Index

1. Introduction:

1.1. Project Overview

The **Mental Health Prediction** project analyzes workplace survey data to understand factors influencing mental health. It uses machine learning techniques to predict whether individuals are likely to seek treatment for mental health issues. Key features include age, gender, family history, work environment, and awareness of mental health resources. The project involves data preprocessing, exploratory analysis, and model training using classification algorithms. It helps identify trends that can guide mental health support strategies in organizations. The insights aim to promote early intervention and reduce stigma around mental health in the workplace.

1.2. Objectives

- To develop a predictive model that can identify individuals at risk of mental health issues.
- To preprocess and clean the survey data for accurate analysis.
- To evaluate different machine learning models and select the best-performing one.
- To provide insights into the factors influencing mental health treatment seeking behavior.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement

The primary problem addressed by this project is the lack of understanding of mental health issues in the workplace and the factors that influence individuals' decisions to seek treatment. The project seeks to identify these factors and predict treatment-seeking behavior.

2.2. Project Proposal (Proposed Solution)

The proposed solution involves collecting and analyzing survey data to build a machine learning model that predicts whether individuals will seek treatment for mental health issues. The model will be trained on various features derived from the survey responses.

2.3. Initial Project Planning

The project will be executed in several phases, including data collection, preprocessing, model development, optimization, and evaluation. A timeline will be established to ensure timely completion of each phase.

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

The primary data source for this project is a CSV file containing survey responses related to mental health. The data includes various demographic and workplace-related features.

3.2. Data Quality Report

The data quality was assessed, revealing some missing values and inconsistencies in categorical variables. The preprocessing steps included handling missing values, encoding categorical variables, and normalizing numerical features.

3.3. Data Exploration and Preprocessing

- Exploration: Initial exploration of the data revealed trends and patterns in mental health treatment seeking behavior.
- Preprocessing: The data was cleaned and transformed using techniques such as label encoding for categorical variables and KNN imputation for missing values.

4. Model Development Phase

4.1. Feature Selection Report

The features selected for the model include:

- Age
- Gender
- Country
- Self-employed status
- Family history of mental health issues
- Work interference
- Number of employees
- Remote work status

- Company benefits and wellness programs

4.2. Model Selection Report

Various machine learning models were evaluated, including:

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Tree
- Random Forest
- Naive Bayes
- Support Vector Machine (SVM)
- XGBoost
- AdaBoost
- Gradient Boosting

4.3. Initial Model Training Code, Model Validation and Evaluation Report

The initial model training was conducted using the following code:

```
# Load dataset
df = pd.read_csv('survey.csv')
# Preprocess data
X, y, label_encoders, scaler, le_target = preprocess_data(df)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Define models
all_models = {
  'Logistic Regression': LogisticRegression(max_iter=1000, random_state=42),
  'KNN': KNeighborsClassifier(),
  'Decision Tree': DecisionTreeClassifier(random_state=42),
  'Random Forest': RandomForestClassifier(random_state=42),
  'Naive Bayes': GaussianNB(),
  'SVM': SVC(probability=True, random_state=42),
  'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss',
```

```
random_state=42),
    'AdaBoost': AdaBoostClassifier(random_state=42),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42)
}

# Training and evaluation
for name, model in all_models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {acc:.4f}")
```

The evaluation metrics included accuracy, classification report, and confusion matrix.

5. Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation

Hyperparameter tuning was performed using GridSearchCV to optimize model parameters for better performance.

5.2. Performance Metrics Comparison Report

The performance of each model was compared based on accuracy, precision, recall, and F1-score. The best model was selected based on these metrics.

5.3. Final Model Selection Justification

The Random Forest model was selected as the final model due to its superior performance in terms of accuracy and robustness against overfitting.

6. Results

6.1. Output Screenshots

Model.py files outputs:

```
=== Training and Evaluation of All Models ===

--- Logistic Regression ---
Accuracy: 0.7559
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.69      0.73       118
           1       0.75      0.81      0.78       136

    accuracy                           0.76       254
   macro avg       0.76      0.75      0.75       254
weighted avg       0.76      0.76      0.75       254

Confusion Matrix:
 [[ 82  36]
 [ 26 110]]
```

```
--- KNN ---
Accuracy: 0.7165
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.81      0.73       118
           1       0.79      0.64      0.71       136

    accuracy                           0.72       254
   macro avg       0.73      0.72      0.72       254
weighted avg       0.73      0.72      0.72       254

Confusion Matrix:
 [[95 23]
 [49 87]]
```

```
--- Decision Tree ---
Accuracy: 0.8504
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.92      0.85       118
           1       0.92      0.79      0.85       136

    accuracy                           0.85       254
   macro avg       0.86      0.86      0.85       254
weighted avg       0.86      0.85      0.85       254

Confusion Matrix:
 [[109   9]
 [ 29 107]]
```

```
--- Random Forest ---
Accuracy: 0.9213
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.92      0.92       118
           1       0.93      0.92      0.93       136

    accuracy                           0.92       254
   macro avg       0.92      0.92      0.92       254
weighted avg       0.92      0.92      0.92       254

Confusion Matrix:
 [[109   9]
 [ 11 125]]
```

```
--- Naive Bayes ---
Accuracy: 0.7205
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.68      0.69       118
           1       0.73      0.76      0.74       136

    accuracy                           0.72       254
   macro avg       0.72      0.72      0.72       254
weighted avg       0.72      0.72      0.72       254

Confusion Matrix:
 [[ 80  38]
 [ 33 103]]
```

```
--- SVM ---
Accuracy: 0.7795
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       118
           1       0.78      0.82      0.80       136

    accuracy                           0.78       254
   macro avg       0.78      0.78      0.78       254
weighted avg       0.78      0.78      0.78       254

Confusion Matrix:
 [[ 86  32]
 [ 24 112]]
```

```
--- XGBoost ---
Accuracy: 0.8701
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.93      0.87       118
           1       0.93      0.82      0.87       136

    accuracy                           0.87       254
   macro avg       0.87      0.87      0.87       254
weighted avg       0.88      0.87      0.87       254

Confusion Matrix:
 [[110   8]
 [ 25 111]]
```

```
--- AdaBoost ---
Accuracy: 0.7480
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.67      0.71       118
           1       0.74      0.82      0.78       136

    accuracy                           0.75       254
   macro avg       0.75      0.74      0.74       254
weighted avg       0.75      0.75      0.75       254

Confusion Matrix:
 [[ 79  39]
 [ 25 111]]
```

```
--- Gradient Boosting ---
Accuracy: 0.8386
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.82      0.83       118
           1       0.85      0.85      0.85       136

    accuracy                           0.84       254
   macro avg       0.84      0.84      0.84       254
weighted avg       0.84      0.84      0.84       254

Confusion Matrix:
 [[ 97  21]
 [ 20 116]]
```

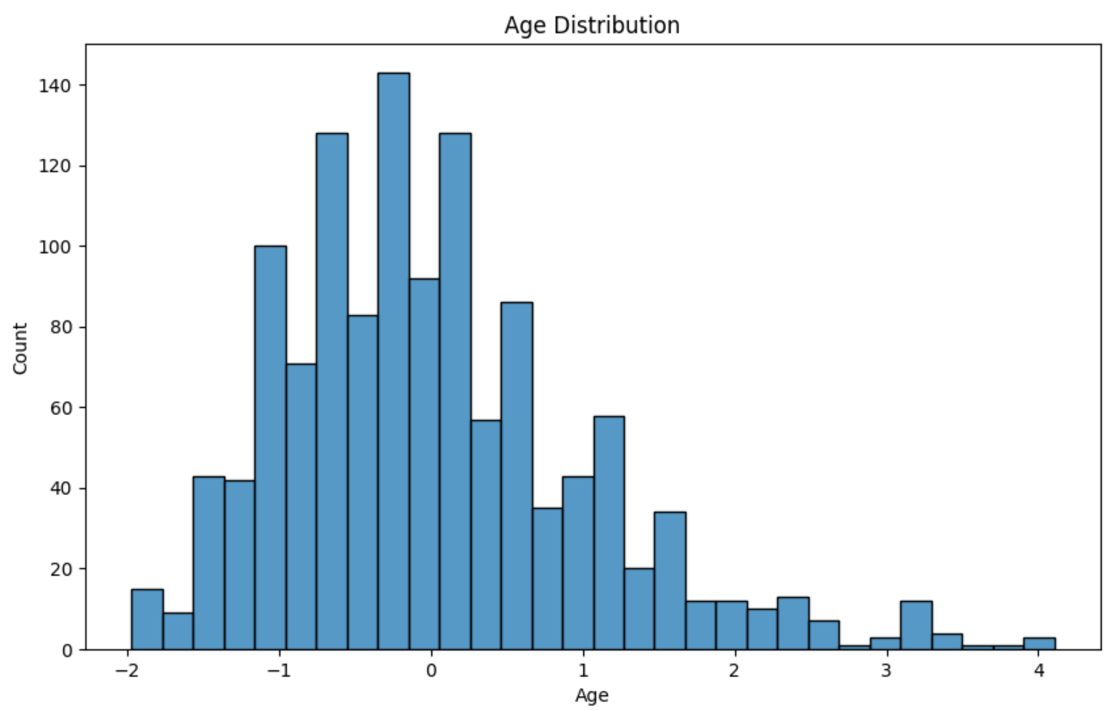
```
Best Model: Random Forest with Accuracy: 0.9213

=== Final Model Evaluation ===
Accuracy: 0.9212598425196851
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.92      0.92       118
           1       0.93      0.92      0.93       136

    accuracy                           0.92       254
   macro avg       0.92      0.92      0.92       254
weighted avg       0.92      0.92      0.92       254

Confusion Matrix:
 [[109   9]
 [ 11 125]]
```

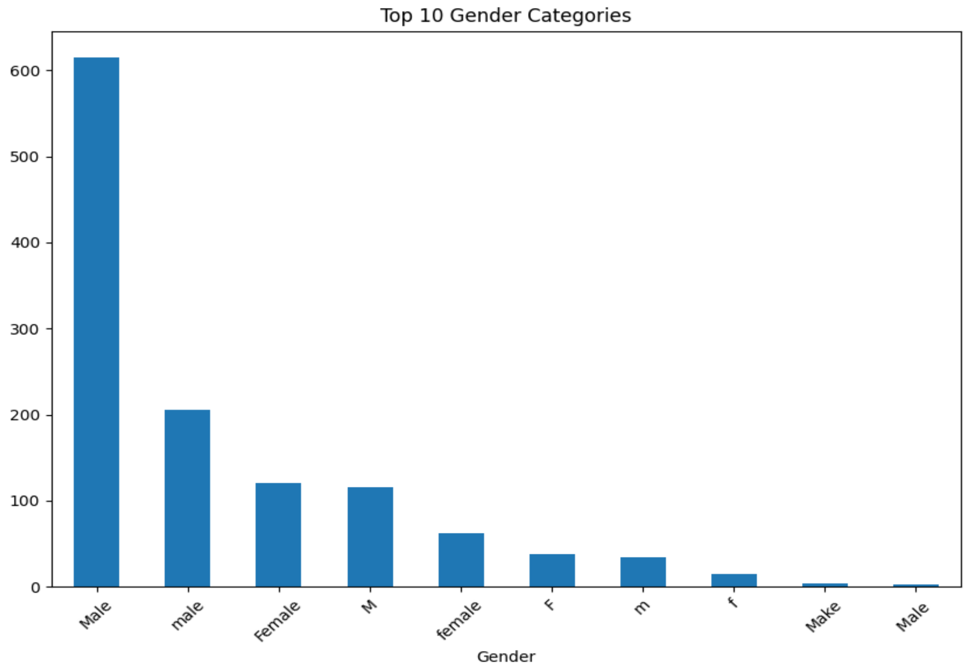
```
Feature Importances:
                      Feature  Importance
5              work_interfere    0.194589
0                         Age    0.093768
4              family_history    0.073447
10               care_options    0.062814
6                no_employees    0.055009
2                     Country    0.053817
14                      leave    0.040972
20      phys_health_interview    0.037498
9                    benefits    0.036751
17                  coworkers    0.036009
18                 supervisor    0.035799
15  mental_health_consequence    0.033810
1                      Gender    0.032775
12                  seek_help    0.030743
21         mental_vs_physical    0.030350
13                  anonymity    0.027900
11           wellness_program    0.023639
16    phys_health_consequence    0.021643
7                 remote_work    0.019259
22            obs_consequence    0.016363
8                tech_company    0.016339
19    mental_health_interview    0.016103
3               self_employed    0.010603

Saving model and preprocessing objects with pickle...
Saved successfully!
```

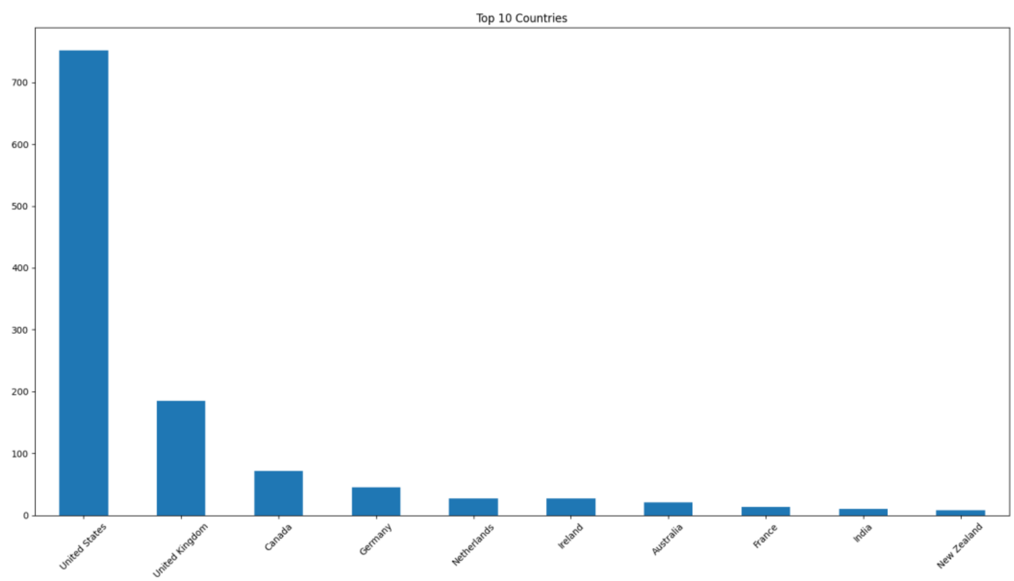
```
Example Prediction:
Prediction: 1
Probability: 0.95
```

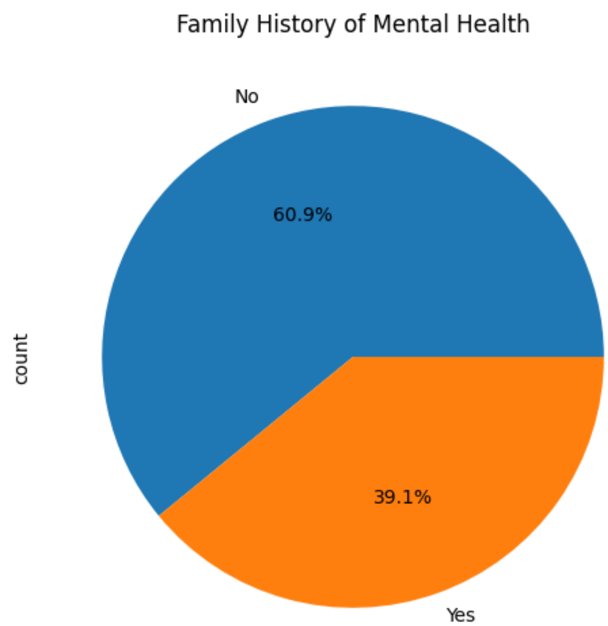
Analysis.py file outputs:

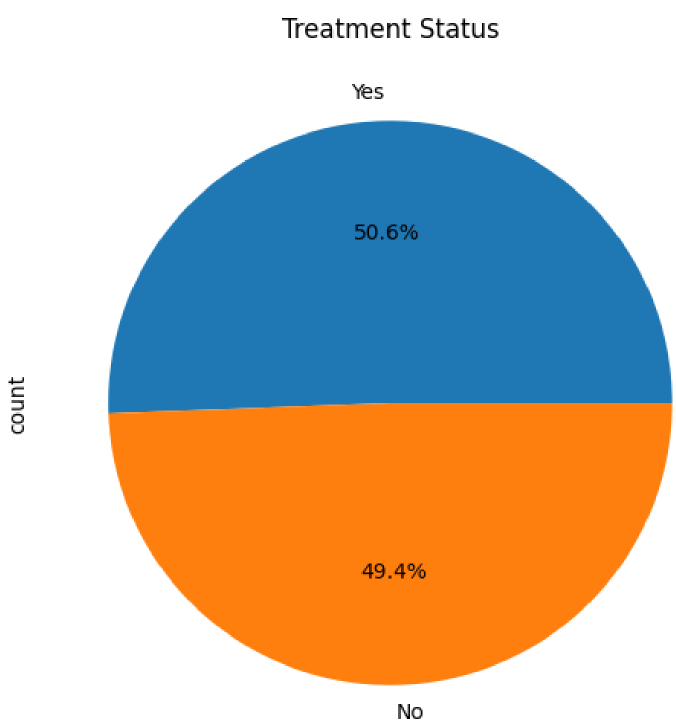
```
=== Descriptive Statistics (Preprocessed Data) ===

Basic Statistics:
               Age       Gender      Country  self_employed  ...  phys_health_interview  mental_vs_physical  obs_consequence    treatment
count  1266.000000  1266.000000  1266.000000    1266.000000  ...            1266.000000         1266.000000      1266.000000  1266.000000
mean      0.011150     0.834913     5.834123       0.122433  ...               0.698262            0.823065         0.151659     0.500000
std       1.021722     0.413695     1.922031       0.327915  ...               0.719429            0.842717         0.358832     0.500198
min      -1.971478     0.000000     0.000000       0.000000  ...               0.000000            0.000000         0.000000     0.000000
25%      -0.728285     1.000000     5.000000       0.000000  ...               0.000000            0.000000         0.000000     0.000000
50%      -0.175755     1.000000     7.000000       0.000000  ...               1.000000            1.000000         0.000000     0.500000
75%       0.514908     1.000000     7.000000       0.000000  ...               1.000000            2.000000         0.000000     1.000000
max       4.106355     2.000000     7.000000       1.000000  ...               2.000000            2.000000         1.000000     1.000000

[8 rows x 24 columns]

Categorical Variables Summary:
               Age       Gender      Country  self_employed  ...  phys_health_interview  mental_vs_physical  obs_consequence    treatment
count  1266.000000  1266.000000  1266.000000    1266.000000  ...            1266.000000         1266.000000      1266.000000  1266.000000
mean      0.011150     0.834913     5.834123       0.122433  ...               0.698262            0.823065         0.151659     0.500000
std       1.021722     0.413695     1.922031       0.327915  ...               0.719429            0.842717         0.358832     0.500198
min      -1.971478     0.000000     0.000000       0.000000  ...               0.000000            0.000000         0.000000     0.000000
25%      -0.728285     1.000000     5.000000       0.000000  ...               0.000000            0.000000         0.000000     0.000000
50%      -0.175755     1.000000     7.000000       0.000000  ...               1.000000            1.000000         0.000000     0.500000
75%       0.514908     1.000000     7.000000       0.000000  ...               1.000000            2.000000         0.000000     1.000000
max       4.106355     2.000000     7.000000       1.000000  ...               2.000000            2.000000         1.000000     1.000000

[8 rows x 24 columns]
```

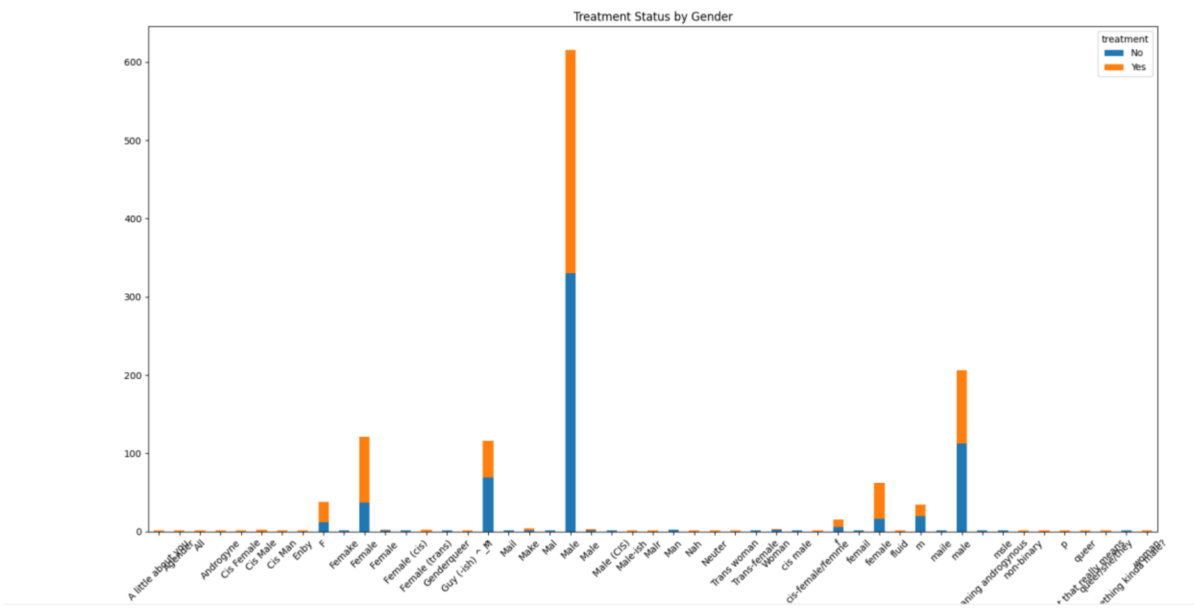


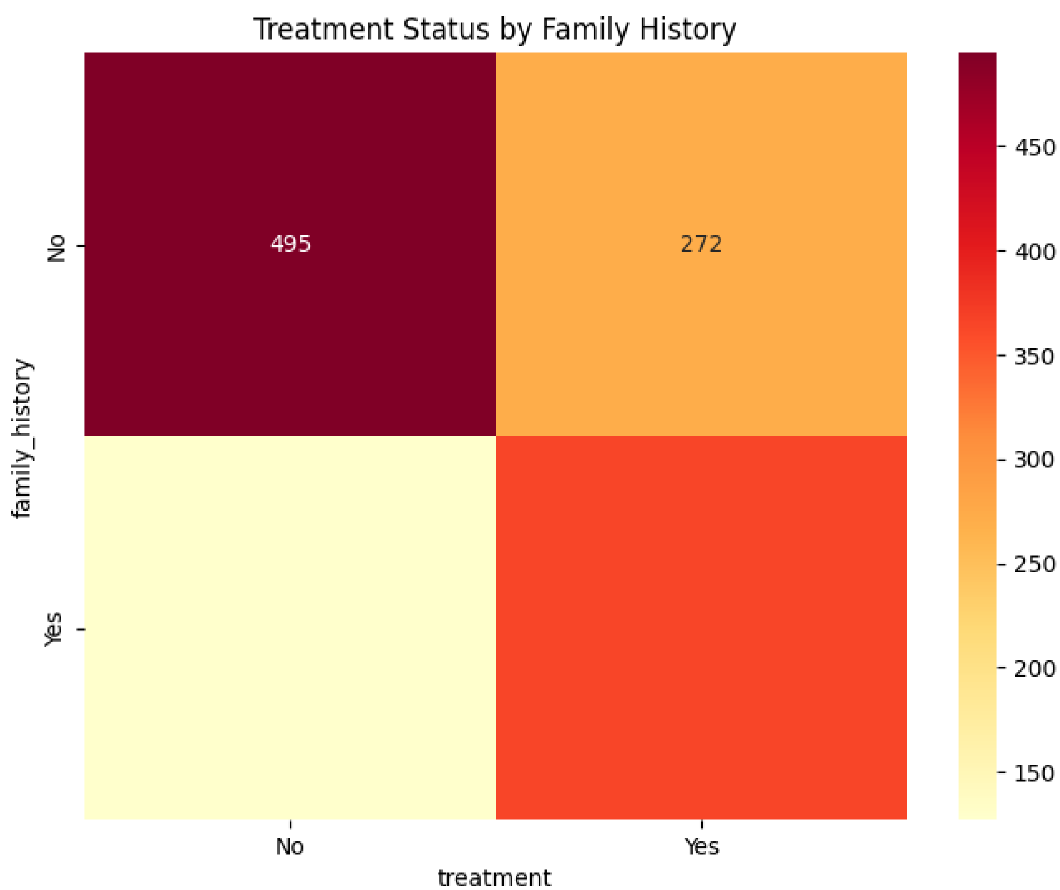

Top 10 Gender Categories
600
500
400
300
200
100
0
Male
male
Female
M
female
F
m
f
Make
Male
Gender

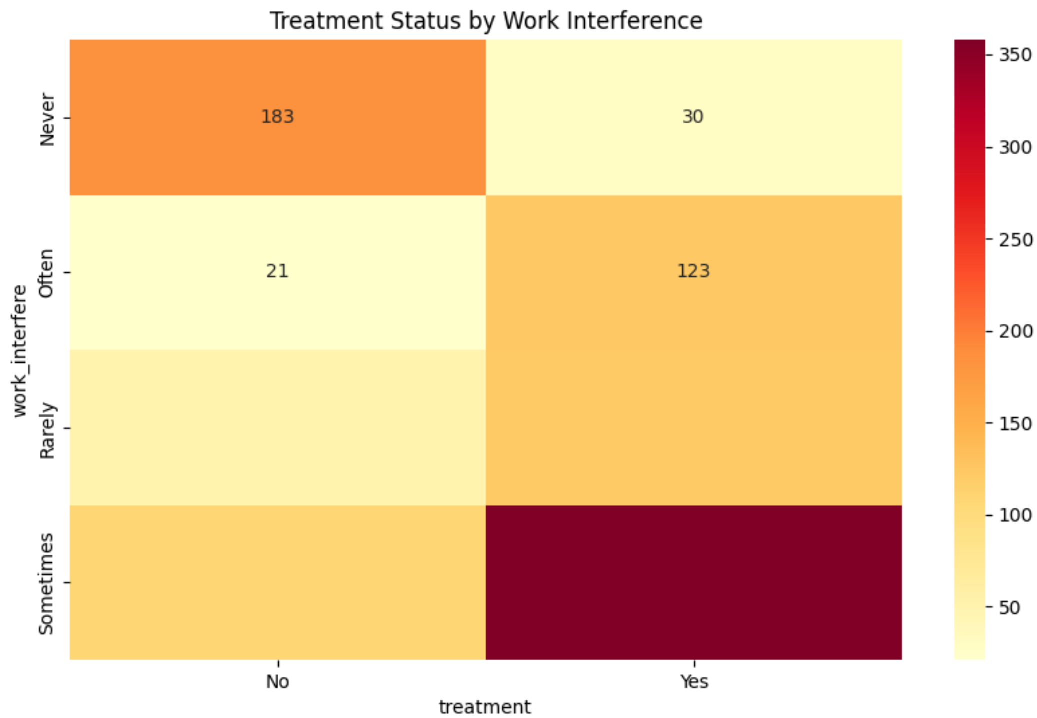

Top 10 Countries
700
600
500
400
300
200
100
0
United States
United Kingdom
Canada
Germany
Netherlands
Ireland
Australia
France
India
New Zealand

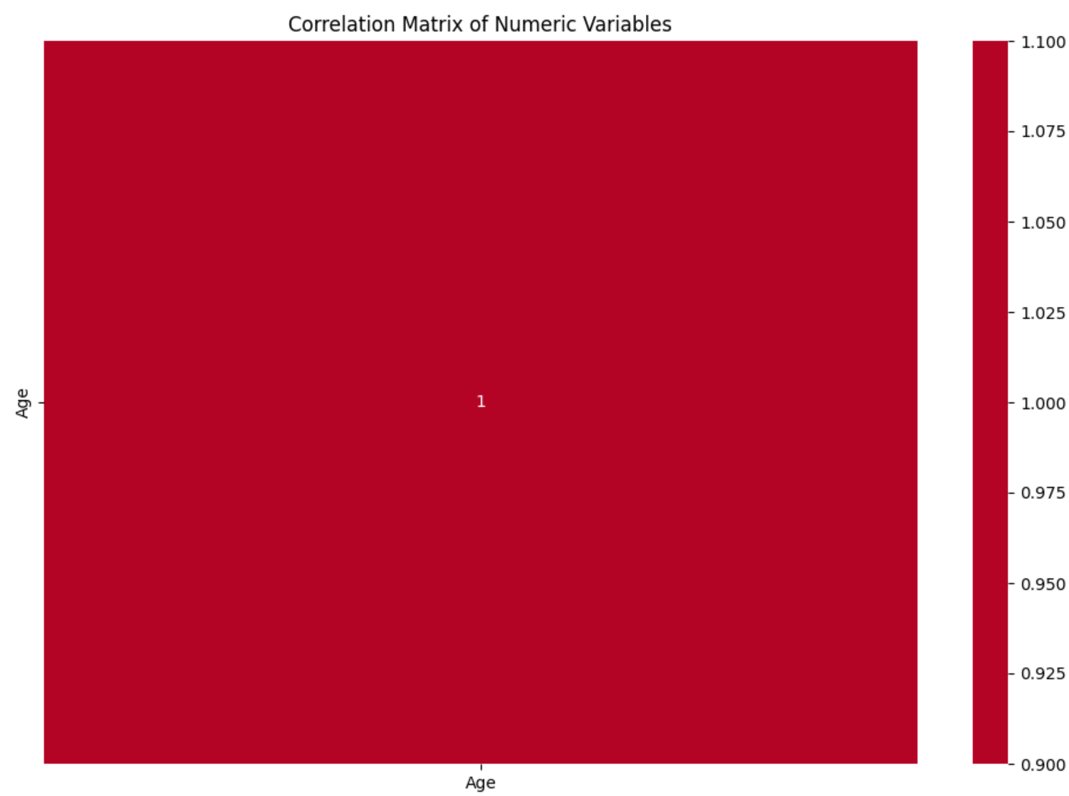

Family History of Mental Health
No
60.9%
count
39.1%
Yes

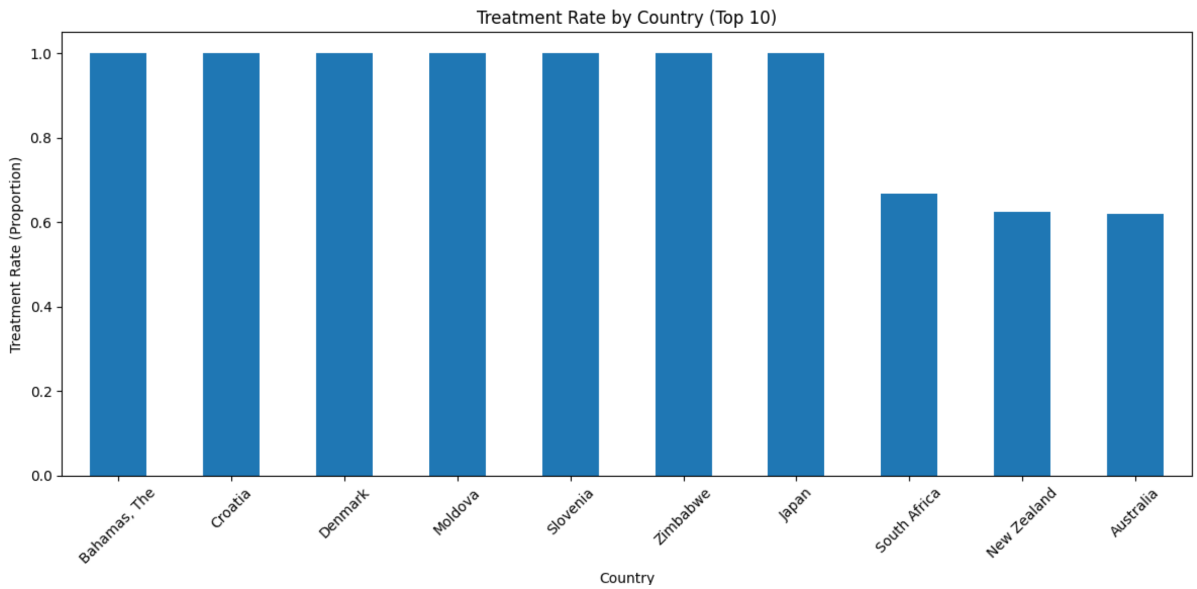

Treatment Status
Yes
50.6%
count
49.4%
No

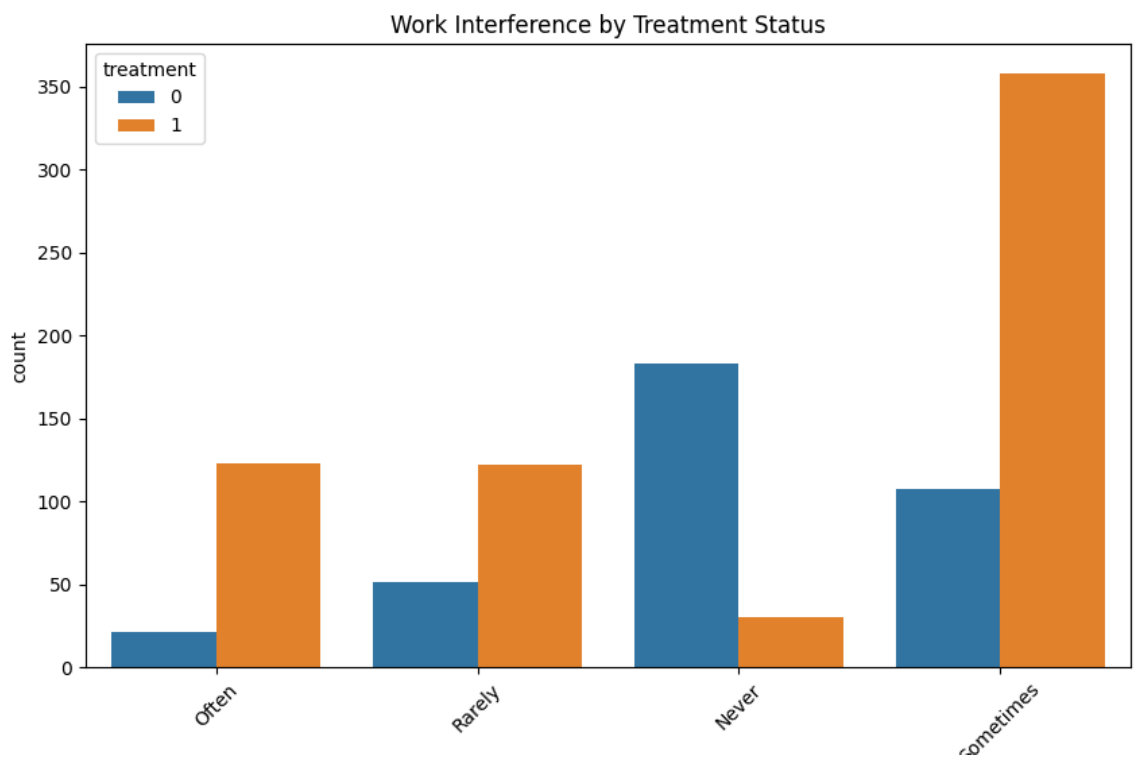

Treatment Status by Gender
treatment
No
Yes
0
100
200
300
400
500
600


Treatment Status by Family History
495
272
No
Yes
family_history
No
Yes
treatment
450
400
350
300
250
200
150

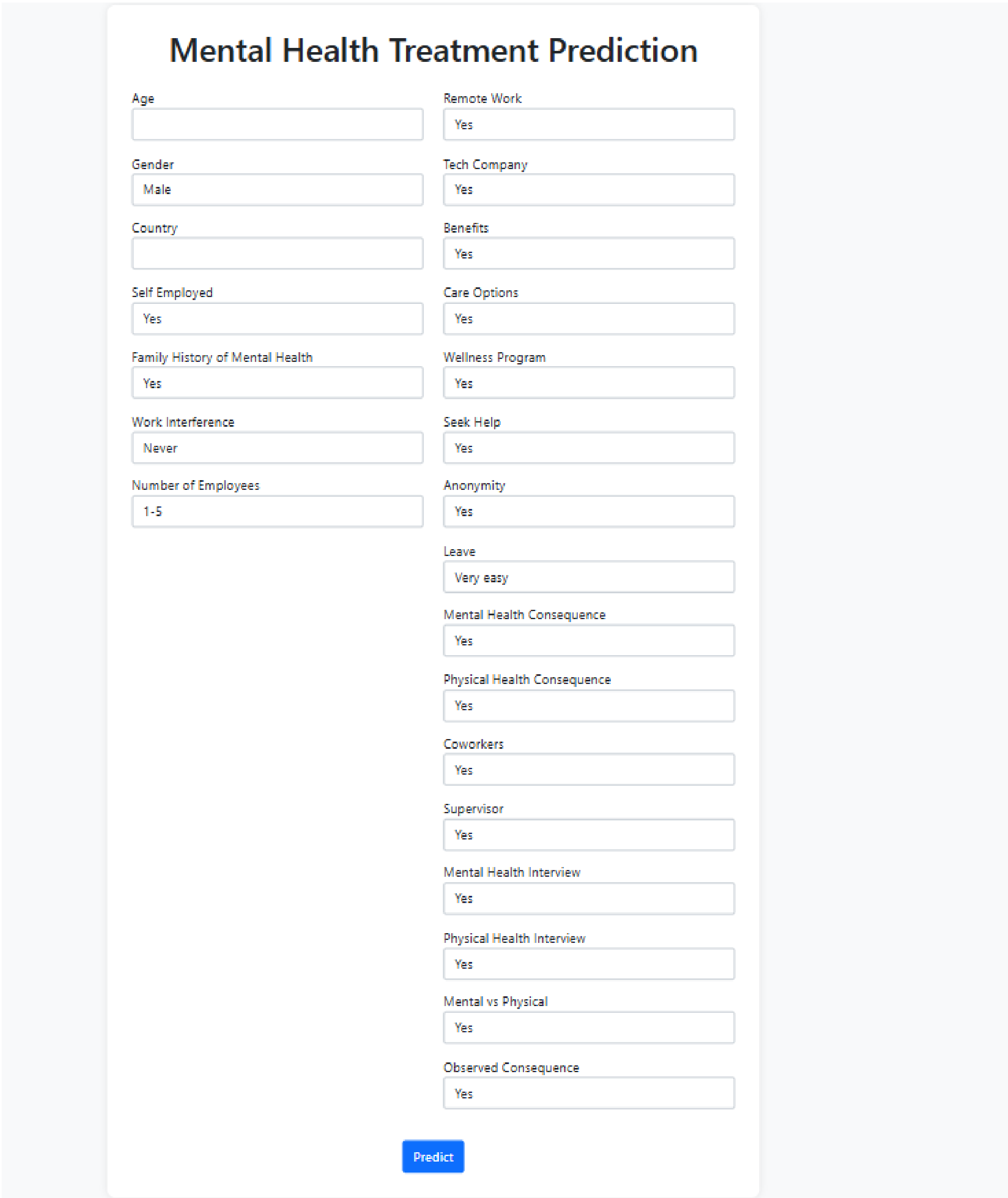

Treatment Status by Work Interference
Never
Often
Rarely
Sometimes
work_interfere
183
30
21
123
No
Yes
treatment
350
300
250
200
150
100
50

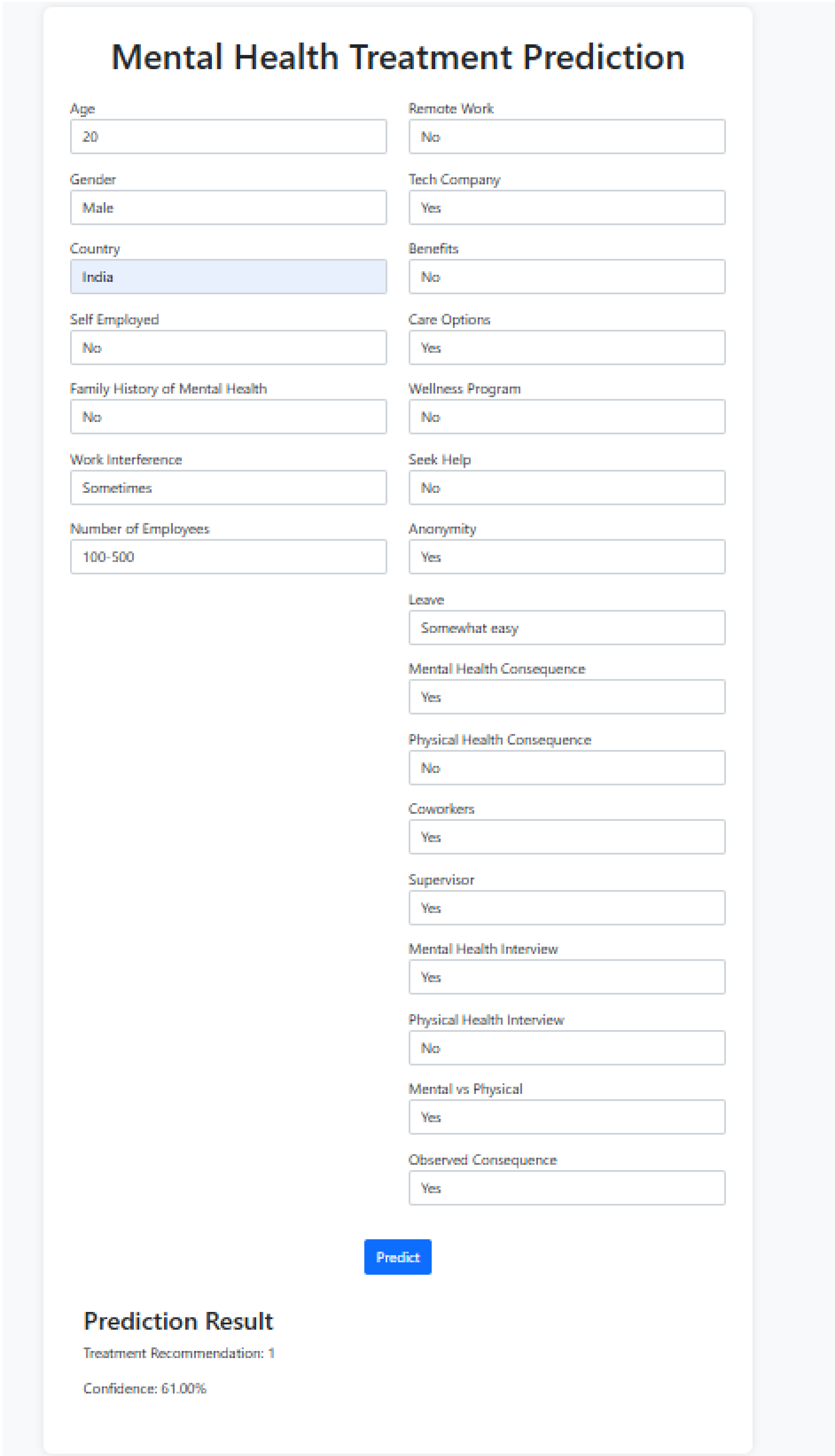

Correlation Matrix of Numeric Variables
Age
1
Age
1.100
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900


Treatment Rate by Country (Top 10)
Treatment Rate (Proportion)
1.0
0.8
0.6
0.4
0.2
0.0
Bahamas, The
Croatia
Denmark
Moldova
Slovenia
Zimbabwe
Japan
South Africa
New Zealand
Australia
Country


Work Interference by Treatment Status
treatment
0
1
count
350
300
250
200
150
100
50
0
Often
Rarely
Never

App.py file outputs:




Mental Health Treatment Prediction
Age
20
Gender
Male
Country
India
Self Employed
No
Family History of Mental Health
No
Work Interference
Sometimes
Number of Employees
100-500
Remote Work
No
Tech Company
Yes
Benefits
No
Care Options
Yes
Wellness Program
No
Seek Help
No
Anonymity
Yes
Leave
Somewhat easy
Mental Health Consequence
Yes
Physical Health Consequence
No
Coworkers
Yes
Supervisor
Yes
Mental Health Interview
Yes
Physical Health Interview
No
Mental vs Physical
Yes
Observed Consequence
Yes
Predict
Prediction Result
Treatment Recommendation: 1
Confidence: 61.00%

7. Advantages & Disadvantages

1. Advantages:

- The model provides insights into mental health treatment seeking behavior.
- It can help organizations identify at-risk employees and provide necessary support.

2. Disadvantages:

- The model's accuracy is dependent on the quality of the input data.
- There may be biases in the data that affect the model's predictions.

8. Conclusion

The Mental Health Prediction project successfully developed a predictive model to analyze mental health treatment-seeking behavior among individuals in workplace settings. By using machine learning techniques on survey data, the model identifies key factors that influence whether a person is likely to seek help for mental health issues. The insights gained from this analysis can help organizations recognize early signs of mental distress and improve their mental health support systems. This enables the creation of healthier, more supportive work environments that encourage open conversations around mental well-being.

9. Future Scope

Future work could involve expanding the dataset, incorporating additional features, and exploring more advanced machine learning techniques to enhance prediction accuracy.

10. Appendix

10.1. Source Code

Model.py:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV,
        cross_val_score

from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
        GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.impute import KNNImputer
from sklearn.utils import resample
import pickle
import warnings
warnings.filterwarnings('ignore')

# Load dataset
df = pd.read_csv('survey.csv')


def clean_gender(gender):
    gender = str(gender).strip().lower()
    if gender in ['male', 'm', 'male-ish', 'maile', 'mal', 'cis male', 'man', 'msle', 'mail',
        'make', 'malr', 'cis man']:
        return 'Male'
    elif gender in ['female', 'f', 'cis female', 'woman', 'femake', 'female (cis)', 'femail',
        'cis-female/femme', 'female ', 'femail']:
        return 'Female'
    else:
        return 'Other'



def preprocess_data(df):
    df_processed = df.copy()
    df_processed = df_processed[pd.to_numeric(df_processed['Age'],
        errors='coerce').notnull()]
```

```
df_processed['Age'] = df_processed['Age'].astype(float)
median_age = df_processed[(df_processed['Age'] >= 15) & (df_processed['Age'] <= 70)]['Age'].median()
df_processed.loc[df_processed['Age'] < 15, 'Age'] = median_age
df_processed.loc[df_processed['Age'] > 70, 'Age'] = median_age
df_processed['Gender'] = df_processed['Gender'].apply(clean_gender)
country_counts = df_processed['Country'].value_counts()
rare_countries = country_counts[country_counts < 20].index
df_processed['Country'] = df_processed['Country'].apply(lambda x: 'Other' if x in rare_countries else x)


valid_family_history = ['Yes', 'No']
valid_work_interfere = ['Never', 'Rarely', 'Sometimes', 'Often']
valid_treatment = ['Yes', 'No']
df_processed = df_processed[df_processed['family_history'].isin(valid_family_history)]
df_processed = df_processed[df_processed['work_interfere'].isin(valid_work_interfere)]
df_processed = df_processed[df_processed['treatment'].isin(valid_treatment)]


features = ['Age', 'Gender', 'Country', 'self_employed', 'family_history',
            'work_interfere', 'no_employees', 'remote_work', 'tech_company',
            'benefits', 'care_options', 'wellness_program', 'seek_help',
            'anonymity', 'leave', 'mental_health_consequence',
            'phys_health_consequence', 'coworkers', 'supervisor',
            'mental_health_interview', 'phys_health_interview',
            'mental_vs_physical', 'obs_consequence']


categorical_columns = [col for col in features if df_processed[col].dtype == 'object' or col == 'Gender']
for col in categorical_columns:
    df_processed[col] = df_processed[col].fillna(df_processed[col].mode()[0])


label_encoders = {}
```

```
    for column in categorical_columns:
        le = LabelEncoder()
        df_processed[column] = le.fit_transform(df_processed[column].astype(str))
        label_encoders[column] = le

    imputer = KNNImputer(n_neighbors=5)
    df_processed[features] = imputer.fit_transform(df_processed[features])
    scaler = StandardScaler()
    df_processed['Age'] = scaler.fit_transform(df_processed[['Age']])
    df_processed['treatment'] = df_processed['treatment'].astype(str)

    df_majority = df_processed[df_processed['treatment'] ==
        df_processed['treatment'].mode()[0]]
    df_minority = df_processed[df_processed['treatment'] !=
        df_processed['treatment'].mode()[0]]
    df_minority_upsampled = resample(df_minority, replace=True,
        n_samples=len(df_majority), random_state=42)
    df_balanced = pd.concat([df_majority, df_minority_upsampled])

    X = df_balanced[features]
    le_target = LabelEncoder()
    y = le_target.fit_transform(df_balanced['treatment'])
    return X, y, label_encoders, scaler, le_target

# Prepare data
X, y, label_encoders, scaler, le_target = preprocess_data(df)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Define models
all_models = {
    'Logistic Regression': LogisticRegression(max_iter=1000, random_state=42),
    'KNN': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
```

```
    'Random Forest': RandomForestClassifier(random_state=42),
    'Naive Bayes': GaussianNB(),
    'SVM': SVC(probability=True, random_state=42),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss',
        random_state=42),
    'AdaBoost': AdaBoostClassifier(random_state=42),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42)
}

best_model = None
best_score = 0
best_model_name = ''

print("\n=== Training and Evaluation of All Models ===")
for name, model in all_models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {acc:.4f}")
    if acc > best_score:
        best_score = acc
        best_model = model
        best_model_name = name

print(f"\nBest Model: {best_model_name} with Accuracy: {best_score:.4f}")
print("\n=== Final Model Evaluation ===")
y_pred = best_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

if hasattr(best_model, 'feature_importances_'):
```

```
    importances = pd.DataFrame({
        'Feature': X.columns,
        'Importance': best_model.feature_importances_
    }).sort_values(by='Importance', ascending=False)
    print("\nFeature Importances:\n", importances)

# Save using pickle
print("\nSaving model and preprocessing objects with pickle...")
with open('mental_health_model.pkl', 'wb') as f:
    pickle.dump(best_model, f)
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)
with open('label_encoders.pkl', 'wb') as f:
    pickle.dump(label_encoders, f)
print("Saved successfully!")

# Prediction function
def predict_mental_health(input_data):
    with open('mental_health_model.pkl', 'rb') as f:
        model = pickle.load(f)
    with open('scaler.pkl', 'rb') as f:
        scaler = pickle.load(f)
    with open('label_encoders.pkl', 'rb') as f:
        label_encoders = pickle.load(f)

    input_df = pd.DataFrame([input_data])
    categorical_columns = ['Gender', 'Country', 'self_employed', 'family_history',
                           'work_interfere', 'no_employees', 'remote_work', 'tech_company',
                           'benefits', 'care_options', 'wellness_program', 'seek_help',
                           'anonymity', 'leave', 'mental_health_consequence',
                           'phys_health_consequence', 'coworkers', 'supervisor',
                           'mental_health_interview', 'phys_health_interview',
```

```
                 'mental_vs_physical', 'obs_consequence']

    for column in categorical_columns:
        known_categories = label_encoders[column].classes_
        input_df[column] = input_df[column].apply(lambda x: x if x in
            known_categories else known_categories[0])
        input_df[column] = label_encoders[column].transform(input_df[column])

    input_df['Age'] = scaler.transform(input_df[['Age']])
    prediction = model.predict(input_df)
    probability = model.predict_proba(input_df)
    return {'prediction': prediction[0], 'probability': probability[0].max()}

# Example usage
if __name__ == "__main__":
    example_input = {
        'Age': 30,
        'Gender': 'Male',
        'Country': 'United States',
        'self_employed': 'No',
        'family_history': 'Yes',
        'work_interfere': 'Sometimes',
        'no_employees': '26-100',
        'remote_work': 'No',
        'tech_company': 'Yes',
        'benefits': 'Yes',
        'care_options': 'Yes',
        'wellness_program': 'Yes',
        'seek_help': 'Yes',
        'anonymity': 'Yes',
        'leave': 'Somewhat easy',
        'mental_health_consequence': 'No',
        'phys_health_consequence': 'No',
```

```python
        'coworkers': 'Yes',
        'supervisor': 'Yes',
        'mental_health_interview': 'No',
        'phys_health_interview': 'No',
        'mental_vs_physical': 'Yes',
        'obs_consequence': 'No'
    }

    result = predict_mental_health(example_input)
    print("\nExample Prediction:")
    print(f"Prediction: {result['prediction']}")
    print(f"Probability: {result['probability']:.2f}")
```

Analysis.py:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from model import preprocess_data

# Read and preprocess the dataset
raw_df = pd.read_csv('survey.csv')
df, y, label_encoders, scaler, le_target = preprocess_data(raw_df)
# For analysis, merge y back if needed:
df['treatment'] = y

# 1. Descriptive Statistics
print("\n=== Descriptive Statistics (Preprocessed Data) ===")
print("\nBasic Statistics:")
print(df.describe())

print("\nCategorical Variables Summary:")
```

```
print(df.describe(include=['object', 'int', 'float']))

# 2. Univariate Analysis
print("\n=== Univariate Analysis ===")

# Age Distribution
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', bins=30)
plt.title('Age Distribution')
plt.show()

# Gender Distribution
df = pd.read_csv("survey.csv")
plt.figure(figsize=(10, 6))
df['Gender'].value_counts().head(10).plot(kind='bar')
plt.title('Top 10 Gender Categories')
plt.xticks(rotation=45)
plt.show()

# Country Distribution
plt.figure(figsize=(12, 6))
df['Country'].value_counts().head(10).plot(kind='bar')
plt.title('Top 10 Countries')
plt.xticks(rotation=45)
plt.show()

# Family History of Mental Health
plt.figure(figsize=(8, 6))
df['family_history'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Family History of Mental Health')
plt.show()
```

```
# Treatment Status
plt.figure(figsize=(8, 6))
df['treatment'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Treatment Status')
plt.show()

# 3. Bivariate Analysis
print("\n=== Bivariate Analysis ===")

# Gender vs Treatment
plt.figure(figsize=(10, 6))
treatment_by_gender = pd.crosstab(df['Gender'], df['treatment'])
treatment_by_gender.plot(kind='bar', stacked=True)
plt.title('Treatment Status by Gender')
plt.xticks(rotation=45)
plt.show()

# Family History vs Treatment
plt.figure(figsize=(8, 6))
sns.heatmap(pd.crosstab(df['family_history'], df['treatment']), annot=True, fmt='d',
        cmap='YlOrRd')
plt.title('Treatment Status by Family History')
plt.show()

# Work Interference vs Treatment
plt.figure(figsize=(10, 6))
sns.heatmap(pd.crosstab(df['work_interfere'], df['treatment']), annot=True, fmt='d',
        cmap='YlOrRd')
plt.title('Treatment Status by Work Interference')
plt.show()

# 4. Correlation Analysis
print("\n=== Correlation Analysis ===")
```

```python
numeric_columns = df.select_dtypes(include=[np.number]).columns
correlation_matrix = df[numeric_columns].corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix of Numeric Variables')
plt.show()

# 5. Additional Insights
print("\n=== Additional Insights ===")

df = pd.read_csv("survey.csv")
df['treatment'] = df['treatment'].map({'Yes': 1, 'No': 0})
treatment_rate =
        df.groupby('Country')['treatment'].mean().sort_values(ascending=False)

plt.figure(figsize=(12, 6))
treatment_rate.head(10).plot(kind='bar')
plt.title('Treatment Rate by Country (Top 10)')
plt.xticks(rotation=45)
plt.ylabel('Treatment Rate (Proportion)')
plt.tight_layout()
plt.show()

# Work Interference by Treatment Status
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='work_interfere', hue='treatment')
plt.title('Work Interference by Treatment Status')
plt.xticks(rotation=45)
plt.show()
```

App.py:

```
from flask import Flask, render_template, request, jsonify
import pickle
import pandas as pd
from model import predict_mental_health

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get data from request
        data = request.get_json()

        # Ensure all required fields are present
        required_fields = [
            'Age', 'Gender', 'Country', 'self_employed', 'family_history',
            'work_interfere', 'no_employees', 'remote_work', 'tech_company',
            'benefits', 'care_options', 'wellness_program', 'seek_help',
            'anonymity', 'leave', 'mental_health_consequence',
            'phys_health_consequence', 'coworkers', 'supervisor',
            'mental_health_interview', 'phys_health_interview',
            'mental_vs_physical', 'obs_consequence'
        ]

        # Check if all required fields are present
        for field in required_fields:
            if field not in data:
```

```
            return jsonify({
                'success': False,
                'error': f'Missing required field: {field}'
            })

        # Convert Age to integer
        try:
            data['Age'] = int(data['Age'])
        except ValueError:
            return jsonify({
                'success': False,
                'error': 'Age must be a number'
            })

        # Make prediction
        result = predict_mental_health(data)

        return jsonify({
            'success': True,
            'prediction': str(result['prediction']),
            'probability': float(result['probability'])
        })
    except Exception as e:
        return jsonify({
            'success': False,
            'error': str(e)
        })

if __name__ == '__main__':
    app.run(debug=True)
```

10.2. GitHub & Project Demo Link

GITHUB LINKS:

1) M B ABINAYAA - https://github.com/A-Beeeeeee/Mental_Health_Prediction
2) K MANEESH RAM - https://github.com/Maneesh1605/Mental-health-Prediction
3) S KSHITIJ - https://github.com/ksh-20/Mental-Health-Prediction

DEMONSTRATION VIDEO LINK:

https://drive.google.com/drive/folders/1pzoA7yuDnCqfdpw6PZFovPz7_M4gK0zw