Q2: *Predicting Mental Illness*

Install Python libraries

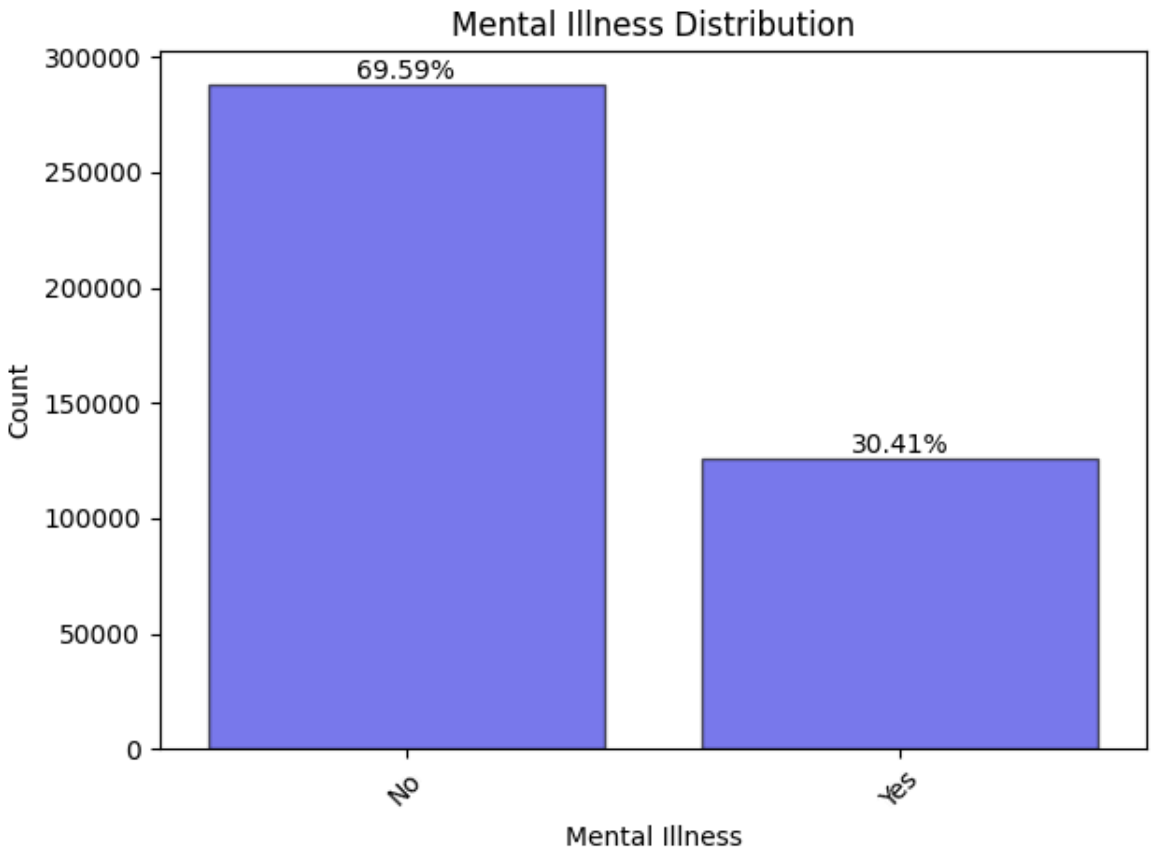
In [20]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from data.visualize import display_column_count_hist, display_features_by_count,
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)
```

Ingest Data

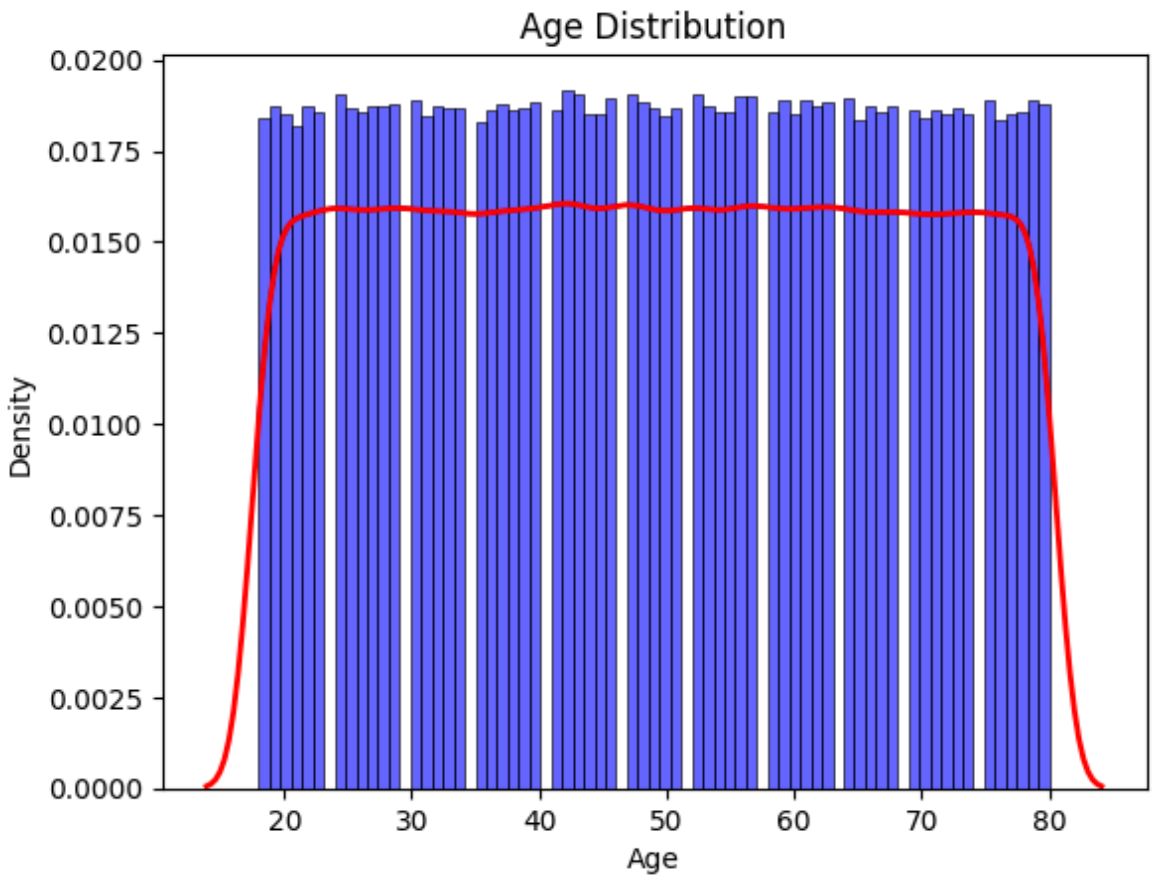
In [21]:
```
source_dataset = pd.read_csv('./../data/depression_analysis/depression_data.csv'
currated_dataset = source_dataset.copy()
```

Visualise Data

In [22]:
```
display_features_by_count(
    dataset=source_dataset,
    feature_name='History of Mental Illness',
    title='Mental Illness Distribution',
    xlabel='Mental Illness'
    )
```

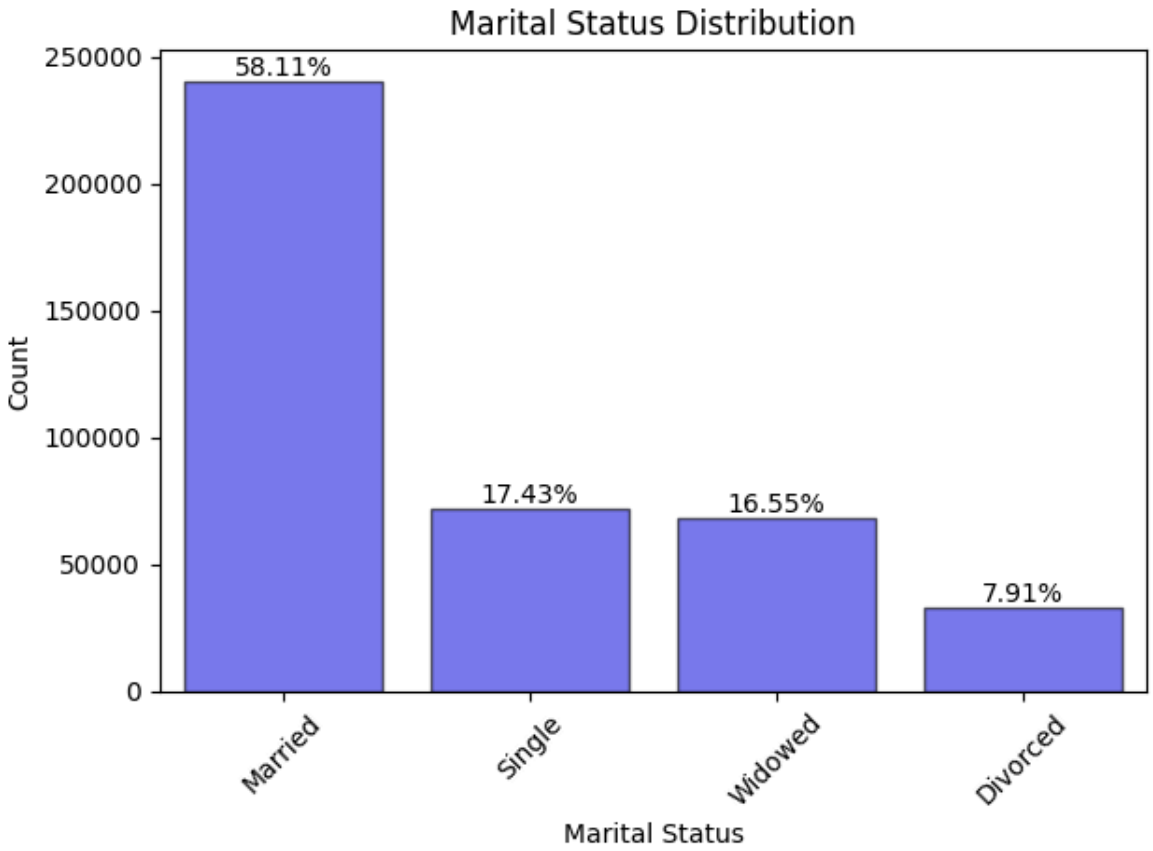


In [23]:
```
display_column_count_hist(
    source_dataset=source_dataset,
    column_name="Age",
)
```

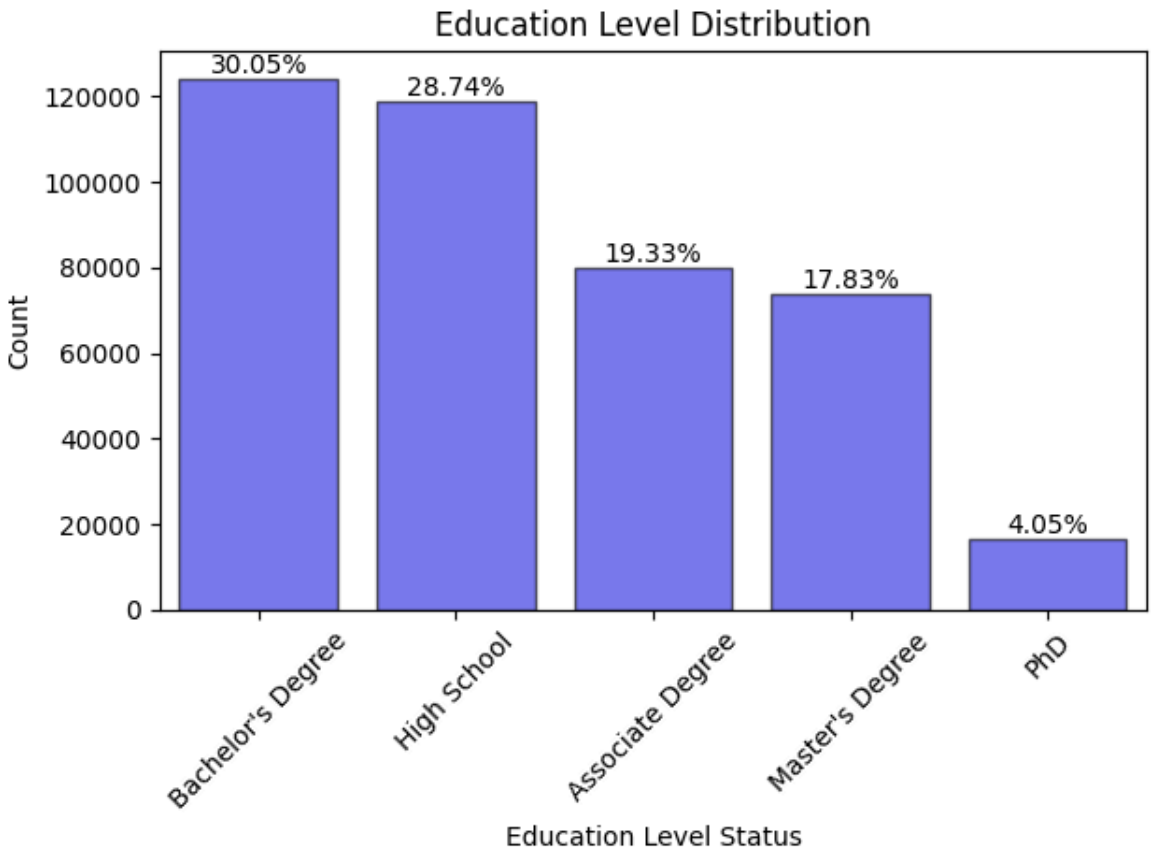


Hypothesis: Age is quite granular. Is there statistical power in whether an individual is more susceptible to depression if they are 18 or 19? Intuition would suggest not. Age ranges, for example, 18-30, may have more predictive power, as cohorts within similar age groups may have distinct characters at a generational level.

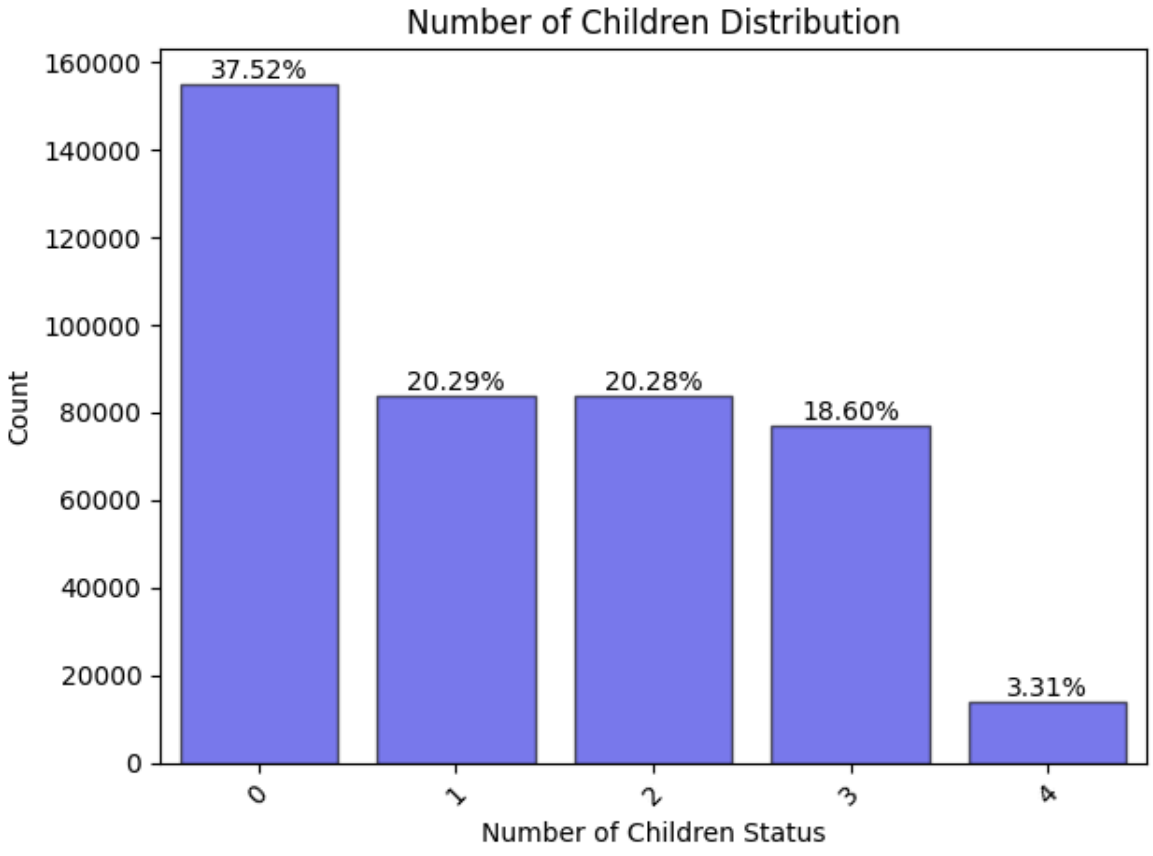
In [24]:
```
display_features_by_count(
    dataset=source_dataset,
    feature_name='Marital Status',
    title='Marital Status Distribution',
    xlabel='Marital Status'
    )
```



In [25]:
```
display_features_by_count(
    dataset=source_dataset,
    feature_name='Education Level',
    title='Education Level Distribution',
    xlabel='Education Level Status'
    )
```



In [26]:
```
display_features_by_count(
    dataset=source_dataset,
    feature_name='Number of Children',
    title='Number of Children Distribution',
    xlabel='Number of Children Status')
```



Display Summary Statistics & Dataset Feature Types

In [27]:
```
source_dataset.describe()
# Information is limited owing to multiple data columns not yet being encoded.
```

Out[27]:

	Age	Number of Children	Income
count	413768.000000	413768.000000	413768.000000
mean	49.000713	1.298972	50661.707971
std	18.158759	1.237054	40624.100565
min	18.000000	0.000000	0.410000
25%	33.000000	0.000000	21001.030000
50%	49.000000	1.000000	37520.135000
75%	65.000000	2.000000	76616.300000
max	80.000000	4.000000	209995.220000

In [28]:
```
source_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 413768 entries, 0 to 413767
Data columns (total 16 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Name                            413768 non-null  object 
 1   Age                             413768 non-null  int64  
 2   Marital Status                  413768 non-null  object 
 3   Education Level                 413768 non-null  object 
 4   Number of Children              413768 non-null  int64  
 5   Smoking Status                  413768 non-null  object 
 6   Physical Activity Level         413768 non-null  object 
 7   Employment Status               413768 non-null  object 
 8   Income                          413768 non-null  float64
 9   Alcohol Consumption             413768 non-null  object 
 10  Dietary Habits                  413768 non-null  object 
 11  Sleep Patterns                  413768 non-null  object 
 12  History of Mental Illness       413768 non-null  object 
 13  History of Substance Abuse      413768 non-null  object 
 14  Family History of Depression    413768 non-null  object 
 15  Chronic Medical Conditions      413768 non-null  object 
dtypes: float64(1), int64(2), object(13)
memory usage: 50.5+ MB
```

Data Wrangling

Data is well curated, therefore wrangling is minimal. We need to encode the Data

Finally, the "name" column, i.e. the name of the individual should have no statistical power on the incidence of mental health. It will be removed (at the training stage)

In [29]:
```
# Define encoding settings
encoding_settings = {
    "Education Level": {"PhD": 4, "Master's Degree": 3, "Bachelor's Degree": 2,
```

```
    "Marital Status": {"Married": 3, "Divorced": 2, "Widowed": 1, "Single": 0},
    "Smoking Status": {"Non-smoker": 2, "Former": 1, "Current": 0},
    "Physical Activity Level": {"Sedentary": 2, "Moderate": 1, "Active": 0},
    "Employment Status": {"Employed": 1, "Unemployed": 0},
    "Dietary Habits": {"Healthy": 2, "Unhealthy": 1, "Moderate": 0},
    "Sleep Patterns": {"Good": 2, "Fair": 1, "Poor": 0},
    "Alcohol Consumption": {"High": 2, "Moderate": 1, "Low": 0},
}

# Apply the encoding settings
currated_dataset = currated_dataset.replace({'Yes': 1, 'No': 0})
currated_dataset.replace(encoding_settings, inplace=True)
```

Feature Engineering

Previously it was suggested that the continuous nature of the "Age" feature may be too granular, and therefore loose potentially significant information on generational affect. Create age ranges.

Feature Selection: "Name will be dropped". Given the subjectivity of this process, bias can create issues. I will first train the model and then use other statistical methods to derive feature importance. This insight will inform the feature selection

In [30]:
```
#Define age groups and their corresponding labels
def age_group(age):
    if age <= 20:
        return 0
    elif 20 <= age <= 30:
        return 1
    elif 30 <= age <= 40:
        return 2
    elif 40 <= age <= 50:
        return 3
    elif 50 <= age <= 60:
        return 4
    elif 60 <= age <= 70:
        return 5
    elif 70 <= age <= 80:
        return 5

# Apply the age grouping to the dataset
currated_dataset['Age'] = currated_dataset['Age'].apply(age_group)
```

Machine Learning

Model Selection

That data is non-linear, and it's size is medium to large, therefore models such as Gradient Boost and Random Forests will be ideal. Speed is a key consideration for me, which is why I leaned towards XGBoost initially. The XGBoost model also offered slight increase in performance.

That said, Random Forests does offer distinct advantages in terms of interpretability. Given that explainability is central to the core problem statement, i.e. what drives mental health

illness, I decided to lean towards this model instead.

In [31]:
```
currated_dataset = currated_dataset.drop(columns=['Name'])
X = currated_dataset.drop(columns=['History of Mental Illness'])
y = currated_dataset['History of Mental Illness']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

# # Initialize and train the RandomForestClassifier
# model = XGBClassifier(
#     objective='binary:logistic',
#     eval_metric='logloss',
#     use_label_encoder=False,
#     random_state=42,

# )

model = RandomForestClassifier(random_state=42)

model.fit(
    X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# # Calculate the accuracy of the model

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.64
              precision    recall  f1-score   support

           0       0.70      0.84      0.76     86319
           1       0.34      0.19      0.24     37812

    accuracy                           0.64    124131
   macro avg       0.52      0.51      0.50    124131
weighted avg       0.59      0.64      0.61    124131
```

The accuracy is 64%, however this is not the full picture, and the value may be misleading when viewed in context of precision.

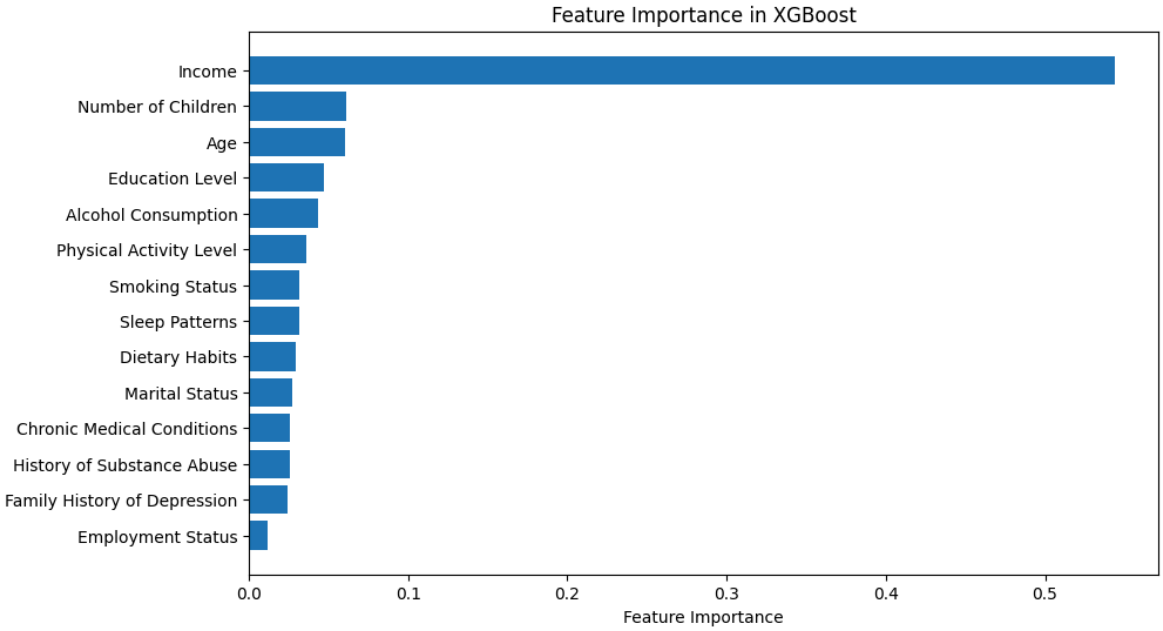
The model performs well in predicting whether a person has mental illness. I.e. it has a 70% precision on successfully predicting class 0 ('No to Mental Health Issues.). Precision on successfully predicting class 1 ('Yes to Mental Health Issues') was very poor at 34%.

Given that the problem statement is to predict Mental Health Issues given a set of features, a low precision on class 1 makes the usefulness of the model fairly limited.

Therefore an accuracy of 64% creates a very distorted view. Recall for the positive class "1" is very low, indicating that the model is struggling to identify those with mental illness. The precision for class 1 is also low, meaning that when the model predicts "yes", it's often wrong. The model is likely overfitting to the majority class 0 ("No"), which the overwhelming part of the dataset (70% of the data)

Feature Selection

In [32]:
```
display_feature_importance(model, X)
```

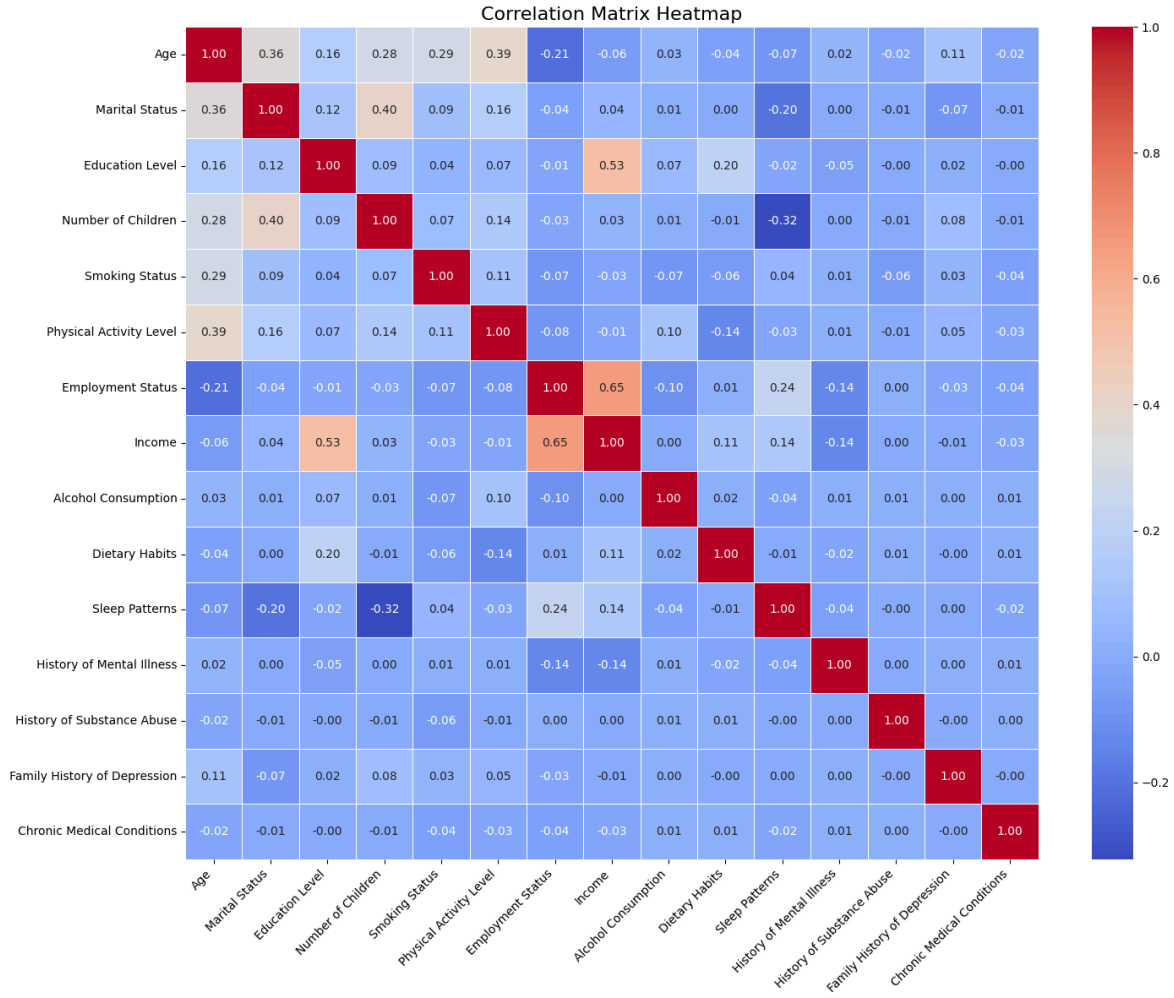


The feature importance makes interesting reading. Income is highly important. All other features seem to have less influence on the predictor.

The findings potentially suggest that correlation or multicollinearity between features. For example, employment status would have a positive correlation on income.

Assessing correlation could aid the feature selection process

In [33]:
```
display_correration_matrix(currated_dataset)
```



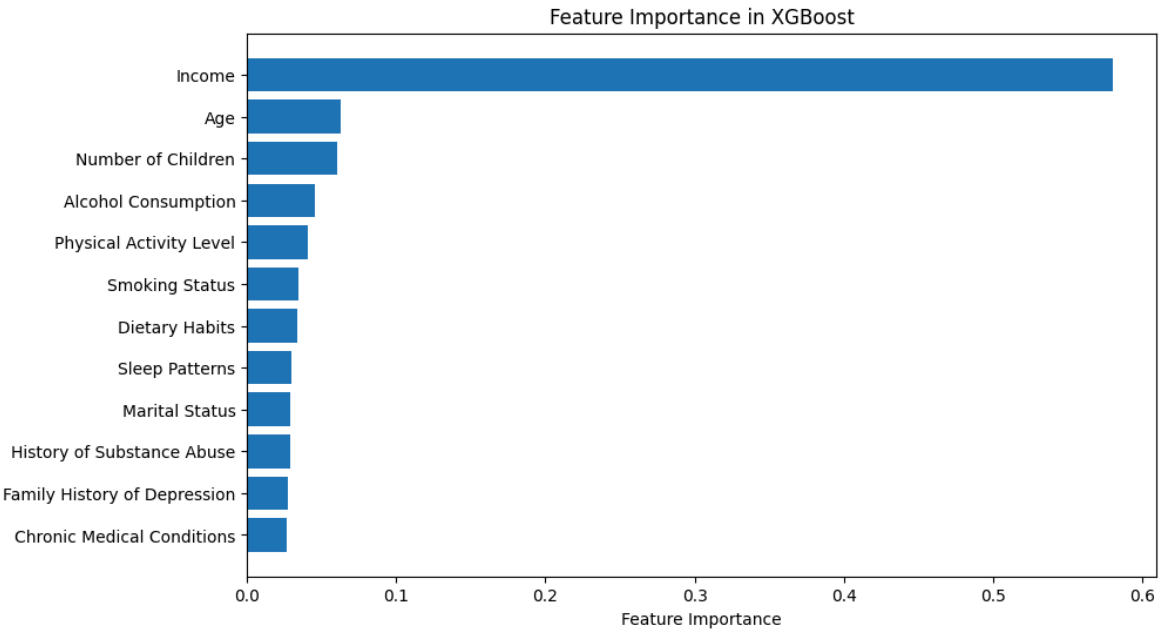
I was somewhat surprised by these results as I thought correlation would be more pronounced across the features.

The next step was to remove correlated features.

In [34]:

```
reduced_feature_df = currated_dataset.copy()
reduced_feature_df = currated_dataset.drop(columns=['Education Level', 'Employme

X = reduced_feature_df.drop(columns=['History of Mental Illness'])
y = reduced_feature_df['History of Mental Illness']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_

model.fit(
    X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# # Calculate the accuracy of the model

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
display_feature_importance(model, X)
```

```
Accuracy: 0.64
              precision    recall  f1-score   support

           0       0.70      0.84      0.77    201305
           1       0.35      0.19      0.25     88333

    accuracy                           0.64    289638
   macro avg       0.52      0.52      0.51    289638
weighted avg       0.59      0.64      0.61    289638
```

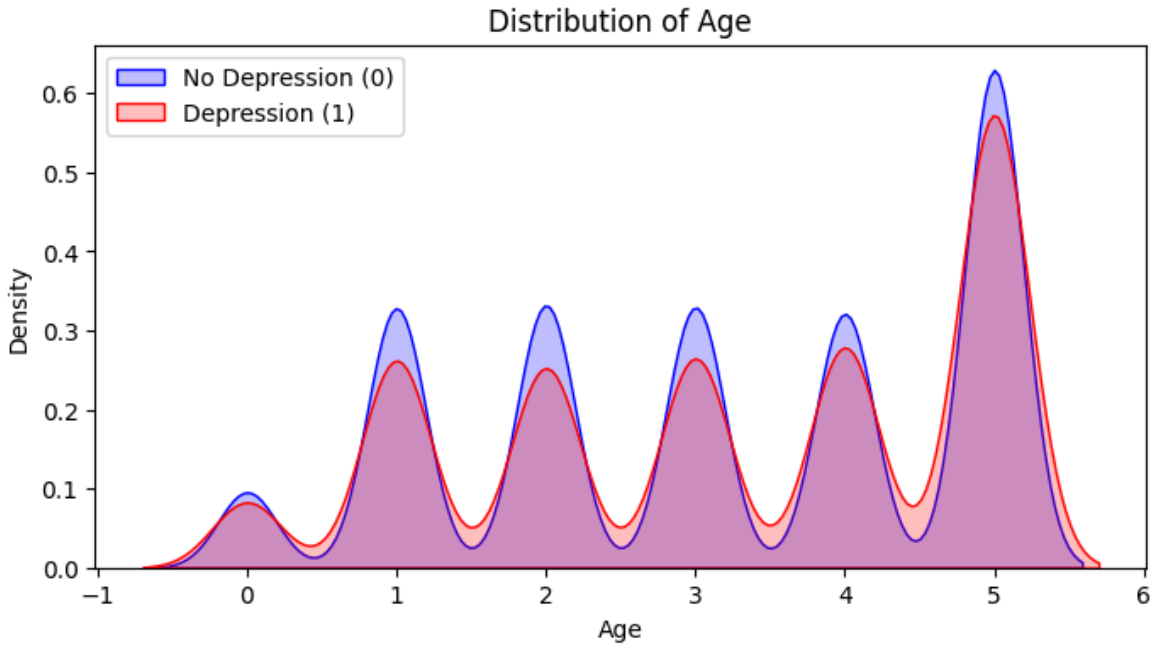


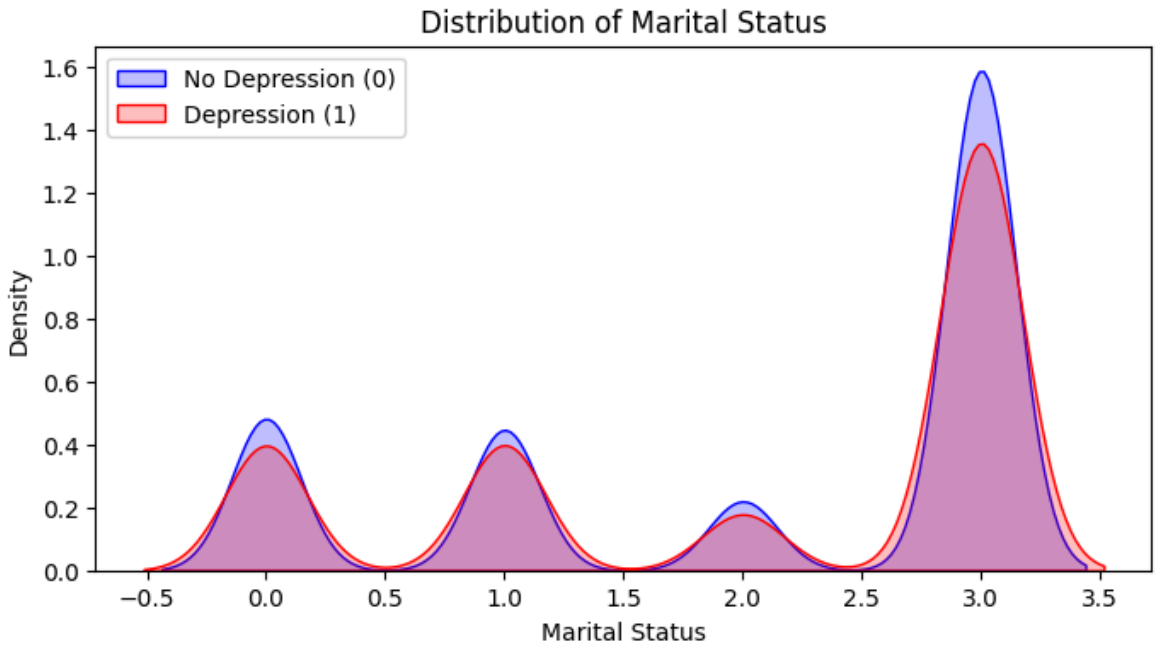
No improvement

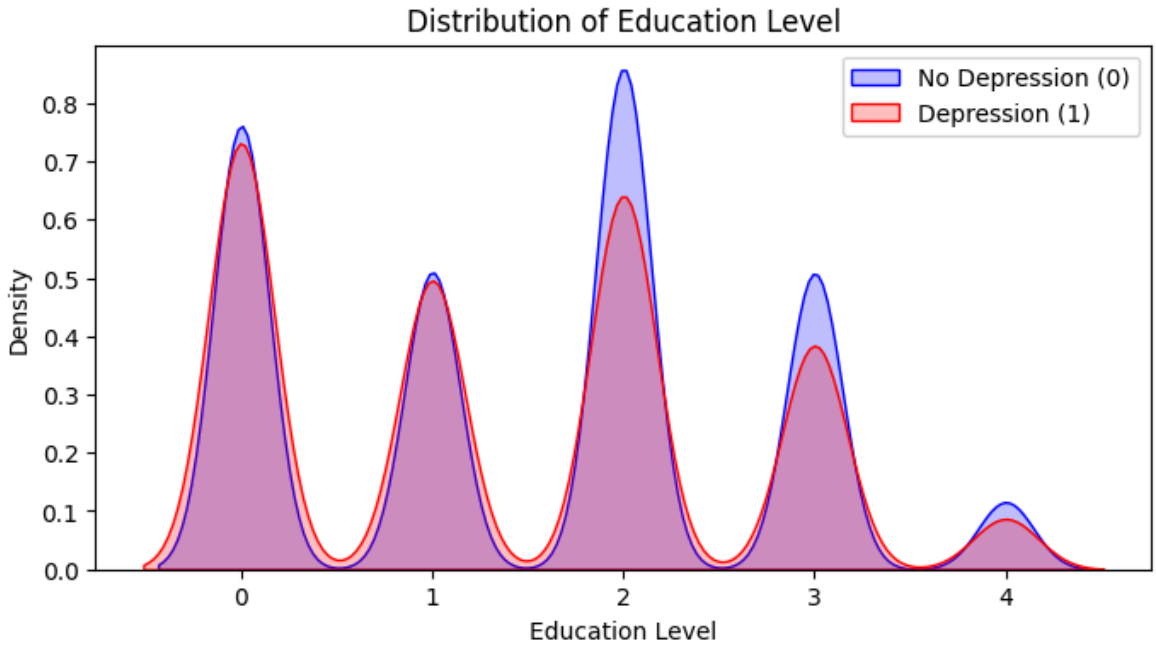
Next I investigate whether the distribution of the data. If there is a high degree of overlap, then it maybe the case that existing features fail to distinguish between classes. As such, introducing more complex models, or removing features may not improve improvement, as the model will struggle to draw a decision boundary between class 0 and class 1.

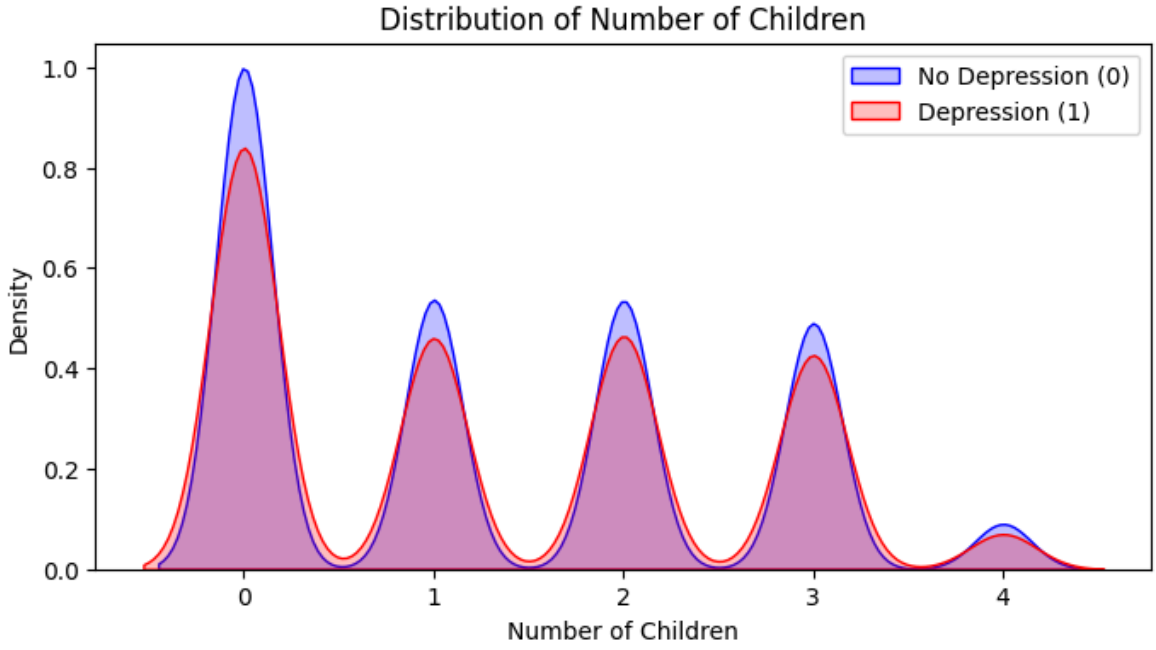
In [35]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

distribution_check = currated_dataset.copy()

# Separate data by classes
class_0 = distribution_check[distribution_check['History of Mental Illness'] ==
class_1 = distribution_check[distribution_check['History of Mental Illness'] ==

# Plot distribution for each feature
for feature in distribution_check.columns:
    if feature != 'History of Mental Illness':  # Skip the target column
        plt.figure(figsize=(8, 4))
        sns.kdeplot(class_0[feature], label='No Depression (0)', shade=True, col
        sns.kdeplot(class_1[feature], label='Depression (1)', shade=True, color=
        plt.title(f'Distribution of {feature}')
        plt.legend()
        plt.show()

# Pairplot to visualize feature interactions (if there aren't too many features)
```

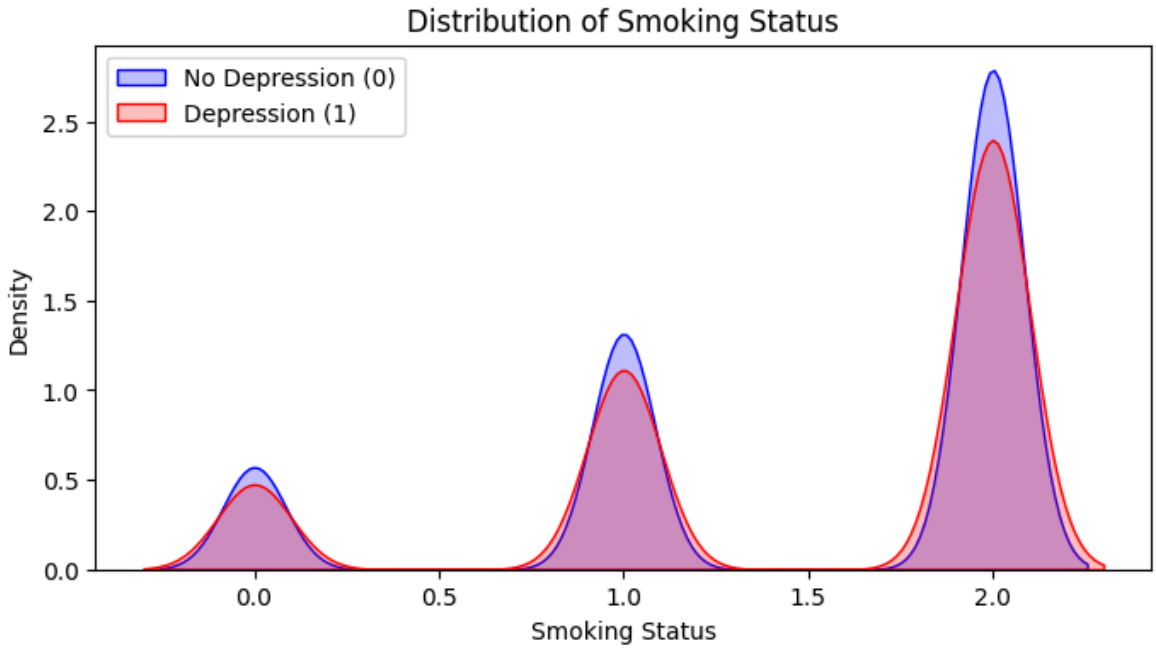
```
selected_features = ['Income', 'Employment Status']  # Replace with relevant fea
sns.pairplot(distribution_check, hue='History of Mental Illness', vars=selected_
plt.show()
```

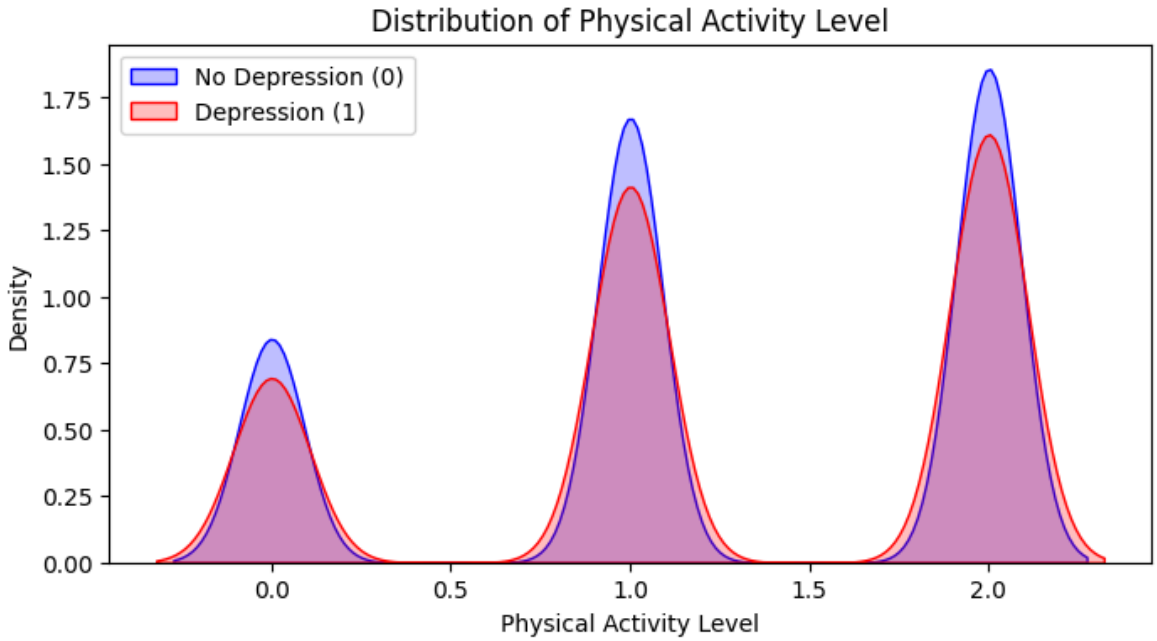


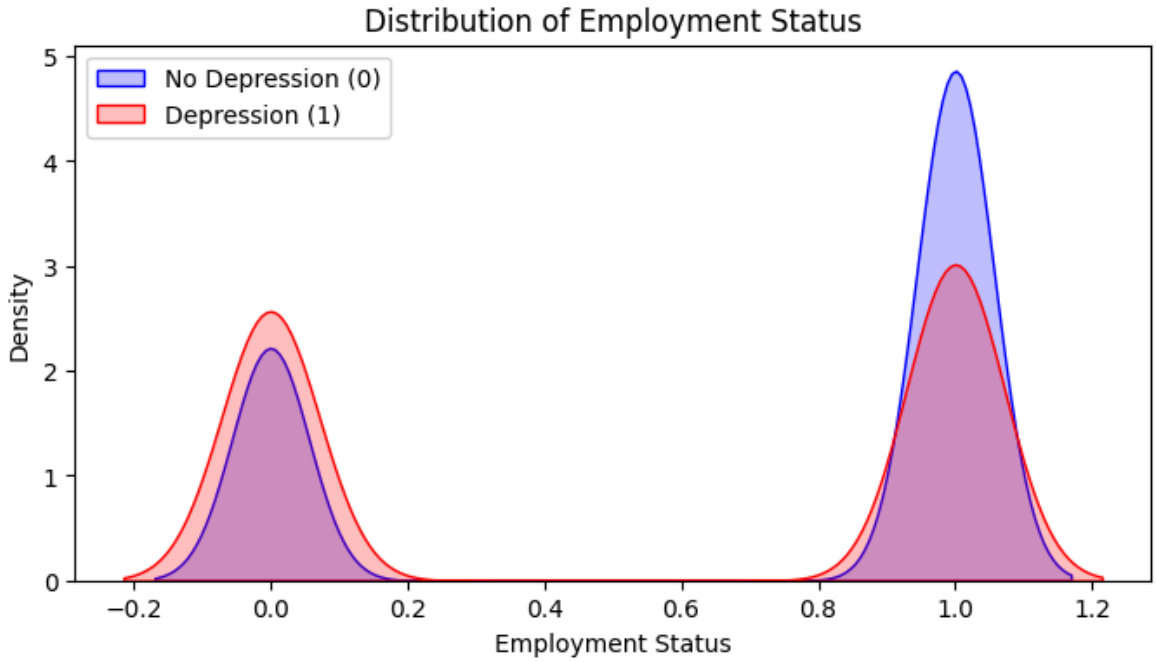


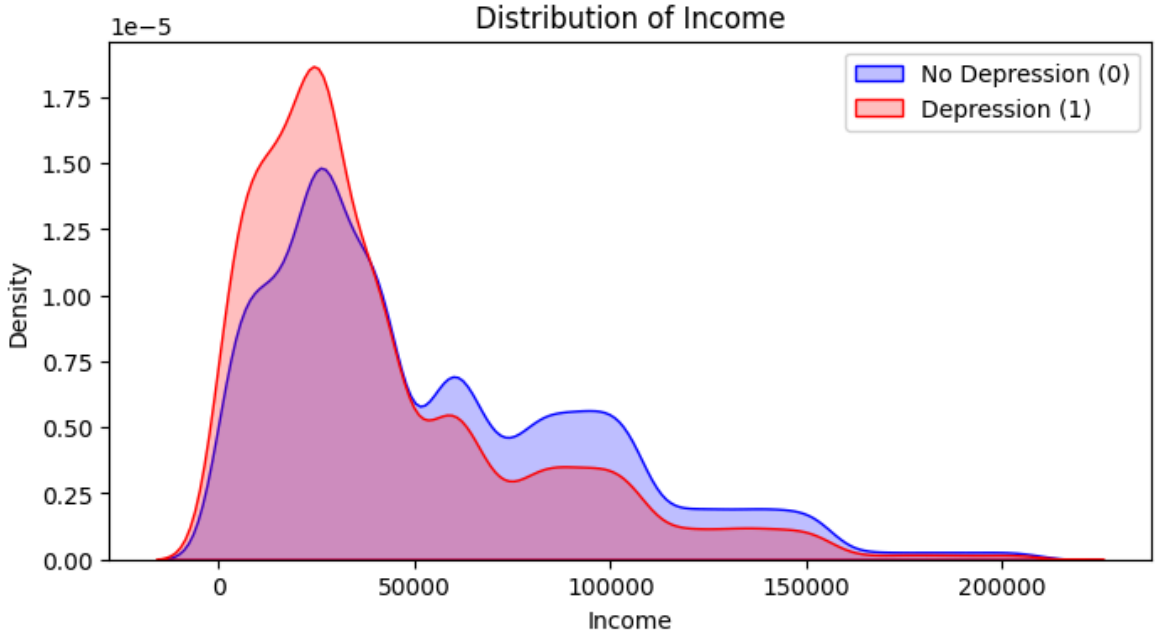

Distribution of Education Level
No Depression (0)
Depression (1)
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Education Level

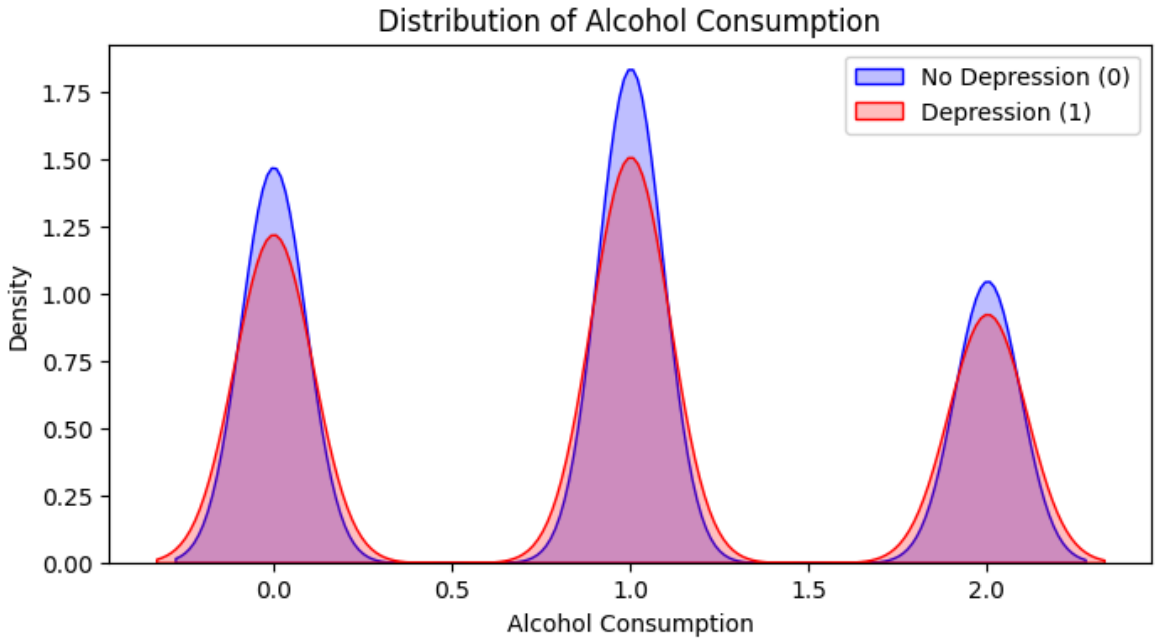

Distribution of Number of Children
No Depression (0)
Depression (1)
Density
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
Number of Children

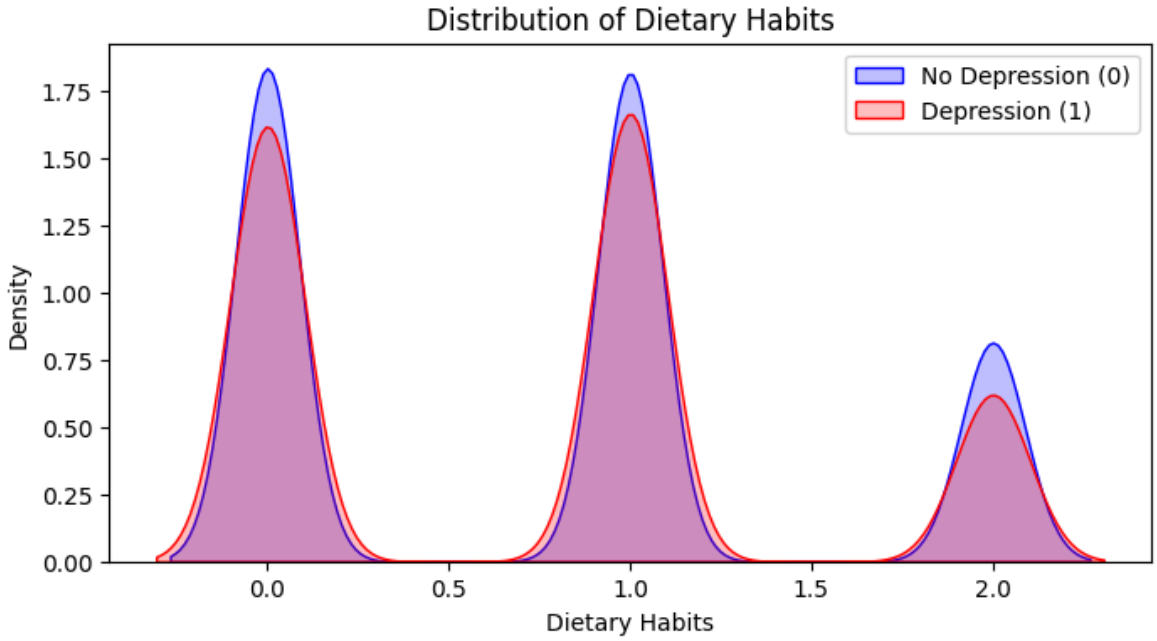

Distribution of Smoking Status
No Depression (0)
Depression (1)
Density
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.5
1.0
1.5
2.0
Smoking Status

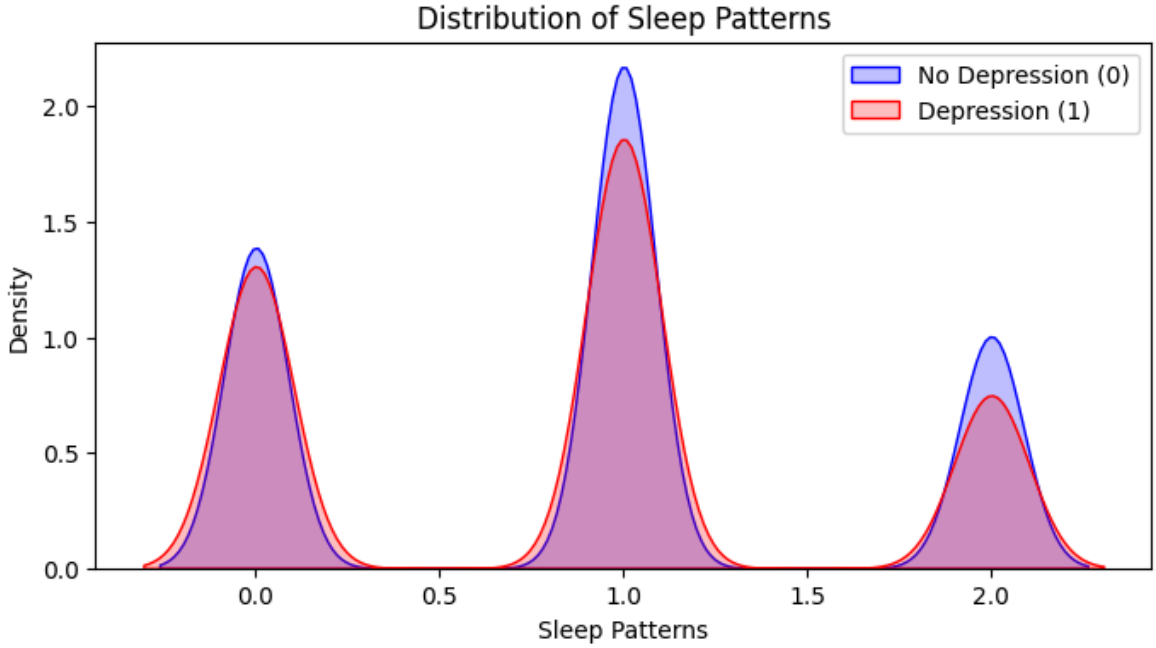

Distribution of Physical Activity Level
No Depression (0)
Depression (1)
Density
Physical Activity Level

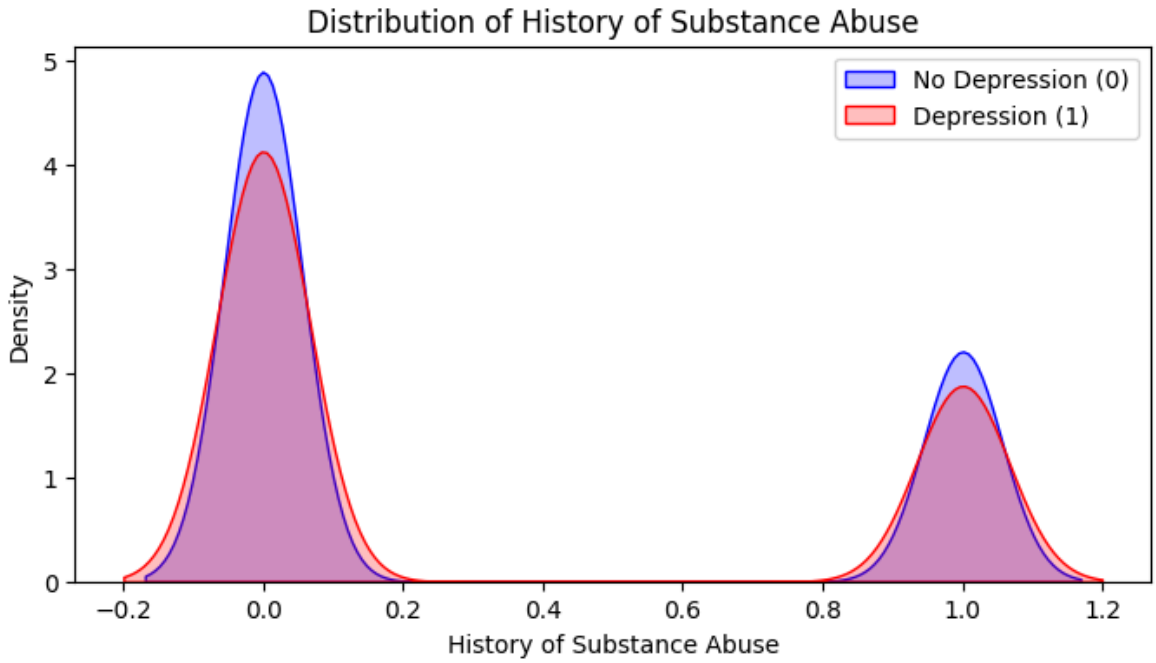

Distribution of Employment Status
No Depression (0)
Depression (1)
Density
Employment Status

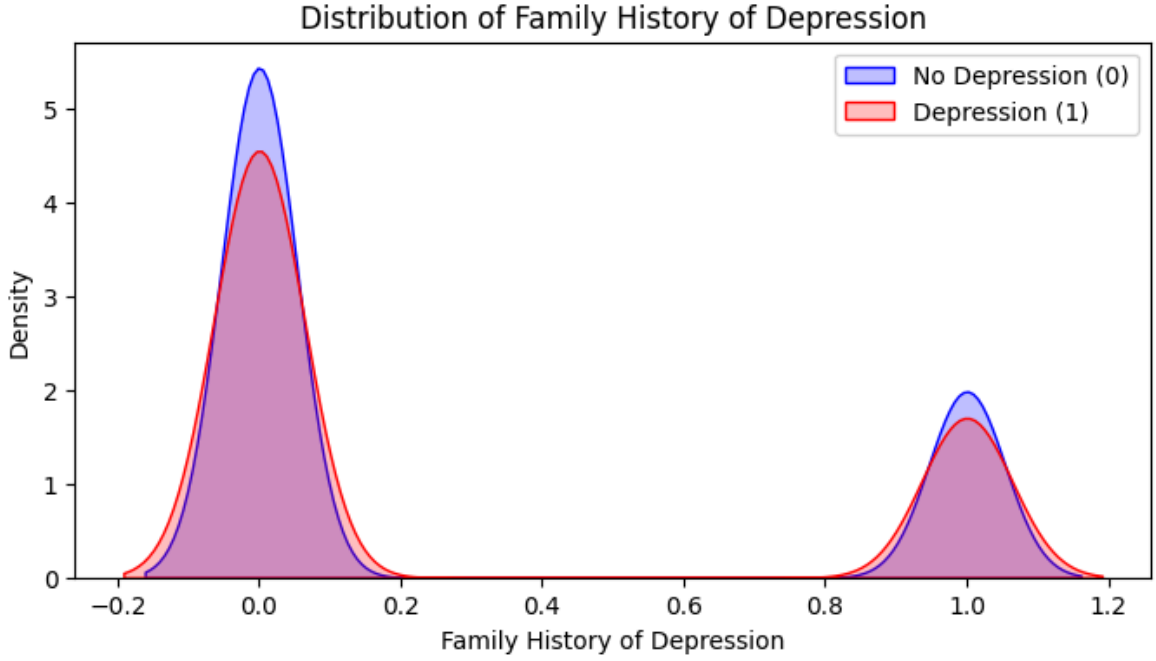

Distribution of Income
1e−5
No Depression (0)
Depression (1)
Density
Income

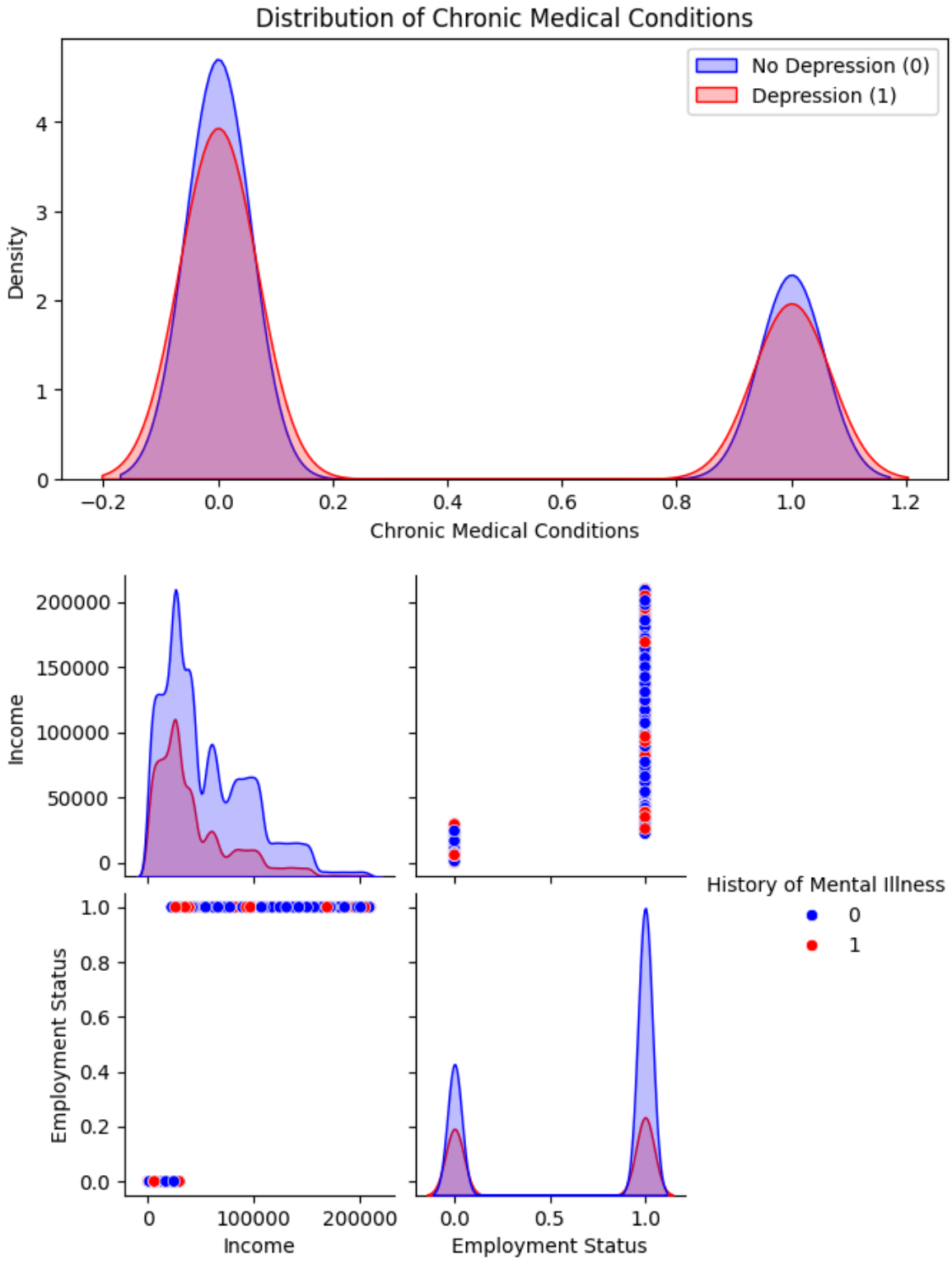

Distribution of Alcohol Consumption
No Depression (0)
Depression (1)
Density
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
Alcohol Consumption


Distribution of Dietary Habits
No Depression (0)
Depression (1)
Density
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
Dietary Habits


Distribution of Sleep Patterns
No Depression (0)
Depression (1)
Density
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
Sleep Patterns


Distribution of History of Substance Abuse
No Depression (0)
Depression (1)
Density
History of Substance Abuse
0
1
2
3
4
5
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Distribution of Family History of Depression
No Depression (0)
Depression (1)
Density
Family History of Depression
0
1
2
3
4
5
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2



The results show a high degree of overlap. Income exhibits the least overlap (however, it's still not great). This aligns with our observation that it is the most important feature for explaining mental illness within the dataset.

The results do bring into question the usefulness of most of the features in the dataset in explaining mental illness.

Conclusion

Removing features does not seem to have a huge effect on the model performance.

1. *Redundant Information was not affecting the Model. The Random Forest model may be handling the correlated features well, therefore correlated features are not introducing instability or noise into your model. That said, given those features have little predictive power, their inclusion is questionable.*

2. *Features Have Limited Predictive Power The Features are not the primary drivers of the target variable (e.g., depression may depend on unmeasured factors like genetics or social dynamics). We may need to source additional features that could improve model performance, e.g., healthcare history or stress levels.*

Mental illness is a complex condition. What we may be seeing here in the data, is that trying to ascribe features such as physical activity for example, may be too simplistic in explaining the phenomena. Mental illness is as much environmental as it is genetic, and therefore it's triggers may also be complex and poorly understood.