```
# Importing necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.impute import SimpleImputer
```

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

```
df1.describe()
```

	Year	DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)
count	6840.000000	6840.000000
mean	2004.500000	4.018362
std	9.606974	2.266026
min	1990.000000	0.210647
25%	1997.000000	3.000007
50%	2004.500000	4.978177
75%	2012.000000	5.997499
max	2019.000000	13.763317

```
df2.describe()
```

	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 - Depression - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)
count	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000	6840.000000
mean	2004.500000	0.551167	0.673891	0.131662	3.837525	0.746768	3.996499	1.578687
std	9.606974	0.047581	0.295916	0.102558	1.177661	0.463625	0.923321	0.554663
min	1990.000000	0.335251	0.189064	0.040425	1.974653	0.225471	1.946902	0.510988
25%	1997.000000	0.528449	0.539791	0.066567	3.067064	0.415910	3.258877	0.732503
50%	2004.500000	0.557496	0.593985	0.084343	4.064443	0.648092	3.964117	1.485906
75%	2012.000000	0.584760	0.697248	0.279861	4.797286	0.995013	4.592925	2.351262
max	2019.000000	0.599018	1.976204	1.128041	9.533048	3.089984	7.080213	4.089994

```
df1.head()
```

	Entity	Code	Year	DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)
0	Afghanistan	AFG	1990	3.990978
1	Afghanistan	AFG	1991	3.704361
2	Afghanistan	AFG	1992	3.791195
3	Afghanistan	AFG	1993	3.779779
4	Afghanistan	AFG	1994	3.712988

```
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)	Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Depression - Sex: Both - Age: Age-standardized (Percent)	Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)
0	Afghanistan	AFG	1990	0.335670	0.551207	0.113281	4.035257	0.452432	3.125201	0.440258
1	Afghanistan	AFG	1991	0.335125	0.719952	0.126395	4.021766	0.467112	3.122006	0.444250
2	Afghanistan	AFG	1992	0.337426	0.734049	0.121032	4.004245	0.481246	3.128559	0.449551
3	Afghanistan	AFG	1993	0.339449	0.717452	0.117942	4.000663	0.495861	3.135329	0.449590
4	Afghanistan	AFG	1994	0.339687	0.717012	0.114547	4.004653	0.491822	3.098424	0.445776

```
df1.shape
```

(6840, 4)

```
df2.shape
```

(6840, 10)

```
# Merge dataset
data = pd.merge(df1, df2, on=['Entity', 'Code', 'Year'], how='inner')
```

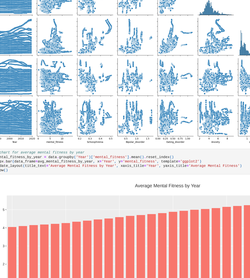
```
# Data cleaning and preprocessing
data.drop('Entity', axis=1, inplace=True)
data.rename(columns=lambda x: x.strip(), inplace=True)
data.rename(columns={'DALYs (Disability-Adjusted Life Years) - Mental disorders - Sex: Both - Age: All Ages (Percent)': 'mental_fitness',
                     'Prevalence - Schizophrenia - Sex: Both - Age: Age-standardized (Percent)': 'Schizophrenia',
                     'Prevalence - Bipolar disorder - Sex: Both - Age: Age-standardized (Percent)': 'Bipolar_disorder',
                     'Prevalence - Eating disorders - Sex: Both - Age: Age-standardized (Percent)': 'Eating_disorder',
                     'Prevalence - Anxiety disorders - Sex: Both - Age: Age-standardized (Percent)': 'Anxiety',
                     'Prevalence - Drug use disorders - Sex: Both - Age: Age-standardized (Percent)': 'drug_usage',
                     'Prevalence - Depression disorders - Sex: Both - Age: Age-standardized (Percent)': 'depression',
                     'Prevalence - Alcohol use disorders - Sex: Both - Age: Age-standardized (Percent)': 'alcohol'}, inplace=True)
```

```
# Convert non-numeric values to NaN
data = data.apply(pd.to_numeric, errors='coerce')
```

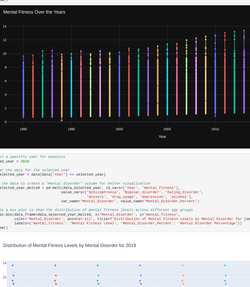
```
# Drop 'Code' column
data.drop('Code', axis=1, inplace=True)
```

```
# Impute missing values with the mean
imputer = SimpleImputer(strategy='mean')
data = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
```

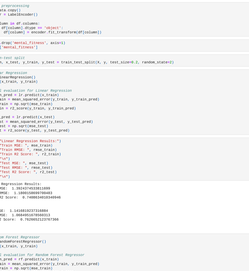
```
# Exploratory analysis
corr_matrix = data.corr()
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



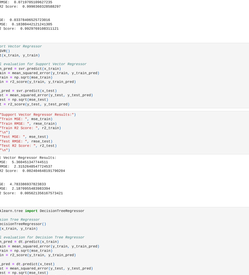

```
# Pairplot with scatterplots
sns.pairplot(data=data, corner=True)
plt.show()
```




```
# Bar chart for average mental fitness by year
avg_mental_fitness_by_year = data.groupby('Year')['mental_fitness'].mean().reset_index()
fig = px.bar(avg_mental_fitness_by_year, x='Year', y='mental_fitness', title='Average Mental Fitness')
fig.update_layout(title_text='Average Mental Fitness by Year', xaxis_title='Year', yaxis_title='Average Mental Fitness')
fig.show()
```




```
# Line chart for mental fitness over the years
fig = px.line(data_frame=data, x='Year', y='mental_fitness', color='Year', markers=True, template='plotly_dark')
fig.update_layout(title_text='Mental Fitness Over the Years', xaxis_title='Year', yaxis_title='Mental Fitness')
fig.show()
```




```
# Select a specific year for analysis
selected_year = 2019

# Filter the data for the selected year
data_selected_year = data[data['Year'] == selected_year]

# Melt the data to create a "mental disorder" column for better visualization
data_selected_year_melted = pd.melt(data_selected_year, id_vars=['Year', 'mental_fitness'],
                    value_vars=['Schizophrenia', 'Bipolar_disorder', 'Eating_disorder',
                                'Anxiety', 'drug_usage', 'depression', 'alcohol'],
                    var_name='Mental_Disorder', value_name='Mental_Fitness_Percent')

# Create a box plot to show the distribution of mental fitness levels across different mental disorders
fig = px.boxplot(data_frame=data_selected_year_melted, x='Mental_Disorder', y='mental_fitness',
                 color='Mental_Disorder', points='all', title='Distribution of Mental Fitness Levels by Mental Disorder for {selected_year}',
                 labels={'mental_fitness': 'Mental Fitness Level', 'Mental_Disorder': 'Mental Disorder', 'Mental_Fitness_Percent': 'Mental Disorder Percentage'})
fig.show()
```

```
# Data preprocessing
df = data.copy()
encoder = LabelEncoder()

for column in df.columns:
    if df[column].dtype == 'object':
        df[column] = encoder.fit_transform(df[column])

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

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

```
# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)

# Model evaluation for Linear Regression
y_train_pred = lr.predict(X_train)
mae_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mae_train)
r2_train = r2_score(y_train, y_train_pred)

y_test_pred = lr.predict(X_test)
mae_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mae_test)
r2_test = r2_score(y_test, y_test_pred)

print("Linear Regression Results:")
print("Train MAE: ", mae_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MAE: ", mae_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
print("\n")

Linear Regression Results:
Train MAE:  0.1416853371881108
Train RMSE:  1.1901200897904663
Train R2 Score:  0.7400194035949946

Test MAE:  1.3248163273338804
Test RMSE:  1.1510267192967766
Test R2 Score:  0.7020051237667566
```

```
# Random Forest Regressor
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

# Model evaluation for Random Forest Regressor
y_train_pred = rf.predict(X_train)
mae_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mae_train)
r2_train = r2_score(y_train, y_train_pred)

y_test_pred = rf.predict(X_test)
mae_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mae_test)
r2_test = r2_score(y_test, y_test_pred)

print("Random Forest Regressor Results:")
print("Train MAE: ", mae_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MAE: ", mae_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
print("\n")

Random Forest Regressor Results:
Train MAE:  0.09117075444235656
Train RMSE:  0.3019161108168928
Train R2 Score:  0.9908408005662637

Test MAE:  0.8197546026257363
Test RMSE:  0.9054266666666667
Test R2 Score:  0.7830051123767366
```

```
# Support Vector Regressor
svr = SVR()
svr.fit(X_train, y_train)

# Model evaluation for Support Vector Regressor
y_train_pred = svr.predict(X_train)
mae_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mae_train)
r2_train = r2_score(y_train, y_train_pred)

y_test_pred = svr.predict(X_test)
mae_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mae_test)
r2_test = r2_score(y_test, y_test_pred)

print("Support Vector Regressor Results:")
print("Train MAE: ", mae_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MAE: ", mae_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
print("\n")

Support Vector Regressor Results:
Train MAE:  2.3112165477107
Train RMSE:  2.3112165477107
Train R2 Score:  -0.0024004004179204

Test MAE:  2.7655803070283
Test RMSE:  2.8238516802882004
Test R2 Score:  -0.0004231945673421
```

```
# Decision Tree Regressor
dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)

# Model evaluation for Decision Tree Regressor
y_train_pred = dt.predict(X_train)
mae_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mae_train)
r2_train = r2_score(y_train, y_train_pred)

y_test_pred = dt.predict(X_test)
mae_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mae_test)
r2_test = r2_score(y_test, y_test_pred)

print("Decision Tree Regressor Results:")
print("Train MAE: ", mae_train)
print("Train RMSE: ", rmse_train)
print("Train R2 Score: ", r2_train)
print("\n")
print("Test MAE: ", mae_test)
print("Test RMSE: ", rmse_test)
print("Test R2 Score: ", r2_test)
print("\n")

Decision Tree Regressor Results:
Train MAE:  0.0
Train RMSE:  0.0
Train R2 Score:  1.0

Test MAE:  0.9663315794652294
Test RMSE:  0.9830277950362218
Test R2 Score:  0.6924843047944185
```

```
# Sample new data for prediction
new_data = pd.DataFrame(
    {'Year': [2020, 2021, 2022, 2023],
     'Schizophrenia': [10.5, 9.8, 9.7, 11.2],
     'Bipolar_disorder': [8.3, 7.5, 8.9, 8.0],
     'Eating_disorder': [3.2, 3.5, 3.8, 3.6],
     'Anxiety': [6.4, 7.5, 6.2, 6.8],
     'drug_usage': [4.7, 5.3, 4.9, 4.8],
     'depression': [12.1, 11.8, 12.5, 12.3],
     'alcohol': [5.6, 5.5, 5.9, 6.0]})

# Data preprocessing for prediction
encoder = LabelEncoder()

for column in new_data.columns:
    if new_data[column].dtype == 'object':
        new_data[column] = encoder.fit_transform(new_data[column])

# Making predictions using the trained Random Forest Regressor model
predicted_mental_fitness = rf.predict(new_data)

# Create a new DataFrame to store the predictions along with other features (if needed)
predictions_df = pd.DataFrame(data={'Year': new_data['Year'], 'Predicted_Mental_Fitness': predicted_mental_fitness})

# Display the predictions
print(predictions_df)

   Year  Predicted_Mental_Fitness
0  2020                  0.68745
1  2021                  0.68745
2  2022                  0.68745
3  2023                  0.68745
```