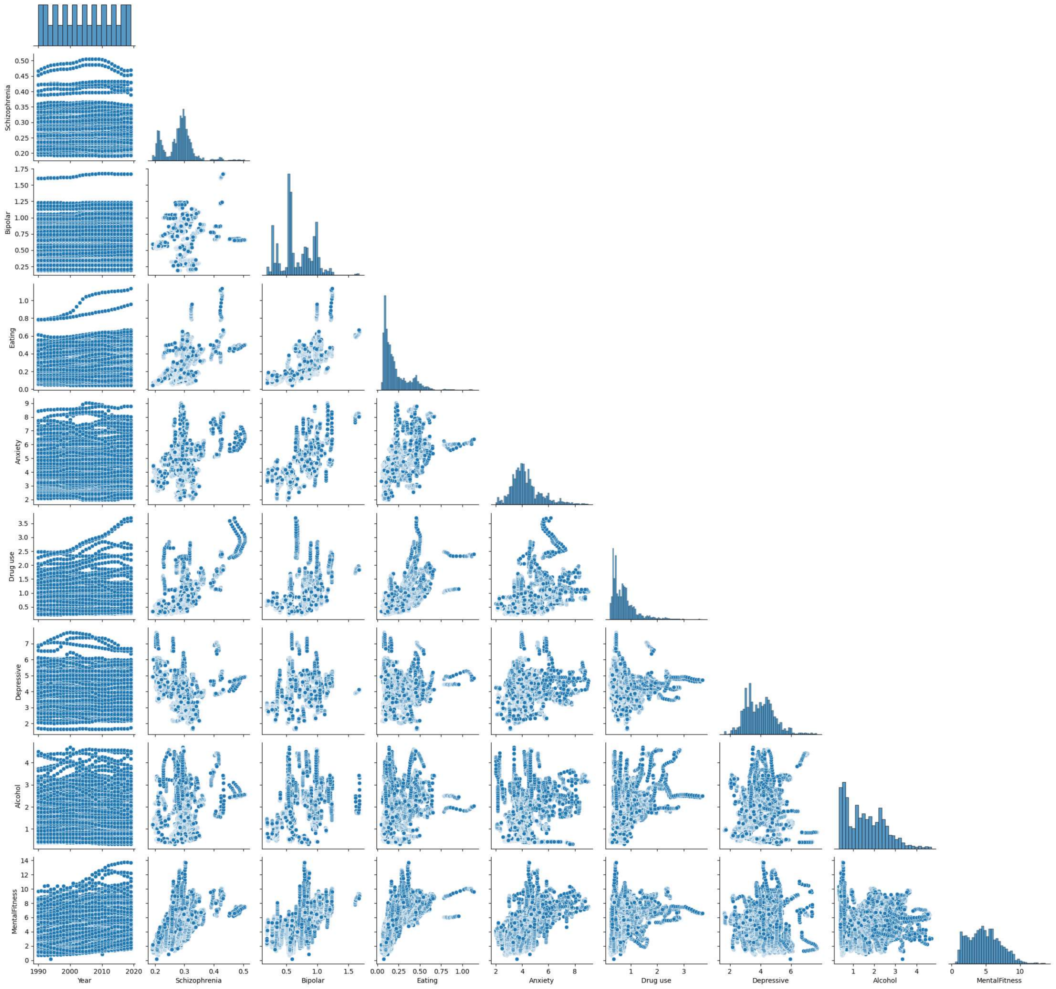
```
plt.figure(figsize=(12,6))
sns.heatmap(data1.corr(),annot=True,cmap='Blues')
plt.plot()
```

```
[]
```



Heat map shows how 1 feature is correlated to another and if it goes towards 1 means they are highly related

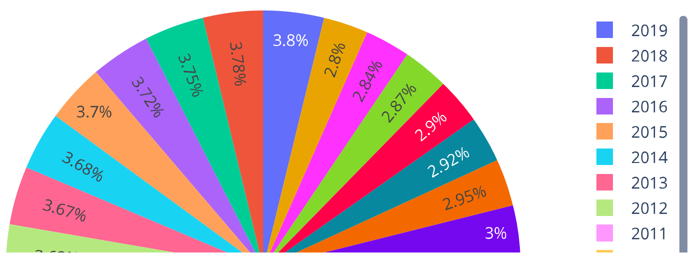
```
sns.pairplot(data1,corner=True)
plt.show()
```



```
mean = data1['MentalFitness'].mean()
mean
```

```
     4.8180618117506135
```

```
fig = px.pie(data1,values='MentalFitness', names='Year')
fig.show()
```



```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6840 entries, 0 to 6839
Data columns (total 11 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        6840 non-null   float64
 4   Eating         6840 non-null   float64
 5   Anxiety        6840 non-null   float64
 6   Drug use       6840 non-null   float64
 7   Depressive     6840 non-null   float64
 8   Alcohol        6840 non-null   float64
 9   Code           6150 non-null   object 
 10  MentalFitness  6840 non-null   float64
dtypes: float64(8), int64(1), object(2)
memory usage: 899.3+ KB
```

```
column_to_drop = "Code"  # Replace "column_name" with the name of the column you want to drop
data1.drop(column_to_drop, axis=1, inplace=True)
```

```
data1.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        6840 non-null   float64
 4   Eating         6840 non-null   float64
 5   Anxiety        6840 non-null   float64
 6   Drug use       6840 non-null   float64
 7   Depressive     6840 non-null   float64
 8   Alcohol        6840 non-null   float64
 9   MentalFitness  6840 non-null   float64
dtypes: float64(8), int64(1), object(1)
memory usage: 845.9+ KB
```

```
from sklearn.preprocessing import LabelEncoder
# Transfer non numeric data to numeric labels
l= LabelEncoder()
for i in data1.columns:
  if data1[i].dtype == 'object':
    data1[i]=l.fit_transform(data1[i])
```

```
data1.shape
```

```
(6840, 10)
```

Split the data

```
# Training an testing
x = data1.drop('MentalFitness',axis=1)
y = data1['MentalFitness']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
```

```
print(" X train : ", x_train)
print(" X test : ", x_test)
print(" X tarain : ", y_train)
print(" X test : ", y_test)
```

```
 X train :        Country  Year  Schizophrenia   Bipolar    Eating   Anxiety  Drug use  \
839        27  2019       0.329339  0.656791  0.459976  3.093946  0.859447
5815      193  2015       0.256836  0.814188  0.174171  5.867417  0.364848
4405      146  2015       0.353118  0.832142  0.425403  5.065427  1.679728
3813      127  1993       0.305036  0.576889  0.145265  3.892060  0.651911
3442      114  2012       0.326715  0.357418  0.145606  4.838487  0.585521
...       ...   ...            ...       ...       ...       ...       ...
6443      214  2013       0.281855  0.538326  0.110408  2.122076  0.466393
3606      120  1996       0.307256  0.347041  0.113928  3.925706  0.624032
5704      190  1994       0.279838  0.926735  0.196345  4.227663  0.688639
6637      221  1997       0.356200  0.790745  0.417114  5.221426  1.463586
2575       85  2015       0.271403  0.844266  0.167731  4.232530  0.507362

      Depressive   Alcohol
839     1.736138  0.975514
5815    4.231454  0.420401
4405    3.939009  2.055928
3813    3.042269  2.054189
3442    3.679132  0.531266
...          ...       ...
6443    3.341884  1.916988
3606    4.315890  1.263044
5704    4.648631  1.487007
6637    3.878680  2.123414
2575    3.293464  2.436116

[5472 rows x 9 columns]
 X test :        Country  Year  Schizophrenia   Bipolar    Eating   Anxiety  Drug use  \
4143      138  1993       0.220314  0.557777  0.099236  2.971630  0.271602
1260       42  1990       0.323175  0.295753  0.130021  4.100610  0.772461
4329      144  1999       0.336209  0.296241  0.175765  4.202445  0.779471
2261       75  2001       0.215107  0.550132  0.097027  2.985863  0.442802
2434       81  1994       0.211178  0.541067  0.089330  3.327791  0.249130
...       ...   ...            ...       ...       ...       ...       ...
1511       50  2001       0.237134  1.044455  0.481012  5.320241  1.075572
3095      103  1995       0.270223  0.332577  0.070168  4.361768  0.461737
5570      185  2010       0.287437  0.357223  0.106436  3.183147  0.464734
1556       51  2016       0.227665  0.630480  0.110770  3.379523  0.374533
2957       98  2007       0.292652  0.538307  0.143324  2.245902  0.856730

      Depressive   Alcohol
4143    4.224348  0.481681
1260    3.282559  0.798645
4329    2.814324  1.008391
2261    4.417528  0.530308
2434    4.283293  0.473574
...          ...       ...
1511    4.170109  3.180662
3095    3.016238  0.925335
5570    3.596055  1.168124
1556    4.605067  1.400175
2957    3.695706  3.932414

[1368 rows x 9 columns]
 X tarain :  839     6.056265
5815    4.583907
4405    6.861558
3813    5.304653
```

Model training

Linear Regression

```
# Step 5: Create and train the linear regression model
regression_model = LinearRegression()
regression_model.fit(x_train, y_train)
```

```
▸ LinearRegression
```

```
# Step 6: Make predictions on the test set
y_pred = regression_model.predict(x_test)
```

```
# Step 7: Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
# Step 8: Print the evaluation metrics
print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
     Mean Squared Error: 1.1357545319272384
     R-squared: 0.7638974087055272
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Separate features (X) and target (y)
X = data1.drop(columns=['MentalFitness'])  # Replace 'target_column_name' with the name of your target column
y = data1['MentalFitness']
```

```
# Split the data into 80% training and 20% testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Initialize the Random Forest regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model on the training data
rf_regressor.fit(X_train, y_train)
```

```
      ▾        RandomForestRegressor
      RandomForestRegressor(random_state=42)
```

```
# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)

# Calculate Mean Squared Error and R-squared score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared score:", r2)
```

```
     Mean Squared Error: 0.03028511515868332
     R-squared score: 0.9940461716663964
```

✓ 0s completed at 03:24 ● ×