Depression and Mental Health Data Analysis

Aryaa S A (21z210)
Deepiga S (21z214)
Nirosha V (21z234)
Thiriksha S (21z263)
Brindha L (22z435)

19Z610 - MACHINE LEARNING LABORATORY

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

BRANCH: COMPUTER SCIENCE AND ENGINEERING

Of Anna University

April 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

TABLE OF CONTENTS:

1. PROBLEM STATEMENT:

The "Depression and Mental Health Data Analysis" project aims to leverage the RHMCD-20 dataset to gain deeper insights into the mental health status of individuals across diverse demographics. The dataset, sourced from various backgrounds including teenagers, college students, housewives, and professionals, encompasses key survey questions related to age, gender, occupation, quarantine experiences, and mental health history. The objective of this machine learning project is to analyze the dataset comprehensively, identify significant patterns, and develop a predictive model to understand the factors influencing depression and mental health challenges.

The project involves exploratory data analysis to uncover correlations between different variables and their impact on mental well-being. It will also include the development of a machine learning model that can predict the likelihood of depression based on the provided features. The model's interpretability will be crucial in understanding the key factors contributing to mental health struggles in various segments of the population.

2. DATASET DESCRIPTION:

[2]RHMCD-20 dataset, includes information from a wide range of sources, including teenagers from Bangladesh, college students, housewives, professionals from businesses and corporations, and other people.This is survey data for Depression and Mental Health Data Analysis.

I. **Age:** Represents the age of the participants, providing insight into the age distribution of the sample population.
II. **Gender:** Indicates the gender of the participants, allowing for the examination of gender-specific trends in mental health.
III. **Occupation:** Represents the participants' occupations, facilitating the analysis of mental health challenges across different professional backgrounds.
IV. **Days_Indoors:** Indicates the number of days the participant has not been out of the house, reflecting levels of social isolation and confinement.
V. **Growing_Stress:** Indicates whether the participant's stress is increasing day by day, capturing trends in stress levels over time.
VI. **Quarantine_Frustrations:** Reflects frustrations experienced during the first two weeks of quarantine, providing insights into initial reactions to quarantine measures.
VII. **Changes_Habits:** Represents major changes in eating habits and sleeping patterns, highlighting disruptions to lifestyle behaviours.

VIII. **Mental_Health_History:** Indicates a precedent of mental disorders in the participant's previous generation, exploring the role of family history in mental health.

IX. **Weight_Change:** Highlights changes in body weight during quarantine, capturing fluctuations in weight due to various factors.

X. **Mood_Swings:** Represents extreme mood changes experienced by participants, assessing the severity of mood instability.

XI. **Coping_Struggles:** Measures participants' ability to handle stress (Yes/Maybe/No), providing insights into their resilience.

XII. **Work_Interest:** Indicates changes in motivation for work (Yes/No), revealing potential mental health issues like burnout.

XIII. **Social_Weakness:** Tracks feelings of mental weakness in social interactions (Yes/No), reflecting social anxiety's impact on well-being.

3. METHODOLOGY / MODELS USED:

Random Forest Regression: Used for predicting continuous target variables. It's an ensemble learning method that builds multiple decision trees and combines their predictions to improve accuracy.

GridSearchCV: Used for hyperparameter tuning. It exhaustively searches through a specified parameter grid and performs cross-validation to find the best combination of hyperparameters for the model.

Chi-squared Test: Used for assessing the association between categorical variables. It tests whether there is a significant relationship between two categorical variables.

4. TOOLS USED:

NumPy: Utilized for numerical computations and array operations.

Pandas: Used for data manipulation and analysis, including reading/writing data, cleaning, and transforming datasets.

Matplotlib: Employed for data visualization, including plotting histograms, scatter plots, and other types of graphs.

Seaborn: Built on top of Matplotlib, Seaborn provides high-level interfaces for creating attractive statistical visualizations.

Scikit-learn (sklearn): Used for machine learning tasks such as preprocessing, model selection, evaluation, and hyperparameter tuning.

Statsmodels: Used for statistical modeling, including regression analysis, hypothesis testing, and exploring relationships between variables.

Libraries/APIs Used:

- **LabelEncoder:** From Scikit-learn, used for encoding categorical variables into numerical format.
- **Normalizer:** From Scikit-learn, used for normalizing numerical features.
- **train_test_split:** From Scikit-learn, used for splitting the dataset into training and testing sets.
- **RandomForestRegressor:** From Scikit-learn, used for building the Random Forest Regression model.
- **GridSearchCV:** From Scikit-learn, used for hyperparameter tuning using grid search.

- **confusion_matrix:** From Scikit-learn, used for evaluating the performance of classification models.
- **accuracy_score:** From Scikit-learn, used for calculating the accuracy of classification models.
- **chi2_contingency:** From SciPy, used for performing the chi-squared test for independence between categorical variables.
- **statsmodels.api:** Used for performing statistical analysis, including linear regression and hypothesis testing.

5. CHALLENGES FACED:

Data Quality and Completeness: Ensuring the dataset is free from errors, inconsistencies, and missing values is crucial to prevent biases in analysis. Quality control measures are necessary to maintain the integrity of the data and ensure accurate results.

Complexity of Mental Health: Acknowledging the multifaceted nature of mental health and the limitations of capturing it comprehensively within a dataset, considering various factors that may influence mental well-being beyond what is measured.

Predictive Model Performance: Developing accurate predictive models for depression and mental health challenges while avoiding common pitfalls such as overfitting, underfitting, and model bias.

6. CONTRIBUTION OF TEAM MEMBERS:

Roll No	Name	Contribution
21Z210	Aryaa S A	Data Preparation
21Z214	Deepiga S	Model Selection & Fitting the model
21Z234	Nirosha V	Hyperparamater Tuning
21Z263	Thiriksha S	Hypothesis Testing
22Z435	Brindha L	Mental Health Disorder Symptoms Analysis

7. ANNEXURE I: CODE

Importing the libraries:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import Normalizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score
from scipy.stats import chi2_contingency
import statsmodels.api as sm
```

Mental Health Disorder Symptoms Analysis

```
dataset = pd.read_csv('mental_health_finaldata_1.csv')
dataset.shape
dataset.describe()
```

Gender vs Moodswings

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender', hue='Mood_Swings', data=dataset)
plt.title('Gender vs Mood Swings')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```

Age vs Growing Stress

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue='Growing_Stress', data=dataset)
plt.title('Age vs Growing Stress')
plt.xlabel('Age')
plt.ylabel('Count')
```

```
plt.show()
```

Occupation vs Growing_Stress

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Occupation', hue='Growing_Stress', data=dataset)
plt.title('Occupation vs Growing Stress')
plt.xlabel('Occupation')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

Occupation vs Mood Swings

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Occupation', hue='Mood_Swings', data=dataset)
plt.title('Occupation vs Mood Swings')
plt.xlabel('Occupation')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

Mood Swings vs Growing Stress

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Mood_Swings', hue='Growing_Stress', data=dataset)
plt.title('Mood Swings vs Growing Stress')
plt.xlabel('Mood Swings')
plt.ylabel('Count')
plt.show()
```

Social Weakness vs Mood Swings

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Social_Weakness', hue='Mood_Swings', data=dataset)
plt.title('Social Weakness vs Mood Swings')
plt.xlabel('Social Weakness')
plt.ylabel('Count')
plt.show()
```

Weight Change vs Mood Swings

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Weight_Change', hue='Mood_Swings', data=dataset)
plt.title('Weight Change vs Mood Swings')
plt.xlabel('Weight Change')
plt.ylabel('Count')
plt.show()
```

Hypothesis Testing

```
crosstab = pd.crosstab(dataset["Gender"], dataset["Mood_Swings"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
crosstab = pd.crosstab(dataset["Age"], dataset["Growing_Stress"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
crosstab = pd.crosstab(dataset["Occupation"], dataset["Mood_Swings"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
```

```
crosstab = pd.crosstab(dataset["Occupation"], dataset["Growing_Stress"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
crosstab = pd.crosstab(dataset["Mood_Swings"], dataset["Growing_Stress"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
crosstab = pd.crosstab(dataset["Social_Weakness"], dataset["Mood_Swings"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
crosstab = pd.crosstab(dataset["Weight_Change"], dataset["Mood_Swings"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)
print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")
if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
```

Data Preparation:

Missing values:

```
total_rows = len(dataset)
missing_categorical = {}
missing_numerical = {}
for column in dataset.columns:
    missing_count = dataset[column].isnull().sum()
    missing_percentage = (missing_count / total_rows) * 100
    if dataset[column].dtype == 'object':
        missing_categorical[column] = missing_percentage
    else:
        missing_numerical[column] = missing_percentage
print("Missing value percentages for categorical features:")
for column, percentage in missing_categorical.items():
    print(f"{column}: {percentage:.2f}%")
print("\nMissing value percentages for numerical features:")
for column, percentage in missing_numerical.items():
    print(f"{column}: {percentage:.2f}%")
```

Categorical and Numerical Features:

```
datatype = dataset.dtypes
categorical_features = []
numerical_features = []
for column, dtype in datatype.items():
    if dtype == 'object':
        categorical_features.append(column)
    else:
        numerical_features.append(column)
print("Columns(",len(dataset.columns),"):",dataset.columns)
print("\nCategorical features(",len(categorical_features),"):", categorical_features)
print("Numerical features(",len(numerical_features),"):", numerical_features)
```

Encoding Categorical features:

Encode Age:

```
dataset["Age"].replace("30-Above", "30-100", inplace=True)
```

```
dataset[["age_lower", "age_upper"]] = dataset["Age"].str.split("-", expand=True)
dataset["age_lower"] = pd.to_numeric(dataset["age_lower"])
dataset["age_upper"] = pd.to_numeric(dataset["age_upper"])
dataset.drop(columns=["Age"], inplace=True)
print(dataset.head())
```

Encode Gender:

```
label_encoder = LabelEncoder()
dataset["Gender"] = label_encoder.fit_transform(dataset["Gender"])
print(dataset.head())
```

Encode Occupation:

```
dataset = pd.get_dummies(dataset, columns=["Occupation"], prefix="Occupation")
label_encoder = LabelEncoder()
for col in dataset.columns:
    if col.startswith("Occupation_"):
        dataset[col] = label_encoder.fit_transform(dataset[col])
print(dataset.head())
```

Encode Days_Indoor:

```
def split_days(days):
    if days == 'Go out Every day':
        return 0, 0
    elif days == 'More than 2 months':
        return 60, 60
    else:
        days_list = [int(s) for s in days.replace(' days', '').split('-') if s.isdigit()]
        return days_list[0], days_list[1]
dataset[['Indoor_Days_Lower', 'Indoor_Days_Higher']] =
dataset['Days_Indoors'].apply(split_days).apply(pd.Series)
dataset.drop(columns=["Days_Indoors"], inplace=True)
print(dataset.head())
```

Encode Mood_Swings:

```
print(dataset["Mood_Swings"].unique())
mapping = {"Low": 0, "Medium": 1, "High": 2}
dataset["Mood_Swings"] = dataset["Mood_Swings"].map(mapping)
print(dataset.head())
```

Encode Growing_Stress, Quarantine_Frustrations, Changes_Habits, Mental_Health_History, Weight_Change, Coping_Struggles, Work_Interest, Social_Weakness:

```
mapping = {"No": 0, "Maybe": 1, "Yes": 2}
for column in ['Growing_Stress', 'Quarantine_Frustrations', 'Changes_Habits',
'Mental_Health_History',
            'Weight_Change', 'Coping_Struggles', 'Work_Interest', 'Social_Weakness']:
   dataset[column] = dataset[column].map(mapping)
print(dataset.head())
```

Feature Scaling:

```
norm = Normalizer()
dependent_variable = 'Mood_Swings'
independent_variables = dataset.columns.drop(dependent_variable)
n_data = norm.fit_transform(dataset[independent_variables])
n_dataset = pd.DataFrame(n_data, columns=independent_variables)
n_dataset = pd.concat([n_dataset, dataset[dependent_variable]], axis=1)
print(n_dataset.head())
n_dataset.describe()
```

Distribution plot:

```
plt.figure(figsize=(600, 4))
for i, column in enumerate(n_dataset.columns):
   plt.subplot(1, len(n_dataset.columns), i+1)
   sns.histplot(n_dataset[column], kde=True)
   plt.title(column)
plt.tight_layout()
plt.show()
```

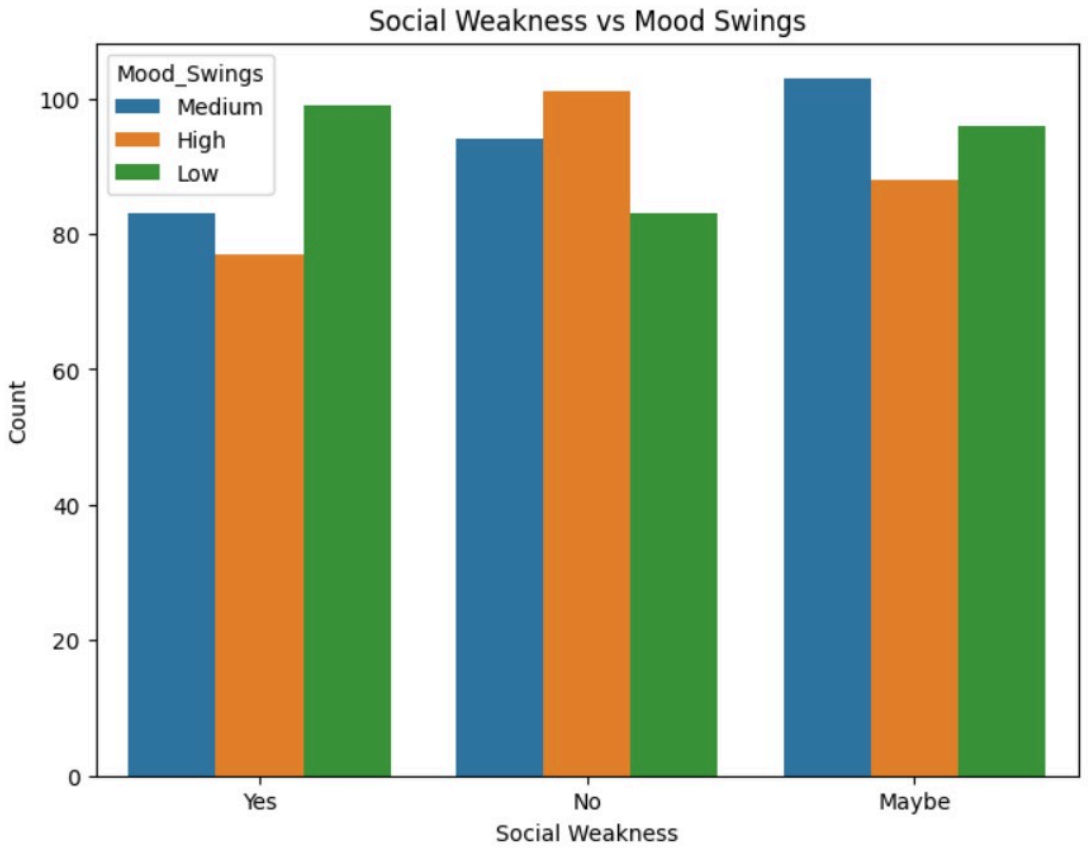
Prediction of a Symptom given others

Prediction of Mood_Swings:

```
dataset = pd.read_csv('mental_health_data_proc.csv')
X = dataset.drop(columns = ['Mood_Swings'])
y = dataset["Mood_Swings"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train_const = sm.add_constant(X_train)
model_sm = sm.OLS(y_train, X_train_const).fit()
print(model_sm.summary())
print(dataset.corr())
rf_regressor = RandomForestRegressor()
rf_regressor.fit(X_train, y_train)
y_pred = rf_regressor.predict(X_test)
rf_regressor
param_grid_rf = {
   'n_estimators': [200],
   'max_depth': [None, 10, 20],
   'min_samples_split': [2, 5, 10],
   'min_samples_leaf': [1, 2, 4]
}
rf = RandomForestRegressor()
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf,
scoring='neg_mean_squared_error', cv=5)
grid_search_rf.fit(X_train, y_train)
best_params_rf = grid_search_rf.best_params_
best_score_rf = -grid_search_rf.best_score_
print("Best Parameters for Random Forest:", best_params_rf)
print("Best Score for Random Forest:", best_score_rf)
best_model_rf = grid_search_rf.best_estimator_
y_pred_rf = best_model_rf.predict(X_test)
threshold = 1
y_pred_rf_thresholded = (y_pred_rf > threshold).astype(int)
for i in range (1,10):
  predictions = rf_regressor.predict(X_test.iloc[i:i+1, :])
  print(f"Predictions for the {i} dataframe in X_test:", predictions[0])
accuracy = -grid_search_rf.best_score_
print("Accuracy:", accuracy)
```

8. ANNEXURE II: SNAPSHOTS OF THE OUTPUT

Symptoms Analysis:



Hypothesis Testing:

```
# Hypothesis
#H0: There is no relationship between Social_Weakness and Mood_Swings
#HA: There is a significant relationship between Social_Weakness and Mood_Swings

# Statistical Test: Chi-squared test
crosstab = pd.crosstab(dataset["Social_Weakness"], dataset["Mood_Swings"])
alpha = 0.5
chi2, p, _, _ = chi2_contingency(crosstab)

print(f"Chi-squared value: {chi2}")
print(f"P-value: {p}")

if p < alpha:
    print("Reject the null hypothesis - Statistically significant relationship")
else:
    print("Fail to reject the null hypothesis - No statistically significant relationship")
```

```
Chi-squared value: 758.6548629090453
P-value: 0.425330778257634
Reject the null hypothesis - Statistically significant relationship
```

OLS Regression Results:

OLS Regression Results

Dep. Variable:	Mood_Swings	R-squared:	0.034
Model:	OLS	Adj. R-squared:	0.007
Method:	Least Squares	F-statistic:	1.255
Date:	Tue, 16 Apr 2024	Prob (F-statistic):	0.212
Time:	16:35:35	Log-Likelihood:	-778.31
No. Observations:	659	AIC:	1595.
Df Residuals:	640	BIC:	1680.
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4564	0.512	0.891	0.373	-0.549	1.462
Gender	-3.8917	3.088	-1.260	0.208	-9.955	2.171
Growing_Stress	1.0558	1.859	0.568	0.570	-2.595	4.706
Quarantine_Frustrations	-2.5559	1.836	-1.392	0.164	-6.161	1.049
Changes_Habits	-0.8918	1.836	-0.486	0.627	-4.497	2.714
Mental_Health_History	1.0571	1.873	0.564	0.573	-2.621	4.735
Weight_Change	-1.6863	1.882	-0.896	0.371	-5.382	2.010
Coping_Struggles	-1.8480	1.538	-1.202	0.230	-4.867	1.171
Work_Interest	-0.5819	1.930	-0.301	0.763	-4.372	3.209
Social_Weakness	-4.0328	1.962	-2.056	0.040	-7.885	-0.181
age_lower	1.1054	0.588	1.880	0.061	-0.049	2.260
age_upper	0.3542	0.408	0.869	0.385	-0.447	1.155
Occupation_Business	-6.4649	10.968	-0.589	0.556	-28.003	15.074
Occupation_Corporate	-5.0942	10.715	-0.475	0.635	-26.136	15.947
Occupation_Housewife	-8.3609	10.459	-0.799	0.424	-28.898	12.176
Occupation_Others	-9.0542	11.133	-0.813	0.416	-30.916	12.808
Occupation_Student	-0.7042	10.539	-0.067	0.947	-21.400	19.991
Indoor_Days_Lower	0.2533	0.428	0.592	0.554	-0.587	1.094
Indoor_Days_Higher	0.3097	0.233	1.329	0.184	-0.148	0.767

Prediction and Accuracy:

```
Best Parameters for Random Forest: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 200}
Best Score for Random Forest: 0.6828923869298488
```

```
for i in range (1,10):
  predictions = rf_regressor.predict(X_test.iloc[i:i+1, :])
  print(f"Predictions for the {i} dataframe in X_test:", predictions[0])
```

```
Predictions for the 1 dataframe in X_test: 1.0
Predictions for the 2 dataframe in X_test: 1.12
Predictions for the 3 dataframe in X_test: 0.82
Predictions for the 4 dataframe in X_test: 1.12
Predictions for the 5 dataframe in X_test: 0.78
Predictions for the 6 dataframe in X_test: 1.5
Predictions for the 7 dataframe in X_test: 1.13
Predictions for the 8 dataframe in X_test: 0.48
Predictions for the 9 dataframe in X_test: 0.94
```

```
]
accuracy = -grid_search_rf.best_score_

print("Accuracy:", accuracy)
```

```
Accuracy: 0.6828923869298488
```

9. REFERENCES:

[1]https://www.sih.gov.in/sih2023PS

[2]https://www.kaggle.com/datasets/shashwatwork/depression-and-mental-health-data-analysis

[3]https://www.kaggle.com/code/elizavetakochergina/mental-health-eda

[4]https://www.psychologytoday.com/intl/tests/health/mental-health-assessment

[5]https://towardsdatascience.com/random-forest-regression-5f605132d19d

[6]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[7]https://www.geeksforgeeks.org/random-forest-regression-in-python/

[8]https://www.geeksforgeeks.org/hyperparameter-tuning/

[9]https://towardsdatascience.com/hyperparameter-tuning-explained-d0ebb2ba1d35

[10]https://www.keboola.com/blog/random-forest-regression