```
from IPython.display import HTML

html_content = """
<!DOCTYPE html>
<html>

<head>
    <title>Mental Fitness Tracker Project</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
    <style>
        body {
            font-family: Verdana, sans-serif;
        }

        .project-info {
            color: black;
            display: fill;
            border-radius: 25px;
            background-color: #808080; /* Grey background color */
            font-size: 110%;
            font-family: Verdana;
            letter-spacing: 0.5px;
            padding: 20px;
            text-align: center;
            max-width: 500px;
            margin: 0 auto;
        }

        a {
            color: red; /* Hyperlink color (change to your desired color) */
            text-decoration: none; /* Optional: Remove underline from the hyperlink */
        }

        .github-button {
            display: flex;
            align-items: center;
            justify-content: center;
            margin: 10px auto; /* Adjust margin to separate the links */
            padding: 8px 12px; /* Smaller padding */
            width: 100px; /* Adjust width as needed */
            background-color: #24292e; /* GitHub color */
            color: white;
            border: none;
            border-radius: 5px;
            font-size: 16px;
            text-decoration: none;
            transition: background-color 0.2s ease-in-out;
        }

        .github-button i {
            margin-right: 5px; /* Reduce space between icon and text */
        }

        .github-button:hover {
            background-color: #1c2024; /* GitHub color on hover */
            cursor: pointer;
        }
    </style>
</head>

<body>
    <div>
        <a class="github-button" href="https://github.com/SrSurajithPranav/Mental_Fitness_Tracker_Project">
            <i class="fab fa-github"></i>GitHub
        </a>
    </div>

    <div class="project-info">
        <p style="color: black;">
            Mental Fitness Tracker Project by <a href="https://www.linkedin.com/in/surajith-pranav-234a2b221">Surajith Pranav</a>
        </p>
    </div>
</body>

</html>"""



display(HTML(html_content))
```

GitHub

Mental Fitness Tracker Project by Surajith Pranav

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        IMPORT LIBRARIES
    </p>
</div>
"""

display(HTML(html_content))
```

IMPORT
LIBRARIES

```
import warnings
warnings.filterwarnings('ignore')
#import all libraries
import pandas as pd   #data processing ,CSV I/O
import numpy as np    #linear algebra
# import matplotlib.pyplot as plt
# import seaborn as sns
# from sklearn.model_selection import train_test_split
# from sklearn.linear_model import Ridge, Lasso, ElasticNet, LinearRegression, BayesianRidge
# 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
import seaborn as sns #seaborn in python data visulization library basesd on matplotlib
import matplotlib.pyplot as plt #matplotlib is a low level graph plotting library in python that serves as a visulization utility
import plotly.express as px #allows you to create interactive plots with very little code
```

```
#prevalence-by-mental-and-substance-use-disorder.csv
df1 = pd.read_csv('prevalence-by-mental-and-substance-use-disorder.csv')
#mental-and-substance-use-as-share-of-disease.csv
df2 = pd.read_csv('mental-and-substance-use-as-share-of-disease.csv')
```

```
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	1.70

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        MERGING DATASETS
    </p>
</div>
"""

display(HTML(html_content))
```

MERGING
DATASETS

```
#merging two datasets prevalence-by-mental-and-substance-use-disorder.csv &mental-and-substance-use-as-share-of-disease.csv
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)	(Disab Ad Life - disor Sex: Ag (Per
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	
5	Afghanistan	AFG	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5.098495	0.445422	
6	Afghanistan	AFG	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5.100580	0.444837	
7	Afghanistan	AFG	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5.105474	0.443938	
8	Afghanistan	AFG	1998	0.221130	0.716139	0.100343	4.777377	0.422401	5.113707	0.443665	

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        DATA CLEANING
    </p>
</div>
"""

display(HTML(html_content))
```

DATA
CLEANING

```
#filling missing values in dataset
data.isnull().sum()
#drop the column
data.drop('Code', axis=1, inplace=True)
#view the data
data.head(10)
#size =row*column ,shape=tuple of array dimensions(row,col)
data.size,data.shape
#column set
data.set_axis(['Country','Year','Schizophrenia', 'Bipolar_disorder', 'Eating_disorder','Anxiety','drug_usage','depression','alcohol','mer
data.head(10) #our target or dependent if mental_fitness
```

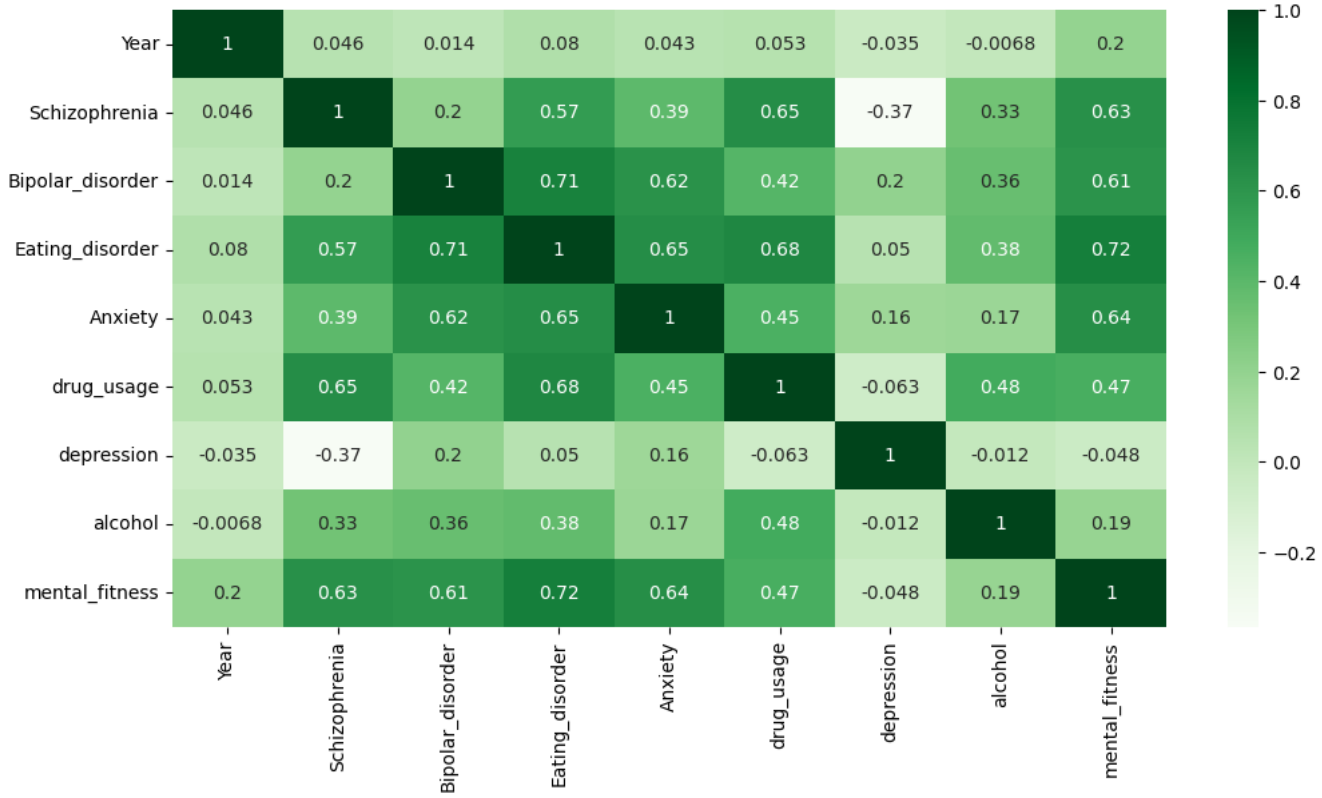
	Country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	Anxiety	drug_usage	depression	alcohol	mental_fitness
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.454202	5.125291	0.444036	1.70
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.447112	5.116306	0.444250	1.73
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.441190	5.106558	0.445501	1.79
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.435581	5.100328	0.445958	1.78
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.431822	5.099424	0.445779	1.71
5	Afghanistan	1995	0.224713	0.716686	0.111129	4.780851	0.428578	5.098495	0.445422	1.74
6	Afghanistan	1996	0.223690	0.716388	0.107786	4.777272	0.426393	5.100580	0.444837	1.78
7	Afghanistan	1997	0.222424	0.716143	0.103931	4.775242	0.423720	5.105474	0.443938	1.78
8	Afghanistan	1998	0.221129	0.716139	0.100343	4.777377	0.422491	5.113707	0.442665	1.73

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        DATA VISUALIZATION
    </p>
</div>
"""

display(HTML(html_content))
```

DATA
VISUALIZATION

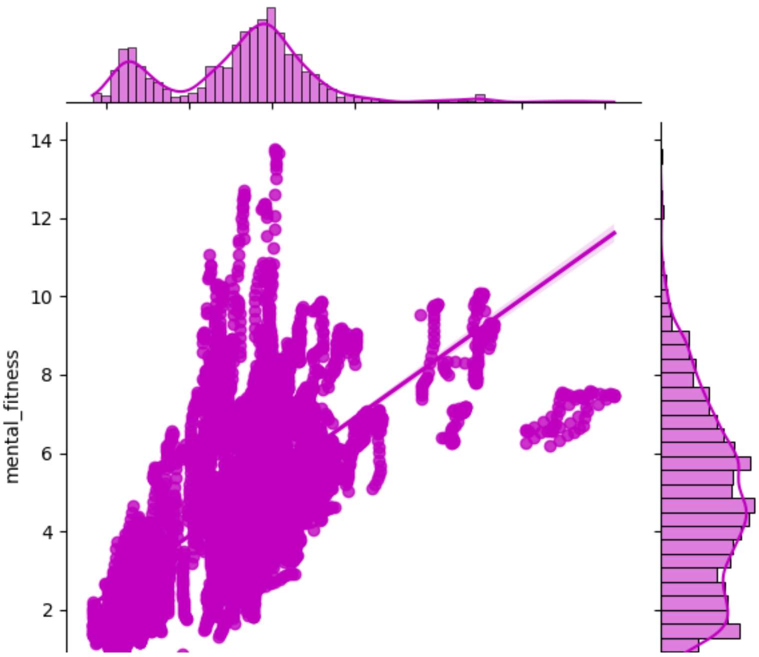
```
plt.figure(figsize=(12,6))
sns.heatmap(data.corr(),annot=True,cmap='Greens')  #heatmap is defined as graphical representation of data using colors for visual repres
plt.plot()
```

```
[]
```

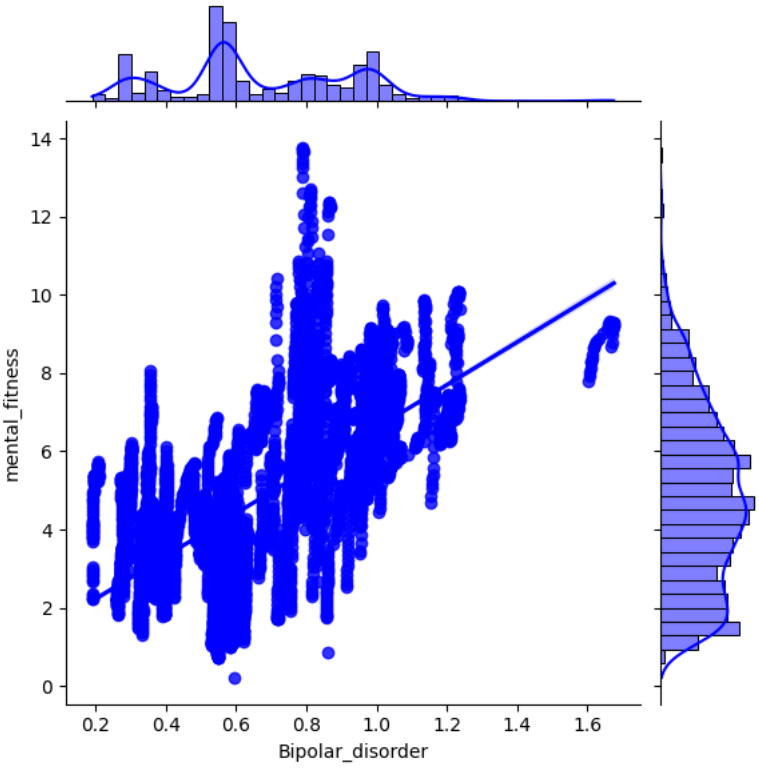


	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	Anxiety	drug_usage	depression	alcohol	mental_fitness
Year	1	0.046	0.014	0.08	0.043	0.053	-0.035	-0.0068	0.2
Schizophrenia	0.046	1	0.2	0.57	0.39	0.65	-0.37	0.33	0.63
Bipolar_disorder	0.014	0.2	1	0.71	0.62	0.42	0.2	0.36	0.61
Eating_disorder	0.08	0.57	0.71	1	0.65	0.68	0.05	0.38	0.72
Anxiety	0.043	0.39	0.62	0.65	1	0.45	0.16	0.17	0.64
drug_usage	0.053	0.65	0.42	0.68	0.45	1	-0.063	0.48	0.47
depression	-0.035	-0.37	0.2	0.05	0.16	-0.063	1	-0.012	-0.048
alcohol	-0.0068	0.33	0.36	0.38	0.17	0.48	-0.012	1	0.19
mental_fitness	0.2	0.63	0.61	0.72	0.64	0.47	-0.048	0.19	1

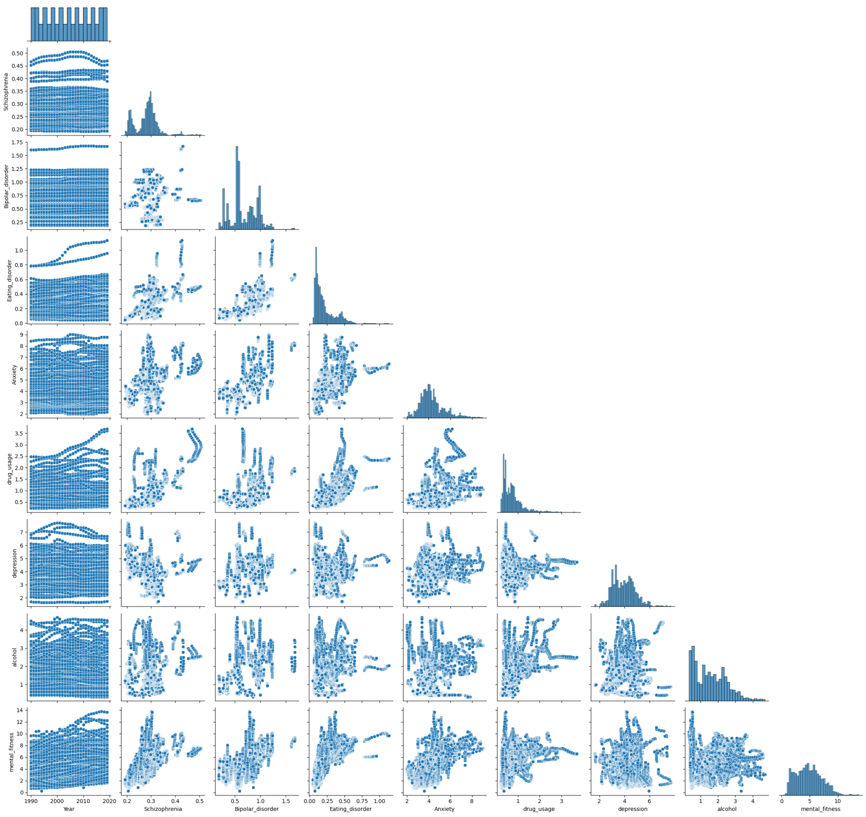
```
sns.jointplot(data,x="Schizophrenia",y="mental_fitness",kind="reg",color="m")
plt.show()
```



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



```
sns.pairplot(data,corner=True)  #paiwise relation ships in a dataset
plt.show()
```

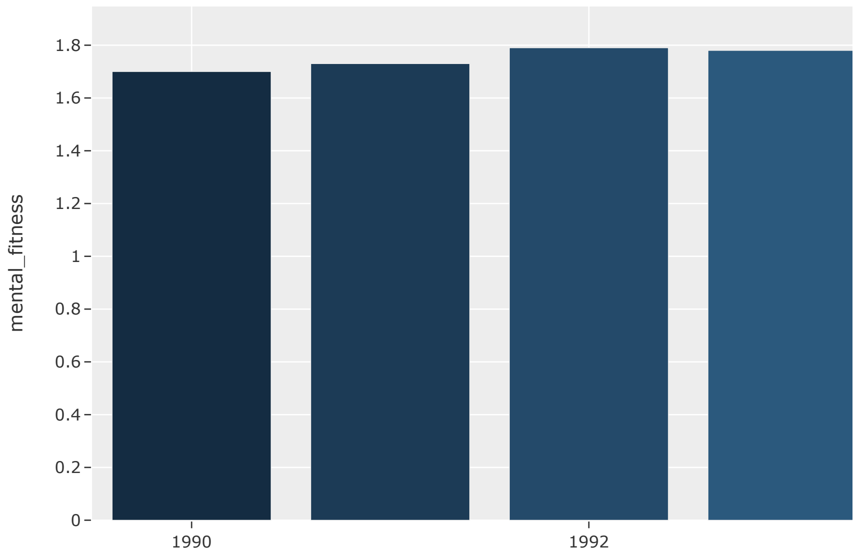


```
mean = data['mental_fitness'].mean()
mean
```

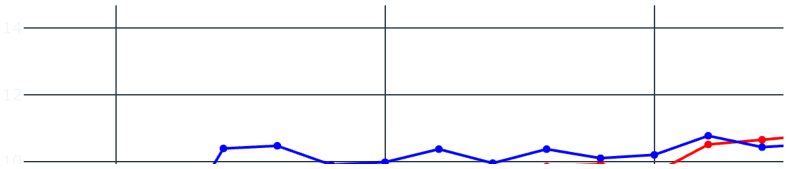
```
    4.82749926578561
```

```
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()
```



```
fig = px.line(data, x="Year", y="mental_fitness", color='Country',markers=True,color_discrete_sequence=['red','blue'],template='plotly_da
fig.show()
```



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

	Country	Year	Schizophrenia	Bipolar_disorder	Eating_disorder	Anxiety	drug_
0	Afghanistan	1990	0.228979	0.721207	0.131001	4.835127	0.4
1	Afghanistan	1991	0.228120	0.719952	0.126395	4.821765	0.4
2	Afghanistan	1992	0.227328	0.718418	0.121832	4.801434	0.4
3	Afghanistan	1993	0.226468	0.717452	0.117942	4.789363	0.4
4	Afghanistan	1994	0.225567	0.717012	0.114547	4.784923	0.4

```
#information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6810 entries, 0 to 6809
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Country           6810 non-null   object 
 1   Year              6810 non-null   int64  
 2   Schizophrenia     6810 non-null   float64
 3   Bipolar_disorder  6810 non-null   float64
 4   Eating_disorder   6810 non-null   float64
 5   Anxiety           6810 non-null   float64
 6   drug_usage        6810 non-null   float64
 7   depression        6810 non-null   float64
 8   alcohol           6810 non-null   float64
 9   mental_fitness    6810 non-null   float64
dtypes: float64(8), int64(1), object(1)
memory usage: 585.2+ KB
```

```
#transform non-numeric labels to numeric labeles
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
for i in df.columns:
    if df[i].dtype == 'object': #transform non-numerical labels (as long as they are hashable and comparable) to numeric labels
        df[i]=l.fit_transform(df[i])
```

```
df.shape
```

```
(6810, 10)
```

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        DATA TRAINING AND TESTING
    </p>
</div>
"""

display(HTML(html_content))
```

DATA
TRAINING
AND TESTING

```
X = df.drop('mental_fitness',axis=1)
y = df['mental_fitness']
from sklearn.model_selection import train_test_split   #used to split the data into training data and testing data
xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=2)
#random_state simply set seeds to the random generator,so that your train test splits are always deterministic,if you don't set seed it w
#tainning(6840,10)
#6840*80/100=5472
```

```
#6840*20/100=1368
print("xtrain: ", xtrain.shape)
print("xtest: ", xtest.shape)
print("ytrain: ", ytrain.shape)
print("ytest: ", ytest.shape)
```

```
xtrain:  (5448, 9)
xtest:  (1362, 9)
ytrain:  (5448,)
ytest:  (1362,)
```

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        LINEAR REGRESSION
    </p>
</div>
"""

display(HTML(html_content))
```

LINEAR
REGRESSION

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
lr = LinearRegression()
lr.fit(xtrain,ytrain)   #fit trainng data

# model evaluation for training set
ytrain_pred = lr.predict(xtrain)
#the mean square error is the average of the square of the difference between observed and predicted value of a variable
mse = mean_squared_error(ytrain, ytrain_pred)   #observed value and predicted value
#root mean square error measures the average difference between values predicted by model and actua values
rmse = (np.sqrt(mean_squared_error(ytrain, ytrain_pred)))
#the coefficent of determination or R2,is a measure that prioivides information about the goodness of fit of a model.In the context of reg
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")
```

```
The model performance for training set
--------------------------------------
MSE is 1.3736969568296475
RMSE is 1.1720481887830583
R2 score is 0.7440790509756825
```

```
from IPython.display import HTML

html_content = """
<div style="color:black; display: flex; justify-content: center; align-items: center; border-radius: 25px; background-color: #808080; for
    <p style="padding: 0; margin: 5px; color: black;">
        RANDOM FOREST REGRESSOR
    </p>
</div>
"""

display(HTML(html_content))
```

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")
```

```
The model performance for training set
--------------------------------------
MSE is 0.004781495370778253
RMSE is 0.06914835768677556
R2 score is 0.9991092032147548
```

```
#linear regression model evaluation for testing set
ytest_pred = lr.predict(xtest)  # (unseen data)
mse = mean_squared_error(ytest, ytest_pred)
rmse = (np.sqrt(mean_squared_error(ytest, ytest_pred)))
r2 = r2_score(ytest, ytest_pred)

print("linear regression model performance for testing set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
# random forest model evaluation for testing set
ytest_pred = rf.predict(xtest)   # (unseen data)
mse = mean_squared_error(ytest, ytest_pred)
rmse = (np.sqrt(mean_squared_error(ytest, ytest_pred)))
r2 = r2_score(ytest, ytest_pred)

print(" random forest model performance for testing set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
linear regression model performance for testing set
--------------------------------------
MSE is 1.2279896845268021
RMSE is 1.108146959805784
R2 score is 0.746121081669699
 random forest model performance for testing set
--------------------------------------
MSE is 0.028798009911894237
RMSE is 0.16969976403016662
R2 score is 0.9940461978641829
```