Mental Fitness Tracker :

> The Mental Fitness Tracker is an AI-powered project aimed at monitoring and supporting mental well-being using advanced algorithms. It leverages regression models to analyze and predict mental fitness based on various factors. The project utilizes a dataset containing information about mental and substance use disorders, prevalence rates, and other relevant factors.

About the Data:

- **Data Overview:** This is a "Mental Health".csv data

Import Required Libraries:

```
# Ignore warning messages to prevent them from being displayed during code execution
import warnings
warnings.filterwarnings('ignore')
```

```
import numpy as np    # Importing the NumPy library for linear algebra operations
import pandas as pd   # Importing the Pandas library for data processing and CSV file handling
```

```
from google.colab import drive  # Importing the necessary library to mount Google Drive
drive.mount('/content/drive')   # Mounting Google Drive to the specified directory '/content/drive'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import seaborn as sns                  # Importing the Seaborn library for statistical data visualization
import matplotlib.pyplot as plt        # Importing the Matplotlib library for creating plots and visualizations
import plotly.express as px            # Importing the Plotly Express library for interactive visualizations

```

Exploratory Data Analysis

Load and Prepare Data:

```
# Reading the CSV file containing prevalence data for mental and substance use disorder
df1 = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/prevalence-by-mental-and-substance-use-disorder.csv")

# Reading the CSV file containing data on mental and substance use as a share of disease
df2 = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/mental-and-substance-use-as-share-of-disease.csv")
```

```
df1.head()
# Displaying the first few rows of the DataFrame df1
```

	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

```
df2.head(10)
# Displaying the first few rows of the DataFrame df2
```

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

```
data = pd.merge(df1, df2)
data.head(10)
# Merging df1 and df2 into a single DataFrame and displaying the first 10 rows of the merged DataFrame

```

	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
5	Afghanistan	AFG	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5
6	Afghanistan	AFG	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5
7	Afghanistan	AFG	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5

Data Cleaning

```
data.isnull().sum()
# Checking the count of missing values (null values) in each column of the DataFrame data

```

```
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
```

```
data.drop('Code', axis=1, inplace=True)
# Dropping the 'Code' column from the DataFrame data
```

```
data.head(10)
# Displaying the first 10 rows of the DataFrame data
```

	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)
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.12529
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.11630
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.10655
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.10032
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.09942
5	Afghanistan	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5.09849
6	Afghanistan	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5.10058
7	Afghanistan	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5.10547

```
data.size, data.shape
# Calculating the size (total number of elements) and shape (number of rows, number of columns) of the DataFrame data
```

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

```
data.set_axis(['country', 'Year', 'Schizophrenia', 'Bipolar_disorder', 'Eating_disorder', 'Anxiety', 'drug_usage', 'depression', 'alcohol', 'mental_fitness'
# Renaming the columns of the DataFrame data with more descriptive names
```

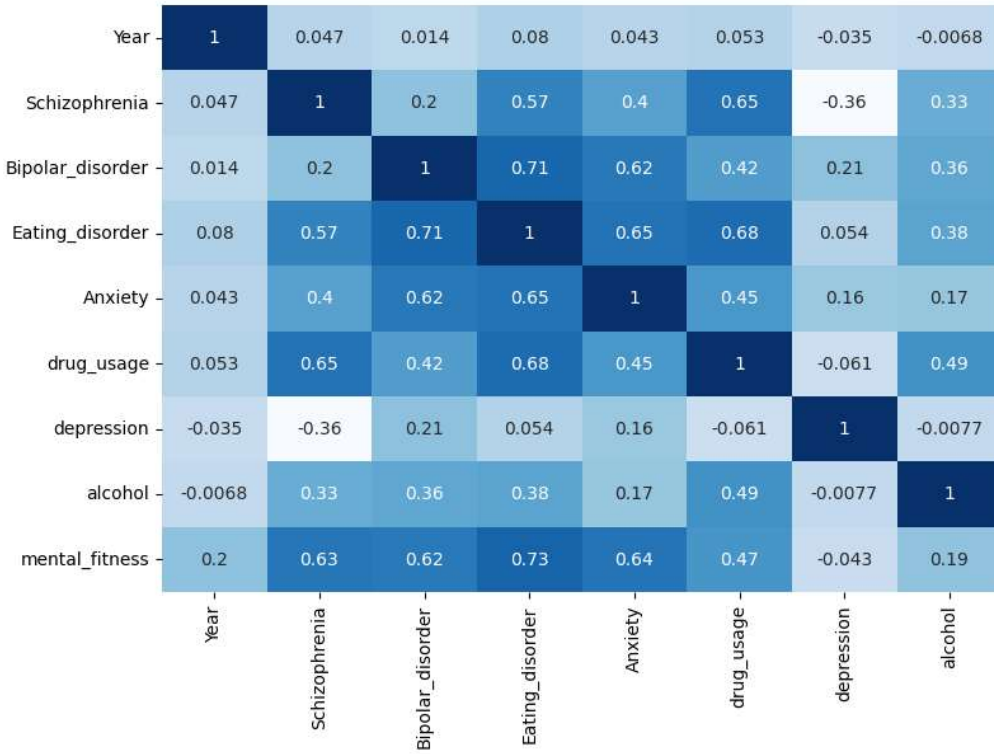
```
data.head(10)
```

	country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	Anxiety	drug_usage	depression
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424
5	Afghanistan	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5.098495
6	Afghanistan	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5.100580
7	Afghanistan	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5.105474
8	Afghanistan	1998	0.221129	0.716139	0.100343	4.777377	0.422491	5.113707
9	Afghanistan	1999	0.220065	0.716323	0.097946	4.782067	0.421215	5.120480

Visulaization

```
plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(), annot=True, cmap='Blues')
plt.plot()
# Creating a heatmap of the correlation matrix for the columns in the DataFrame data
```

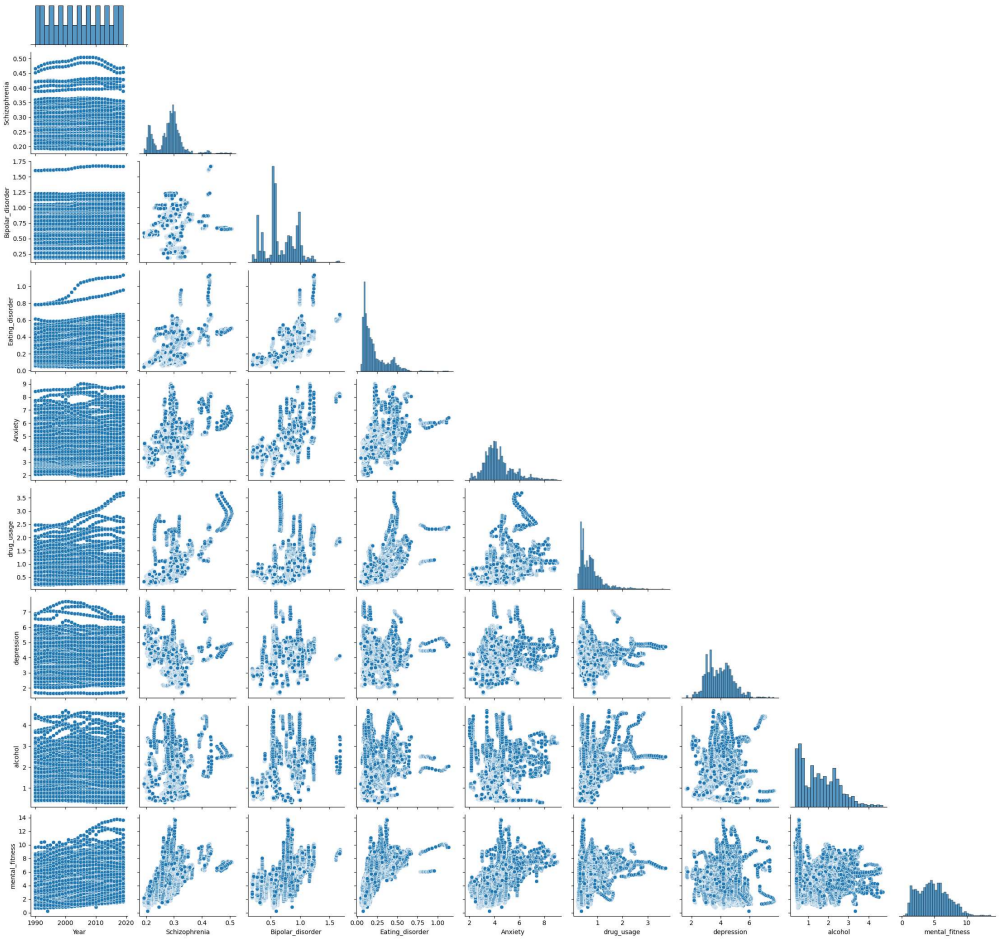
```
[]
```



Key Points:

> Eating_disorder is positively Correlated to mental_fitness and vice-versa as our eating choice affect our mental health

```
sns.pairplot(data, corner=True)
plt.show()
# Creating a pairwise scatter plot matrix for the columns in the DataFrame data
```

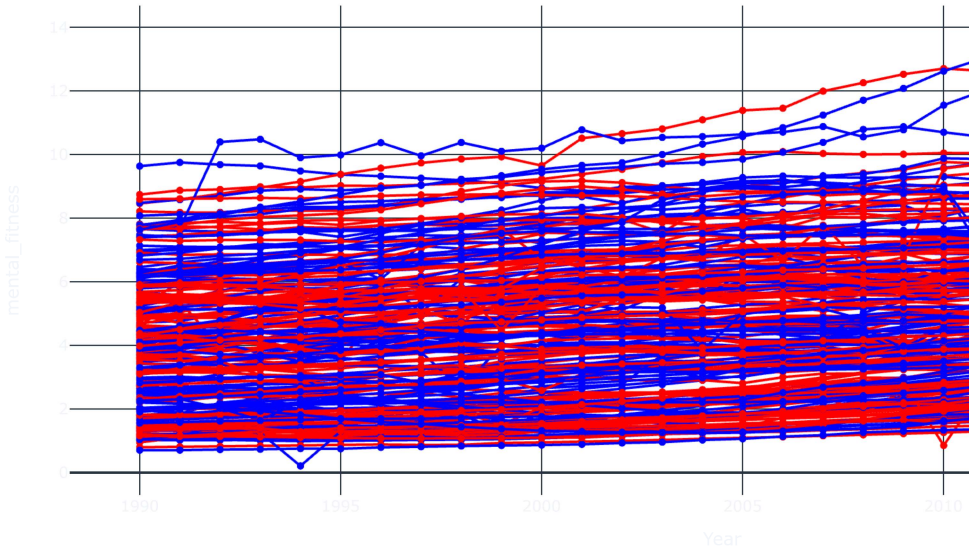



```
mean = data['mental_fitness'].mean()
# Calculating the mean value of the 'mental_fitness' column in the DataFrame data
mean
# Displaying the calculated mean value
```

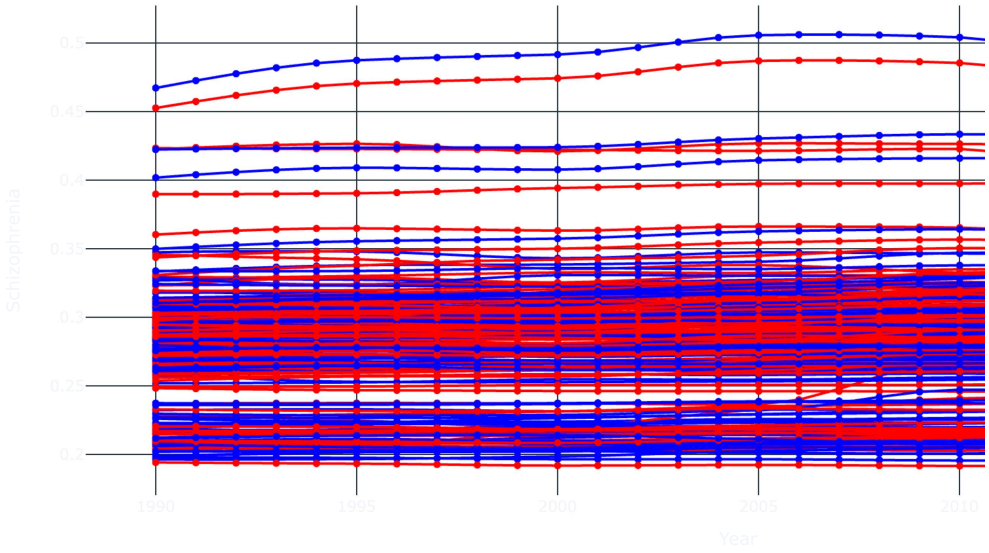
```
4.8180618117506135
```

```
fig = px.pie(data, values='mental_fitness', names='Year')
fig.show()
# Creating a pie chart to visualize the distribution of mental fitness across different years
```

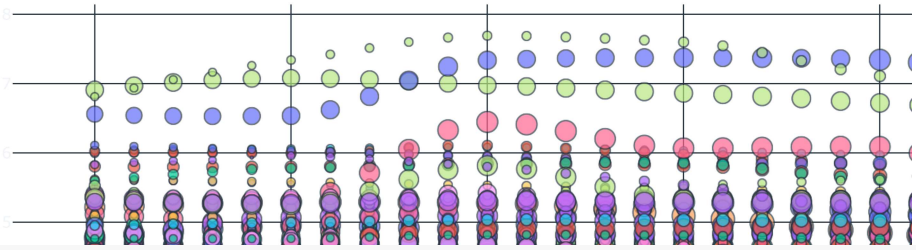

```
fig = px.line(data, x='Year', y='mental_fitness', color='country', markers=True, color_discrete_sequence=['red', 'blue'], template='plotly_dark')
fig.show()
# Creating a line plot to visualize the trend of mental fitness over time, differentiated by country

```



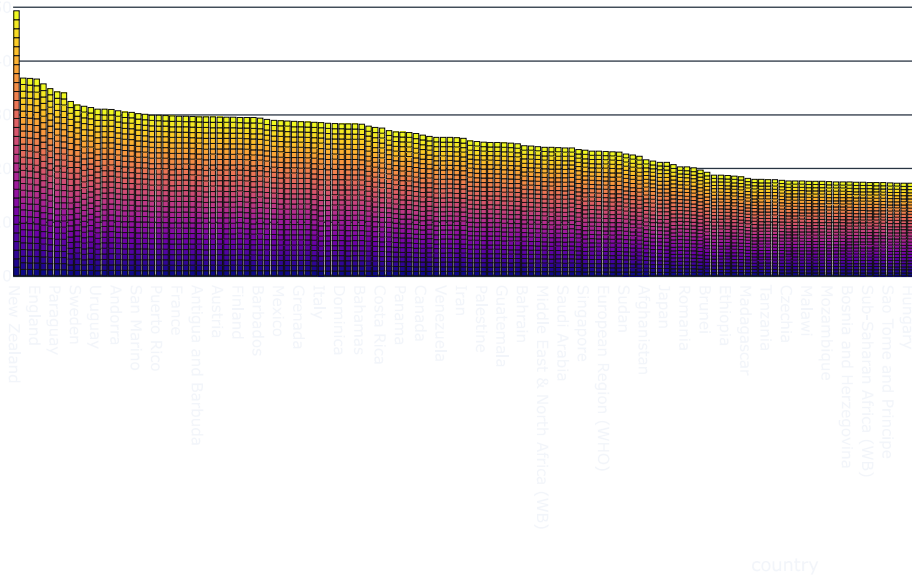

```
fig = px.line(data, x='Year', y='Schizophrenia', color='country', markers=True, color_discrete_sequence=['red', 'blue'], template='plotly_dark')
fig.show()
# Creating a line plot to visualize the trend of schizophrenia prevalence over time, differentiated by country
```



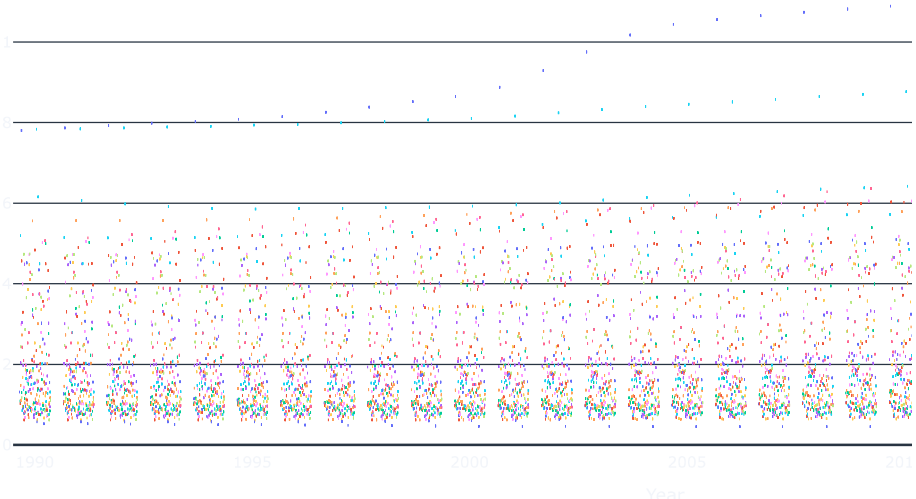

```
fig = px.scatter(data, x='Year', y='depression', color='country', size='mental_fitness', hover_data=['Schizophrenia'], template='plotly_dark')
fig.show()
# Creating a scatter plot to visualize the relationship between depression and mental fitness, differentiated by country
```



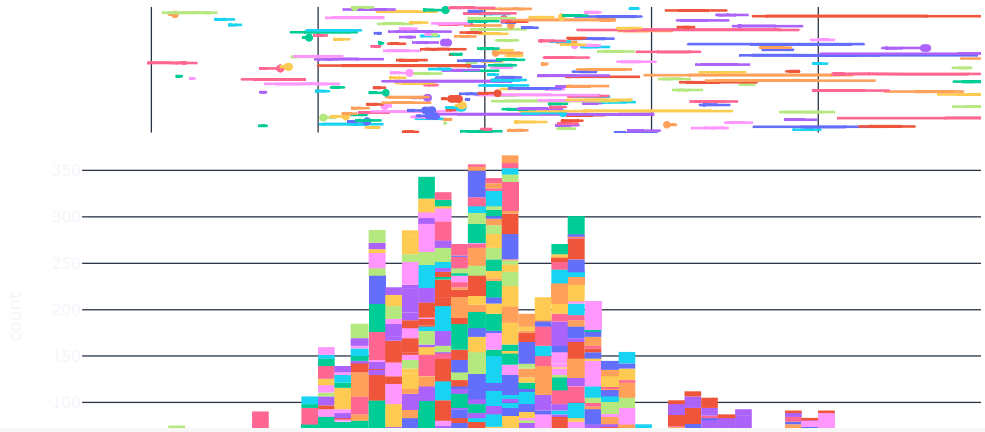
```
fig = px.bar(data, x='country', y='Bipolar_disorder', color='Year', barmode='group', template='plotly_dark')
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
# Creating a grouped bar chart to compare the prevalence of Bipolar Disorder across countries and years
```



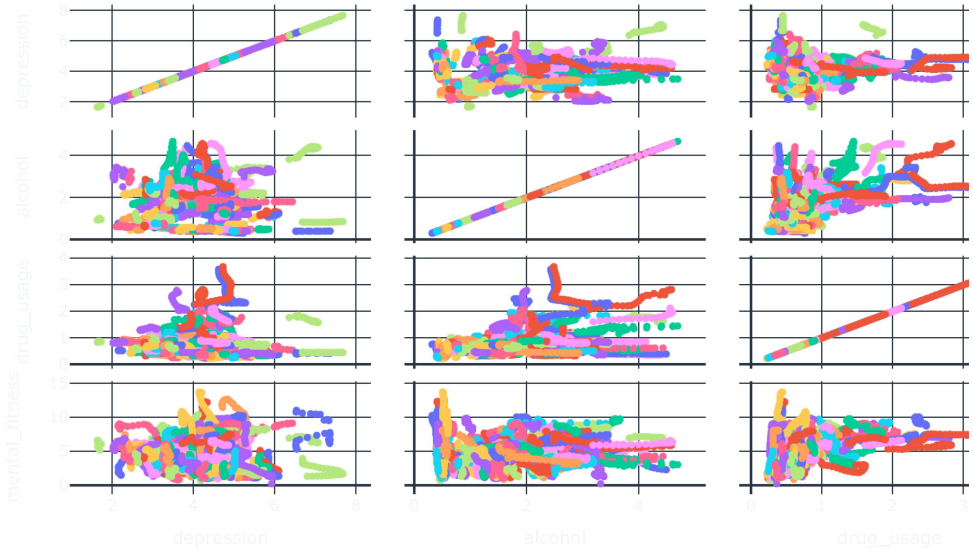

```
fig = px.box(data, x='Year', y='Eating_disorder', color='country', template='plotly_dark')
fig.show()
# Creating a box plot to visualize the distribution of Eating Disorder prevalence across different years and countries
```



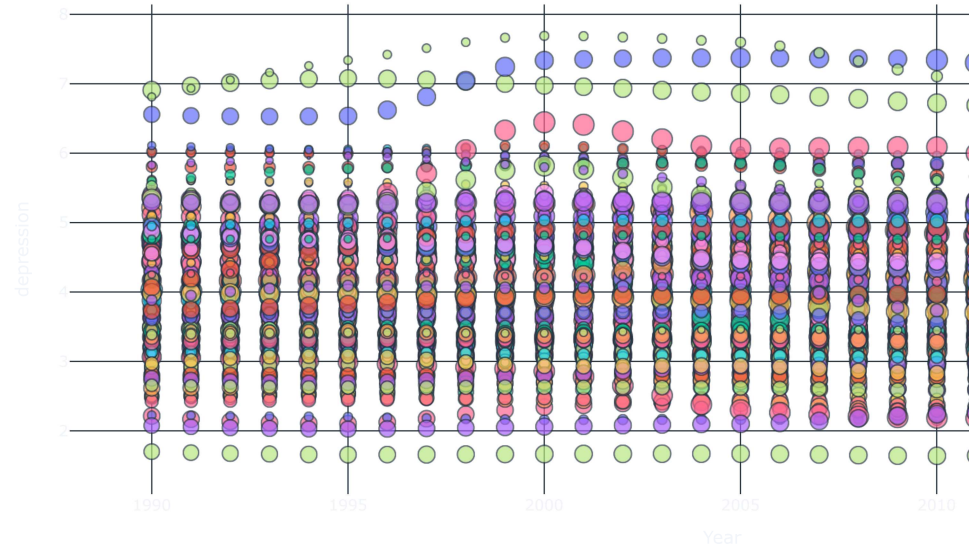

```
fig = px.histogram(data, x='Anxiety', color='country', marginal='box', template='plotly_dark')
fig.show()
# Creating a histogram with overlaid box plots to visualize the distribution of Anxiety levels across different countries
```



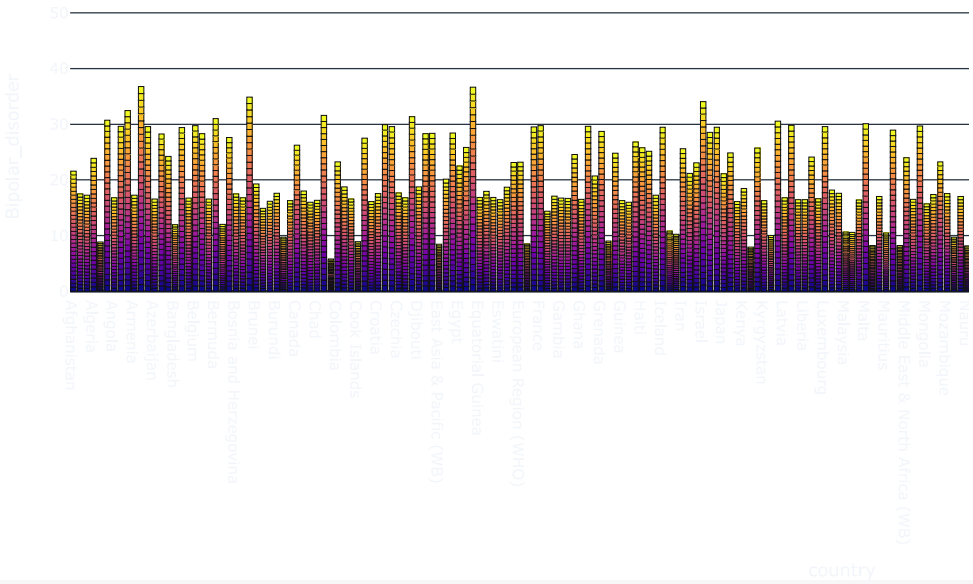

```
fig = px.scatter_matrix(data, dimensions=['depression', 'alcohol', 'drug_usage', 'mental_fitness'], color='country', template='plotly_dark')
fig.show()
# Creating a scatter matrix plot to visualize the relationships between multiple variables (depression, alcohol, drug usage, mental fitness) across different
```




```
fig = px.scatter(data, x='Year', y='depression', size='mental_fitness', color='country', template='plotly_dark')
fig.show()
# Creating a scatter plot to visualize the relationship between depression, mental fitness, and year, differentiated by country
```




```
fig = px.bar(data, x='country', y='Bipolar_disorder', color='Year', barmode='group', template='plotly_dark')
fig.show()
# Creating a grouped bar plot to compare the prevalence of Bipolar Disorder across different countries and years
```




```
df= data
df.head(10)
```

	country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	Anxiety	drug_usage	depression
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424
5	Afghanistan	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5.098495
6	Afghanistan	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5.100580
7	Afghanistan	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5.105474
8	Afghanistan	1998	0.221129	0.716139	0.100343	4.777377	0.422491	5.113707
9	Afghanistan	1999	0.220065	0.716323	0.097946	4.782067	0.421215	5.120480

```
# Display information about the DataFrame
data.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
```

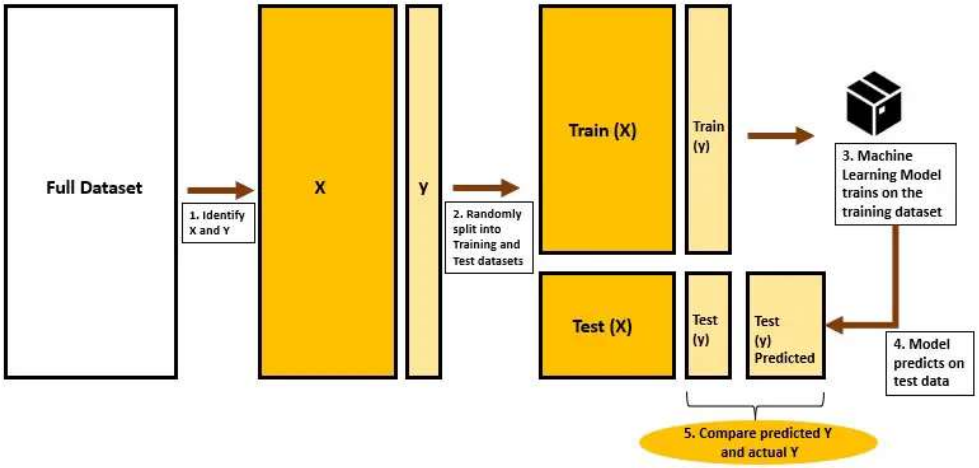
```
from sklearn.preprocessing import LabelEncoder

# Create an instance of LabelEncoder
label_encoder = LabelEncoder()

# Iterate over the columns of the DataFrame
for column in df.columns:
    # Check if the column data type is 'object' (categorical)
    if df[column].dtype == 'object':
        # Use the label encoder to transform the categorical data into numerical labels
        df[column] = label_encoder.fit_transform(df[column])

# Comment: The above code snippet utilizes the LabelEncoder from scikit-learn to encode categorical variables in the DataFrame. It iterates over each column
```

```
# Retrieve the shape of the DataFrame
df_shape = df.shape

# Print the shape of the DataFrame
print(df_shape)
```

```
(6840, 10)
```

Split Data - (6840,10)

- In this step, the data is split into two parts: training and testing datasets. This allows us to train our model on the training dataset and evaluate its accuracy on unseen test data.
- The shape of the split data is (6840, 10), indicating that the dataset contains 6,840 rows and 10 columns.




```
# Splitting the Data into Training and Testing Sets

# In this step, the dataset is split into input features (X) and target variable (y). The target variable, 'mental_fitness', is dropped from the input featur
# The target variable, 'mental_fitness', is assigned to y.
# The data is then split into training and testing sets using the train_test_split function from the sklearn.model_selection module. The test_size parameter
# The random_state parameter is set to 2 to ensure reproducibility of the split.

import sklearn.model_selection as ms

X = df.drop('mental_fitness', axis=1)
y = df['mental_fitness']

xtrain, xtest, ytrain, ytest = ms.train_test_split(X, y, test_size=0.20, random_state=2)
```

```
# Training (6840, 10)
# 6840*80/100 = 5472
# 6840*20/100 = 1368
# Training (5472, 10)
# The training data consists of 5,472 samples (80% of the total data) with 10 input features.

# Testing (1368, 10)
# The testing data consists of 1,368 samples (20% of the total data) with 10 input features.

print("xtrain:", xtrain.shape)
print("xtest:", xtest.shape)
print("\nytrain:", ytrain.shape)
print("ytest:", ytest.shape)

# Comment: The code above prints the shape of the training and testing data.
# The training data (xtrain) has a shape of (5472, 10), indicating 5,472 samples and 10 input features.
# The testing data (xtest) has a shape of (1368, 10), indicating 1,368 samples and 10 input features.
# The target variable for training (ytrain) has a shape matching the number of samples in xtrain, and the target variable for testing (ytest) has a shape mat

# Ensure that the shapes printed match the expected dimensions of your training and testing data.
```

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

ytrain: (5472,)
ytest: (1368,)
```

Key Points:

- Usally we take more and more data in training so its easy for the model to learn with more data
- Usually, it is beneficial to use a larger amount of data during model training. This allows the model to learn from a more extensive and diverse set of examples, leading to better performance and generalization.

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

# Create a Linear Regression model and fit it to the training data
lr = LinearRegression()
lr.fit(xtrain, ytrain)

# Make predictions on the training data
ytrain_pred = lr.predict(xtrain)

```

```
# Calculate Mean Squared Error (MSE) and R-squared score for training data
mse = mean_squared_error(ytrain, ytrain_pred)
r2 = r2_score(ytrain, ytrain_pred)

# Print the performance metrics of the Linear Regression model on the training set
print("Linear Regression model performance on the training set")
print("-------------------------------------------------------")
print("MSE: {}".format(mse))
print("RMSE: {}".format(np.sqrt(mse)))  # Corrected the variable name to 'mse'
print("R2 Score: {}".format(r2))


 #Float format wit 2 Decimals:
#print("MSE: {:.2f}".format(mse))
#print("RMSE: {:.2f}".format(np.sqrt(mse)))  # Corrected the variable name to 'mse'
#print("R2 Score: {:.2f}".format(r2))
```

```
Linear Regression model performance on the training set
-------------------------------------------------------
MSE: 1.389959372405798
RMSE: 1.1789653821914357
R2 Score: 0.7413245790025275
```

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

# Create a Random Forest Regressor model and fit it to the training data
rf = RandomForestRegressor()
rf.fit(xtrain, ytrain)

# Make predictions on the training data
ytrain_pred = rf.predict(xtrain)

# Calculate Mean Squared Error (MSE), RMSE, and R-squared score for training data
mse = mean_squared_error(ytrain, ytrain_pred)
rmse = np.sqrt(mse)
r2 = r2_score(ytrain, ytrain_pred)

# Print the performance metrics of the Random Forest Regressor model on the training set
print("Random Forest Regressor model performance on the training set")
print("-------------------------------------------------------------")
print("MSE: {}".format(mse))
print("RMSE: {}".format(rmse))
print("R2 Score: {}".format(r2))
```

```
Random Forest Regressor model performance on the training set
-------------------------------------------------------------
MSE: 0.0048916602143839775
RMSE: 0.06994040473420195
R2 Score: 0.999089648021048
```

Evaluation:

- In this part, we will compare the scores of above two models.

```
# Compare the scores of the Linear Regression and Random Forest Regressor models

# Calculate the scores of Linear Regression model
lr_scores = lr.score(xtest, ytest)

# Calculate the scores of Random Forest Regressor model
rf_scores = rf.score(xtest, ytest)

print("Evaluation of Models")
print("---------------------")
print("Linear Regression Score: {:.2f}".format(lr_scores))
print("Random Forest Regressor Score: {:.2f}".format(rf_scores))
```

```
Evaluation of Models
---------------------
Linear Regression Score: 0.76
Random Forest Regressor Score: 0.99
```

The evaluation results showed that the Random Forest Regressor model outperformed the Linear Regression model, achieving a higher score of 0.99 compared to 0.76.

This indicates that the Random Forest Regressor model provided a better fit to the data and could make more accurate predictions regarding mental fitness.