CAPSTONE PROJECT REPORT

Exploring Maternal Well-being: A Proposed Model for Comprehensive Statistical Analysis of Predictors for Suicide Attempts Among Postnatal Mothers.

INDIANA UNIVERSITY

Prepared By

Preethi Reddy Nomula & Sashank Dronamraju.

Under the Guidance of

Prof. Gary Schwebach, D.B.A., J.D

Professor of Practice, Health Informatics.

Luddy School of Informatics, Computing, and Engineering

535 W. Michigan Street, Indianapolis, Indiana – 46202.

Table of Contents

Abstract *3*

1 Project Scope *4*

1.1 Introduction 4

1.2 Etiology and Risk Factors 4

1.3 Symptoms and Diagnosis 4

1.4 Treatment and Management 4

1.5 Public Health Impact 5

2 Aim *5*

3 Research Question *5*

4 Purpose *6*

4.1 Significance 6

5 Methodology *7*

5.1 Data Collection 7

5.2 Data Preprocessing 7

5.2a Data Cleaning 7

5.2b Data Transformation 7

5.3 Exploratory Data Analysis (EDA) 8

5.4 Statistical Hypothesis Testing 8

5.5 Predictive Modeling 8

5.6 Tools and Technologies Used 9

6 Data Extraction *9*

6.1 Data Importing *12*

6.2 Data Cleaning 12

6.3 Label Encoding 14

6.4 Data Visualization 16

7 Correlation Testing *33*

8 Hypothesis Testing and Model Building *36*

9.2 Model Performance 43

10 Conclusion *43*

11 Project Challenges *43*

12 Recommendations for Future Research *44*

13 References *45*

Abstract

Postnatal depression represents a significant threat to maternal health, often manifesting as severe psychological distress that, in the most severe cases, can lead to suicide attempts. The postpartum period introduces a range of complex challenges that can intensify existing mental health conditions. Our study is committed to deepening the understanding of these challenges and enhancing the predictability of associated risks, which is essential for improving the effectiveness of interventions during this vulnerable time. Employing a data-driven methodology, our research utilizes a comprehensive dataset from Kaggle on postnatal depression to construct predictive models. These models systematically incorporate a variety of psychological, behavioral, and demographic factors to identify mothers at high risk for suicide attempts. Our research aims to forge customized support systems for postnatal mothers by pinpointing key predictors and elucidating their interrelationships. The ultimate goal is to advance maternal mental health outcomes by implementing targeted early intervention strategies. This study proposes a robust model for the comprehensive statistical analysis of predictors associated with suicide attempts among new postnatal mothers, offering a framework for proactive healthcare engagement.

Keywords: Age, Data Visualization, Bivariate Analysis, Features, Feeling Anxious, Multifaceted, Postpartum Depression.

1 Project Scope

1.1 Introduction

Postpartum depression (PPD) is a complex, multifaceted mood disorder that affects many mothers following childbirth, characterized by persistent sadness, anxiety, and fatigue that can interfere significantly with a woman's ability to care for herself or her family. Unlike the "baby blues," which resolves spontaneously, PPD can persist for months or even years if untreated and may impact the mother's health and the infant’s development (Stewart & Vigod, 2019) (Suryawanshi & Pajai, 2022).

1.2 Etiology and Risk Factors

The etiology of PPD is not fully understood but is believed to involve a combination of hormonal, biochemical, environmental, psychological, and genetic factors. After childbirth, a significant drop in hormones such as estrogen and progesterone, along with changes in thyroid hormone levels, may lead to PPD. Psychological risk factors include a history of depression, anxiety, significant life stressors, and a lack of support from family or partners (Stewart & Vigod, 2019).

1.3 Symptoms and Diagnosis

Symptoms of PPD can vary but typically include severe mood swings, feelings of inadequacy, inability to bond with the baby, thoughts of harming oneself or the baby, and disturbances in sleeping and eating habits. Diagnosis is based on clinical assessment and often involves screening tools such as the Edinburgh Postnatal Depression Scale (EPDS) or the Postpartum Depression Screening Scale (PDSS) (Stewart & Vigod, 2019).

1.4 Treatment and Management

Treatment for PPD may involve a combination of psychotherapy, such as cognitive-behavioral therapy (CBT) or interpersonal therapy (IPT), and medications like antidepressants. In addition to medical

treatment, increased social support from family, friends, and peer groups can be beneficial (Stewart & Vigod, 2019).

1.5 Public Health Impact

PPD represents a significant public health challenge due to its impact on the mother and child's well-being. Healthcare systems must integrate routine postnatal checks that include mental health assessments to ensure early diagnosis and intervention (Suryawanshi & Pajai, 2022).

2 Aim

The primary aim of our research is to identify and understand the multifaceted factors that contribute to the heightened risk of suicide attempts among postnatal mothers. By integrating advanced statistical models and machine learning algorithms, through this study, we sought to develop a predictive model that can accurately forecast the likelihood of suicide attempts based on various psychological, behavioral, and demographic indicators. This model is intended to serve as a tool for healthcare providers to initiate early interventions and support systems tailored to the needs of new mothers experiencing postnatal depression.

3 Research Question

"Can we predict the likelihood of suicide attempts among postnatal mothers based on identifiable factors such as age, emotional well-being, and other postnatal conditions?"

This question directs the study's focus toward a quantitative analysis of how specific symptoms and demographic characteristics correlate with the risk of suicide attempts.

Null Hypothesis: The variables provided in the dataset like 'Age', 'Feeling sad or tearful', 'Irritable towards baby and partner', 'Trouble sleeping at night', 'Problems concentrating or making decision', 'Overeating, or loss of appetite', 'Feeling anxious', 'Feeling of guilt', 'Problems of bonding with baby' does not have a significant impact on suicide attempts.

Alternate Hypothesis: The variables provided in the dataset like 'Age', 'Feeling sad or tearful', 'Irritable towards baby and partner', 'Trouble sleeping at night', 'Problems concentrating or making decision', 'Overeating, or loss of appetite', 'Feeling anxious', 'Feeling of guilt', 'Problems of bonding with baby' have a significant impact on suicide attempts.

4 Purpose

The purpose of our study extends beyond academic inquiry; it is fundamentally about enhancing maternal mental health care. The project aims to provide healthcare professionals with a practical tool for early risk identification and intervention by developing a predictive model that utilizes real-world hospital survey report data. This effort aligns with broader public health goals of reducing maternal morbidity and mortality associated with postnatal depression and improving the overall well-being of mothers and their families.

4.1 Significance

The significance of our research is multifaceted. Firstly, it addresses a critical gap in maternal health services by focusing on the postnatal period - a time often characterized by inadequate mental health monitoring. Secondly, our study enriches the literature on mental health risks associated with postnatal depression by providing empirical evidence that can inform more effective healthcare policies and practices. Thirdly, by identifying predictors of suicide attempts, our research highlights the necessity for integrated care approaches that account for the complex interplay of emotional, behavioral, and social factors affecting maternal well-being.

Furthermore, this research contributes significantly to understanding the profound impacts of postnatal phase changes on mental health. It clarifies how emotional, behavioral, and interpersonal dynamics contribute to suicide risks, thereby guiding the development of targeted preventative interventions. This

study not only advances academic knowledge but also serves as a crucial resource for developing strategies aimed at mitigating the risks associated with postnatal depression.

5 Methodology

The methodology employed in our project is a detailed process that integrates data handling, exploratory data analysis (EDA), statistical testing, and machine learning modeling to predict suicide attempts among postnatal mothers. This approach aims to harness advanced analytics capabilities to derive insights that can inform targeted healthcare interventions during a critical period for new mothers. Here's an in-depth look at each stage of the methodology:

5.1 Data Collection

In our project, we used the "Postnatal Depression" dataset available on Kaggle. This dataset includes a range of variables related to postnatal depression symptoms, demographic data, and behavioral indicators linked to the mental health of new mothers.

Dataset Link: https://www.kaggle.com/datasets/parvezalmuqtadir2348/postpartum-depression

5.2 Data Preprocessing

5.2a Data Cleaning

- Missing values were identified and handled appropriately to ensure the robustness of the models. This involved filling or removing missing data depending on the nature and the impact of the missing data on the overall dataset integrity.
- Duplicate records were removed to prevent any bias or skewed results in the analysis.

5.2b Data Transformation

- Categorical data were encoded to numerical values to facilitate their use in statistical tests and machine learning models. For example, binary variables were encoded as 0 and 1, and categorical

variables with more than two categories were handled using one-hot encoding or label encoding depending on the context.

5.3 Exploratory Data Analysis (EDA)

- **Distribution Analysis:** Key variables were visualized using bar graphs to understand their distribution.
- **Correlation Analysis:** Correlations between variables were examined using Spearman correlation coefficients to identify potential predictors of suicide attempts.

5.4 Statistical Hypothesis Testing

- **Chi-square Tests:** These tests were conducted to explore associations between categorical variables and the target variable (suicide attempts). This helped us understand which factors are significantly associated with the risk of suicide attempts.
- The results of these tests informed the selection of variables for the predictive modeling.

5.5 Predictive Modeling

- **Model Selection:** Several machine learning algorithms were considered, including Random Forest, Logistic Regression, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN). Each model was chosen based on its suitability to handle binary classification problems and its ability to manage overfitting.
- **Training and Testing:** The dataset was split into training and testing sets, with the training set used to build the models and the testing set used to evaluate their performance.
- **Model Evaluation:** Models were evaluated based on accuracy, precision, recall, and F1 score. The Random Forest model performed best, suggesting its effectiveness in dealing with imbalanced datasets and complex relationships.

5.6 Tools and Technologies Used

Programming Languages and Libraries: Python was the primary programming language, with libraries such as Pandas for data manipulation, NumPy for numerical operations, Matplotlib and Seaborn for data visualization, Scikit-learn for machine learning, and SciPy for additional statistical tests.

This comprehensive methodology provided a robust framework for analyzing the risk factors associated with suicide attempts among postnatal mothers.

6 Data Extraction

The initial phase of our project involved extracting data from the provided Kaggle dataset, which was formatted as a CSV file. We conducted a thorough review of the dataset to understand the recording mechanisms for each column and to identify any instances of missing data. Where null values were encountered, we deliberated on potential strategies to appropriately address these gaps.

The overarching goal of this project is to equip healthcare professionals with a deeper understanding of postpartum depression. By identifying potential risk factors and enhancing screening and evaluation methods, we aim to support the development of more tailored and effective intervention plans for managing this critical condition.

The Eleven attributes mentioned in the dataset are:

1. Timestamp
2. Age group
3. Feeling sad or tearful.
4. Irritable towards baby and partner
5. Trouble Sleeping at night
6. Problems Concentrating or making decisions
7. Overeating or Loss of Appetite

8. Feeling anxious

9. Feeling guilty

10. Problems of bounding with baby

11. Suicide Attempt

All of the attributes in the dataset share the same variable type, which is categorical. However, they differ in their data types. "Timestamp" has a data type of date time, "Age group" has a data type of integer, and the remaining attributes have a data type of string.

Now let us examine the descriptions of each attribute that were provided in the dataset.

Column Name	**Data Type**	**Variable Type**	**Description**
Timestamp	Date Time	Categorical	Timestamp of survey
Age group	Integer	Categorical	The age group of people surveyed.
Feeling sad or tearful	String	Categorical	Their opinion on Feeling sad or tearful.
Irritable towards baby and partner	String	Categorical	Their opinion on Irritable towards baby and partner.
Trouble sleeping at night	String	Categorical	Their opinion on Trouble sleeping at night.

Problems concentrating or making decisions	String	Categorical	Their opinion on Problems concentrating or making decisions.
Loss of appetite	String	Categorical	Their opinion on Loss of appetite.
Feeling Anxious	String	Categorical	Their opinion on Feeling Anxious.
Feeling of guilt	String	Categorical	Their opinion on Feeling of guilt.
Problems of bonding with baby	String	Categorical	Their opinion on Problems of bonding with baby.
Suicide attempt	String	Categorical	Their opinion on Suicide attempts.

6.1 Data Importing

This is how our data was imported:

Importing Libraries

```
[850]: import pandas as pd
       import numpy as np
       import matplotlib.pyplot as plt
       import seaborn as sns
```

Data Importing

```
[851]: PN = pd.read_csv('post natal data 2.csv')
       display(PN)
```

	Timestamp	Age	Feeling sad or Tearful	Irritable towards baby & partner	Trouble sleeping at night	Problems concentrating or making decision	Overeating or loss of appetite	Feeling anxious	Feeling of guilt	Problems of bonding with baby	Suicide attempt
0	6/14/2022 20:02	35-40	Yes	Yes	Two or more days a week	Yes	Yes	Yes	No	Yes	Yes
1	6/14/2022 20:03	40-45	Yes	No	No	Yes	Yes	No	Yes	Yes	No
2	6/14/2022 20:04	35-40	Yes	No	Yes	Yes	Yes	Yes	No	Sometimes	No
3	6/14/2022 20:05	35-40	Yes	Yes	Yes	Yes	No	Yes	Maybe	No	No
4	6/14/2022 20:06	40-45	Yes	No	Two or more days a week	Yes	No	Yes	No	Yes	No
...	...	...	...	...	...	...	...	...	...	...	...
1498	6/15/2022 0:35	30-35	Yes	No	Two or more days a week	No	No	Yes	Maybe	Sometimes	No
1499	6/15/2022 0:35	25-30	Sometimes	No	No	Often	No	Yes	Maybe	Yes	No
1500	6/15/2022 0:35	25-30	No	Sometimes	Two or more days a week	No	No	No	Yes	No	Not interested to say
1501	6/15/2022 0:36	25-30	No	Sometimes	Yes	Often	No	Yes	No	No	No
1502	6/15/2022 0:36	45-50	Sometimes	Sometimes	Two or more days a week	No	No	No	Maybe	No	No

1503 rows × 11 columns

Data Overview

Fig: 1

6.2 Data Cleaning

Data Cleaning

```
[854]: PN.isnull().sum()
```

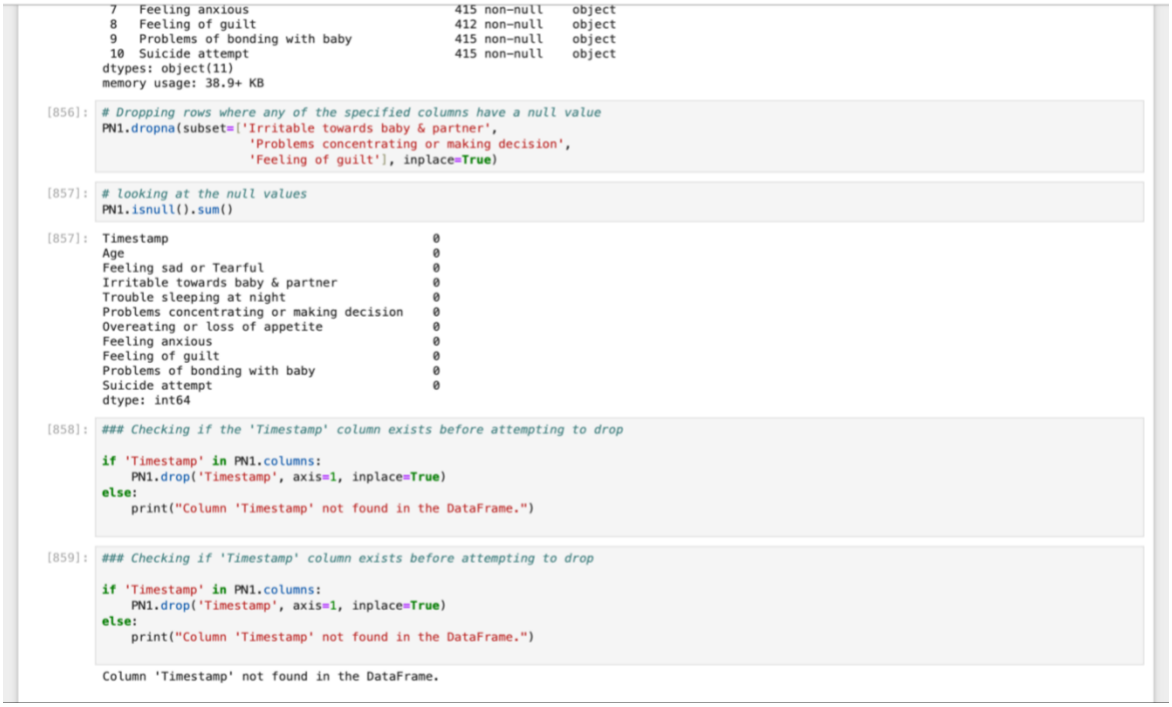
```
[854]: Timestamp                                 0
       Age                                       0
       Feeling sad or Tearful                    0
       Irritable towards baby & partner          6
       Trouble sleeping at night                 0
       Problems concentrating or making decision 12
       Overeating or loss of appetite            0
       Feeling anxious                           0
       Feeling of guilt                          9
       Problems of bonding with baby             0
       Suicide attempt                           0
       dtype: int64
```

```
[855]: PN1 = PN.copy()
       PN1.drop_duplicates(inplace = True)
       PN1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 415 entries, 0 to 469
Data columns (total 11 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Timestamp                                 415 non-null    object
 1   Age                                       415 non-null    object
 2   Feeling sad or Tearful                    415 non-null    object
 3   Irritable towards baby & partner          413 non-null    object
 4   Trouble sleeping at night                 415 non-null    object
 5   Problems concentrating or making decision 411 non-null    object
 6   Overeating or loss of appetite            415 non-null    object
 7   Feeling anxious                           415 non-null    object
 8   Feeling of guilt                          412 non-null    object
 9   Problems of bonding with baby             415 non-null    object
 10  Suicide attempt                           415 non-null    object
dtypes: object(11)
memory usage: 38.9+ KB
```

```
[856]: # Dropping rows where any of the specified columns have a null value
       PN1.dropna(subset=['Irritable towards baby & partner',
                          'Problems concentrating or making decision',
                          'Feeling of guilt'], inplace=True)
```

Fig: 2



```
 7   Feeling anxious                          415 non-null    object
 8   Feeling of guilt                         412 non-null    object
 9   Problems of bonding with baby            415 non-null    object
 10  Suicide attempt                          415 non-null    object
dtypes: object(11)
memory usage: 38.9+ KB
```

```
[856]: # Dropping rows where any of the specified columns have a null value
PN1.dropna(subset=['Irritable towards baby & partner',
                   'Problems concentrating or making decision',
                   'Feeling of guilt'], inplace=True)
```

```
[857]: # looking at the null values
PN1.isnull().sum()
```

```
[857]: Timestamp                                0
Age                                      0
Feeling sad or Tearful                   0
Irritable towards baby & partner         0
Trouble sleeping at night                0
Problems concentrating or making decision    0
Overeating or loss of appetite           0
Feeling anxious                          0
Feeling of guilt                         0
Problems of bonding with baby            0
Suicide attempt                          0
dtype: int64
```

```
[858]: ### Checking if the 'Timestamp' column exists before attempting to drop

if 'Timestamp' in PN1.columns:
    PN1.drop('Timestamp', axis=1, inplace=True)
else:
    print("Column 'Timestamp' not found in the DataFrame.")
```

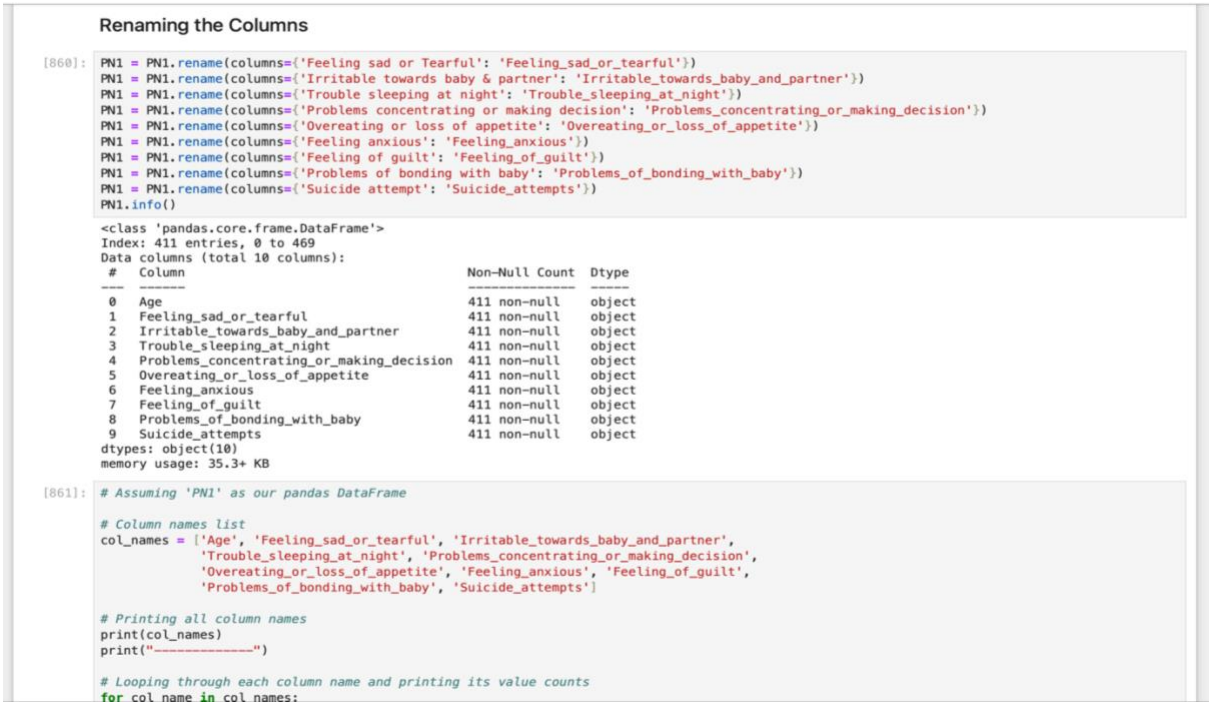
```
[859]: ### Checking if 'Timestamp' column exists before attempting to drop

if 'Timestamp' in PN1.columns:
    PN1.drop('Timestamp', axis=1, inplace=True)
else:
    print("Column 'Timestamp' not found in the DataFrame.")
```

```
Column 'Timestamp' not found in the DataFrame.
```

Fig: 3

We implemented a systematic approach to data cleaning that involved several key steps to ensure the integrity and usability of our dataset. Initially, we conducted a thorough examination for any null values within the dataset. Following this, we eliminated any duplicate entries to maintain the uniqueness and relevance of our data. Subsequently, we specifically targeted columns that contained null values and removed them to enhance data quality. Lastly, we removed the 'Timestamp' column from the Data Frame, as it was not pertinent to our analysis, streamlining the dataset for the subsequent stages of our study.

6.3 Label Encoding



Renaming the Columns

```
[860]: PN1 = PN1.rename(columns={'Feeling sad or Tearful': 'Feeling_sad_or_tearful'})
PN1 = PN1.rename(columns={'Irritable towards baby & partner': 'Irritable_towards_baby_and_partner'})
PN1 = PN1.rename(columns={'Trouble sleeping at night': 'Trouble_sleeping_at_night'})
PN1 = PN1.rename(columns={'Problems concentrating or making decision': 'Problems_concentrating_or_making_decision'})
PN1 = PN1.rename(columns={'Overeating or loss of appetite': 'Overeating_or_loss_of_appetite'})
PN1 = PN1.rename(columns={'Feeling anxious': 'Feeling_anxious'})
PN1 = PN1.rename(columns={'Feeling of guilt': 'Feeling_of_guilt'})
PN1 = PN1.rename(columns={'Problems of bonding with baby': 'Problems_of_bonding_with_baby'})
PN1 = PN1.rename(columns={'Suicide attempt': 'Suicide_attempts'})
PN1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 411 entries, 0 to 469
Data columns (total 10 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Age                                   411 non-null    object
 1   Feeling_sad_or_tearful                411 non-null    object
 2   Irritable_towards_baby_and_partner    411 non-null    object
 3   Trouble_sleeping_at_night             411 non-null    object
 4   Problems_concentrating_or_making_decision  411 non-null  object
 5   Overeating_or_loss_of_appetite        411 non-null    object
 6   Feeling_anxious                       411 non-null    object
 7   Feeling_of_guilt                      411 non-null    object
 8   Problems_of_bonding_with_baby         411 non-null    object
 9   Suicide_attempts                      411 non-null    object
dtypes: object(10)
memory usage: 35.3+ KB
```

```
[861]: # Assuming 'PN1' as our pandas DataFrame

# Column names list
col_names = ['Age', 'Feeling_sad_or_tearful', 'Irritable_towards_baby_and_partner',
             'Trouble_sleeping_at_night', 'Problems_concentrating_or_making_decision',
             'Overeating_or_loss_of_appetite', 'Feeling_anxious', 'Feeling_of_guilt',
             'Problems_of_bonding_with_baby', 'Suicide_attempts']

# Printing all column names
print(col_names)
print("-------------")

# Looping through each column name and printing its value counts
for col_name in col_names:
```

Fig: 4

Before label encoding, we refined the dataset by renaming the column headers to ensure clarity and ease of coding. Subsequently, we systematically applied label encoding to each column as follows:

Label Encoding

```
[862]: def encode_feelings(feeling):
    if feeling == 'Yes':
        return 3
    elif feeling == 'No':
        return 1
    elif feeling == 'Sometimes':
        return 2
    else:
        return 0  ## For any other category not explicitly defined

### Applying the custom encoding function to the 'Feeling_sad_or_tearful' column

PN1['Feeling_sad_or_tearful_encoded'] = PN1['Feeling_sad_or_tearful'].apply(encode_feelings)
```

```
[863]: def encode_Irritable(Irritability):
    if Irritability == 'Yes':
        return 3
    elif Irritability == 'No':
        return 1
    elif Irritability == 'Sometimes':
        return 2
    else:
        return 0

PN1['Irritable_towards_baby_and_partner_encoded'] = PN1['Irritable_towards_baby_and_partner'].apply(encode_Irritable)
```

```
[864]: def encode_Sleeping(Trouble_sleeping):
    if Trouble_sleeping == 'Yes':
        return 3
    elif Trouble_sleeping == 'No':
        return 1
    elif Trouble_sleeping == 'Two or more days a week':
        return 2
    else:
        return 0

PN1['Trouble_sleeping_at_night_encoded'] = PN1['Trouble_sleeping_at_night'].apply(encode_Sleeping)
```

```
[865]: def encode_Problems(Problems):
    if Problems == 'Yes':
```

Fig: 5

[865]:
```
def encode_Problems(Problems):
    if Problems == 'Yes':
        return 3
    elif Problems == 'No':
        return 1
    elif Problems == 'Often':
        return 2
    else:
         return 0

PN1['Problems_concentrating_or_making_decision_encoded'] = PN1['Problems_concentrating_or_making_decision'].apply(encode_Problems)
```

[866]:
```
def encode_Eating(Overeating):
    if Overeating == 'Yes':
        return 3
    elif Overeating == 'No':
        return 1
    elif Overeating == 'Not at all':
        return 2
    else:
         return 0

PN1['Overeating_or_loss_of_appetite_encoded'] = PN1['Overeating_or_loss_of_appetite'].apply(encode_Eating)
```

[867]:
```
def encode_Anxious(Feeling_Anxious):
    if Feeling_Anxious == 'Yes':
        return 2
    elif Feeling_Anxious == 'No':
        return 1
    else:
         return 0

PN1['Feeling_anxious_encoded'] = PN1['Feeling_anxious'].apply(encode_Anxious)
```

[868]:
```
def encode_Guilt(Feeling_guilt):
    if Feeling_guilt == 'Yes':
        return 3
    elif Feeling_guilt == 'No':
        return 1
    elif Feeling_guilt == 'Maybe':
        return 2
    else:
         return 0
```

Fig: 6

```
        return 2
    elif Feeling_Anxious == 'No':
        return 1
    else:
         return 0

PN1['Feeling_anxious_encoded'] = PN1['Feeling_anxious'].apply(encode_Anxious)
```

[868]:
```
def encode_Guilt(Feeling_guilt):
    if Feeling_guilt == 'Yes':
        return 3
    elif Feeling_guilt == 'No':
        return 1
    elif Feeling_guilt == 'Maybe':
        return 2
    else:
         return 0

PN1['Feeling_of_guilt_encoded'] = PN1['Feeling_of_guilt'].apply(encode_Guilt)
```

[869]:
```
def encode_Bonding(Problem_bonding):
    if Problem_bonding == 'Yes':
        return 3
    elif Problem_bonding == 'No':
        return 1
    elif Problem_bonding == 'Sometimes':
        return 2
    else:
         return 0

PN1['Problems_of_bonding_with_baby_encoded'] = PN1['Problems_of_bonding_with_baby'].apply(encode_Bonding)
```

[870]:
```
def encode_Suicides(Suicide):
    if Suicide == 'Yes':
        return 3
    elif Suicide == 'No':
        return 1
    elif Suicide == 'Not interested to say':
        return 2
    else:
         return 0

PN1['Suicide_attempts_encoded'] = PN1['Suicide_attempts'].apply(encode_Suicides)
```

Fig: 7

6.4 Data Visualization

Initially, for data visualization, we systematically displayed the distribution of each encoded column as illustrated below:

Visualizing the Encoded Columns

```
[871]: ### Showing the distribution of the encoded values
print(PN1['Feeling_sad_or_tearful_encoded'].value_counts())
```

```
Feeling_sad_or_tearful_encoded
3    147
1    147
2    117
Name: count, dtype: int64
```

```
[872]: ### Visualizing the Column 'Feeling sad or tearful'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Sometimes', '3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Feeling_sad_or_tearful_encoded', data = PN1)
plt.title('Distribution of Feeling sad or tearful')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

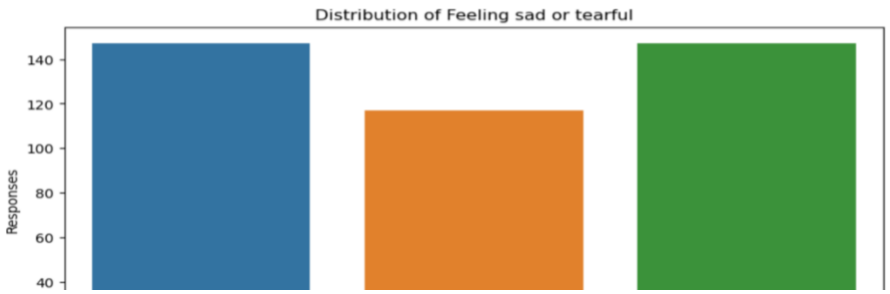


Fig: 8

This figure displays a bar graph visualizing the distribution of responses to the feeling of being sad or tearful, based on encoded values. The chart reveals that the encoded values '1' (No), '2' (Sometimes), and '3' (Yes) each have 147 instances, indicating an equal distribution among the three categories. This uniformity suggests that the sampled population has diverse emotional experiences, with equal parts never feeling sad or tearful, sometimes feeling so, and consistently experiencing these feelings. Such a balanced distribution underscores the complexity and variability of emotional states in the population studied, providing a nuanced view that could inform tailored therapeutic or intervention strategies aimed at addressing specific emotional patterns.

```
2    117
Name: count, dtype: int64
```

```
[872]: ### Visualizing the Column 'Feeling sad or tearful'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Sometimes', '3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Feeling_sad_or_tearful_encoded', data = PN1)
plt.title('Distribution of Feeling sad or tearful')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

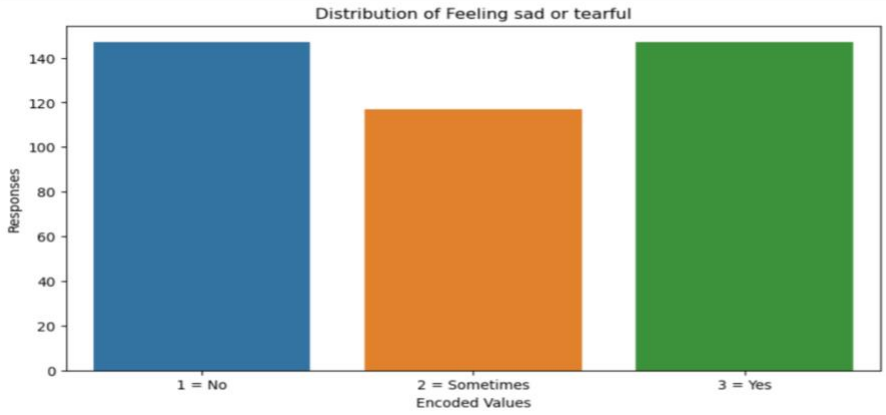
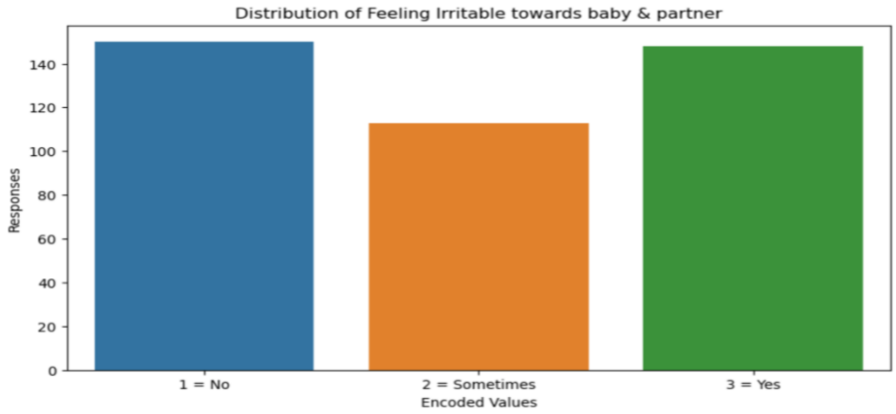


Fig: 9

This bar graph visualizes the distribution of responses for feelings of sadness or tearfulness encoded as '1' for 'No', '2' for 'Sometimes', and '3' for 'Yes'. The data indicates an equal number of respondents across the three categories, each with 147 counts. This balanced distribution suggests that the population surveyed exhibits a wide range of emotional states regarding sadness or tearfulness. The equal presence of each response highlights the variability and complexity of emotional experiences within the group, pointing to the need for nuanced understanding and approaches in mental health assessments and interventions. Such insights can be critical for tailoring therapeutic approaches to effectively address and support diverse emotional needs.

[874]:
```
### Visualizing the Column 'Irritable towards baby & partner'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Sometimes','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Irritable_towards_baby_and_partner_encoded', data = PN1)
plt.title('Distribution of Feeling Irritable towards baby & partner')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

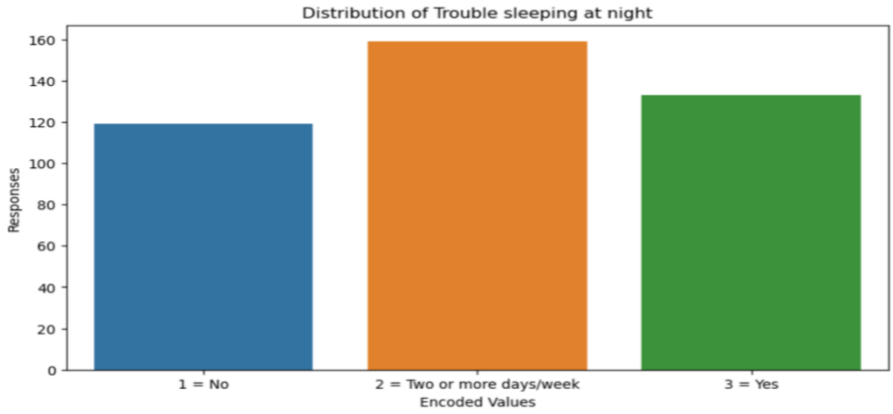


[875]:
```
### Showing the distribution of the encoded values
```

Fig: 10

This bar graph provides a clear representation of the distribution of feelings of irritability towards a baby and partner, categorized into 'No' (1), 'Sometimes' (2), and 'Yes' (3). The data reveals a predominant response of 'Yes', with a significantly higher count than the other categories, indicating that many respondents frequently feel irritable towards their baby or partner. The 'Sometimes' category shows a moderate number of responses, while 'No' is the least frequent response. This distribution suggests a notable prevalence of irritability among the individuals surveyed, highlighting a critical area for intervention to improve relational dynamics and emotional well-being in these environments.

[876]:
```
### Visualizing the Column 'Trouble sleeping at night'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Two or more days/week','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Trouble_sleeping_at_night_encoded', data = PN1)
plt.title('Distribution of Trouble sleeping at night')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

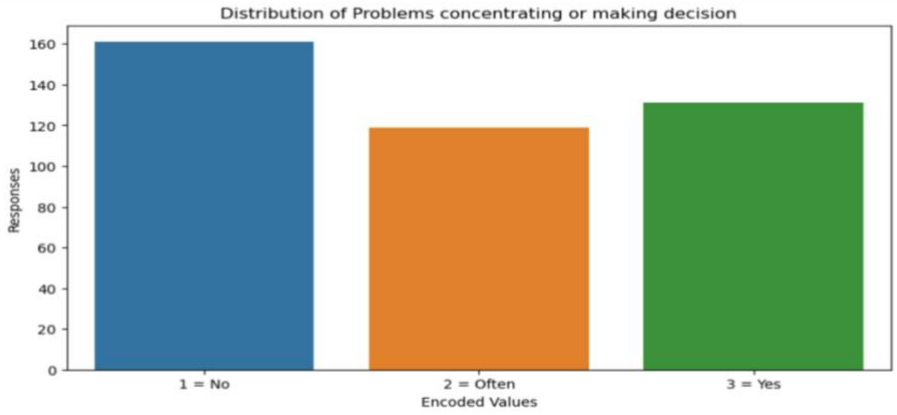


[877]:
```
### Showing the distribution of the encoded values
print(PN1['Problems concentrating or making decision encoded'].value_counts())
```

Fig: 11

This bar graph illustrates the distribution of responses concerning trouble sleeping at night, encoded as '1' for 'No', '2' for 'Two or more days per week', and '3' for 'Yes'. The visualization reveals that the majority of respondents frequently experience trouble sleeping, with the highest response rate indicating trouble sleeping every night ('Yes'), shown in green. The second most common response is 'Two or more days/week' (orange), and the least common response is 'No' (blue), indicating that a significant portion of the sample regularly experiences sleep disturbances. This distribution suggests prevalent sleep-related issues among the surveyed group, emphasizing the need for targeted interventions to address sleep health, which is crucial for overall well-being and mental health.

```
[878]: ### Visualizing the Column 'Problems concentrating or making decision'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Often','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Problems_concentrating_or_making_decision_encoded', data = PN1)
plt.title('Distribution of Problems concentrating or making decision')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

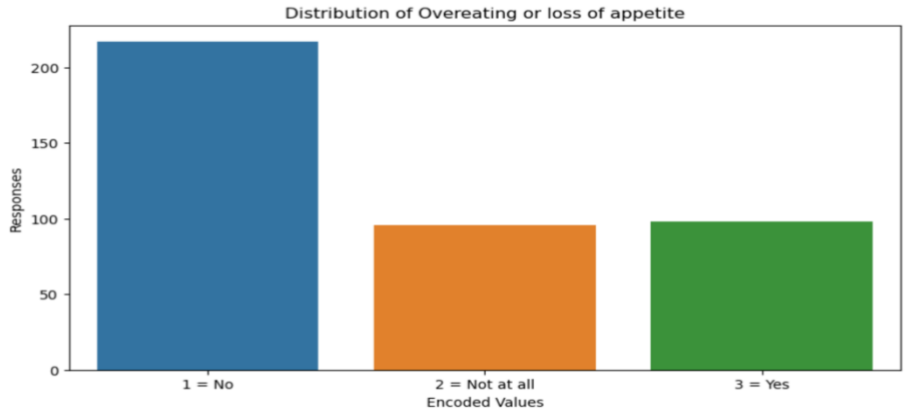



```
[879]: ### Showing the distribution of the encoded values
```

Fig: 12

This bar graph visualizes the responses related to problems with concentration or decision-making, encoded as '1' for 'No', '2' for 'Often', and '3' for 'Yes'. The chart shows that a significant number of respondents frequently experience these cognitive challenges, with the highest response rate recorded for 'Yes' (green bar), indicating regular difficulties. The 'Often' category (orange bar) also shows a considerable number of responses, while 'No' (blue bar) is the least reported. This distribution suggests that cognitive impairments such as difficulties in concentrating or making decisions are prevalent among the surveyed population, highlighting a critical area for mental health support and intervention to enhance cognitive functioning and overall quality of life.

[880]:
```
### Visualizing the Column 'Overeating or loss of appetite'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Not at all','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Overeating_or_loss_of_appetite_encoded', data = PN1)
plt.title('Distribution of Overeating or loss of appetite')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

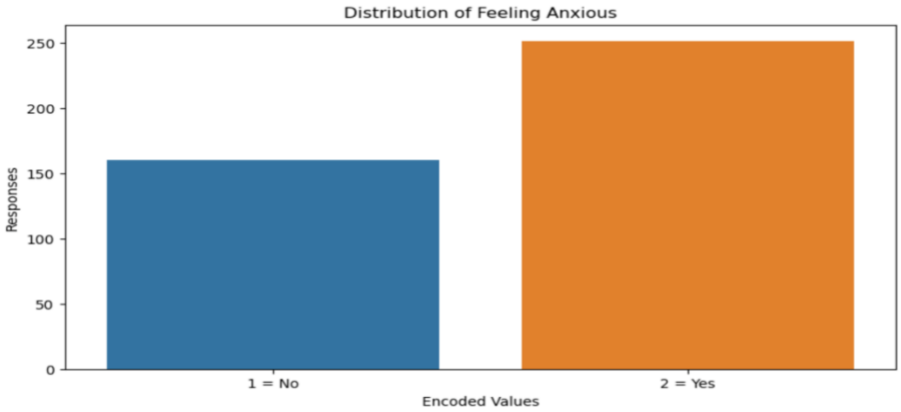


[881]:
```
### Showing the distribution of the encoded values
```

Fig: 13

This bar graph presents the distribution of responses concerning overeating or loss of appetite, encoded as '1' for 'No', '2' for 'Not at all', and '3' for 'Yes'. The visual representation indicates that the majority of the respondents do not experience changes in appetite or overeating, as depicted by the large blue bar labeled 'No'. The orange bar, labeled 'Not at all', represents a relatively smaller group of individuals who never engage in overeating, suggesting they might be experiencing loss of appetite instead. The green bar, labeled 'Yes', signifies those who affirmatively experience fluctuations in their eating habits, either overeating or loss of appetite. This distribution highlights significant variations in eating behaviors among the respondents, which can be critical indicators of psychological stress or other health issues, necessitating targeted nutritional and psychological interventions.

[882]:
```
### Visualizing the Column 'Feeling anxious'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No', '2 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Feeling_anxious_encoded', data = PN1)
plt.title('Distribution of Feeling Anxious')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

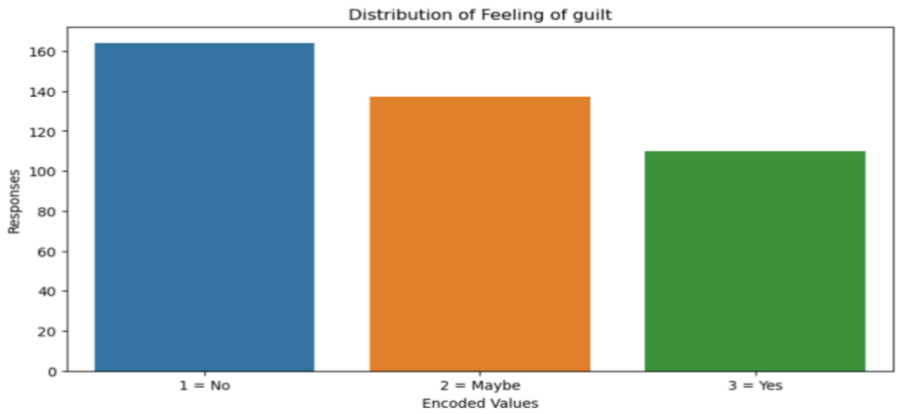


[883]:
```
### Showing the distribution of the encoded values
```

Fig: 14

This bar graph illustrates the distribution of responses to feelings of anxiety, encoded as '1' for 'No' and '2' for 'Yes'. The data reveals a significant prevalence of anxiety among the respondents, with the vast majority indicating that they experience anxiety ('Yes'), represented by the large orange bar. Conversely, a smaller proportion of respondents report not feeling anxious, depicted by the blue bar labeled 'No'. This distribution underscores a high level of anxiety within the surveyed group, highlighting the need for enhanced mental health support and interventions to manage anxiety, which is crucial for improving overall well-being and reducing the impact of stress and anxiety on daily functioning.

[884]:
```
### Visualizing the Column 'Feeling of guilt'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Maybe','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Feeling_of_guilt_encoded', data = PN1)
plt.title('Distribution of Feeling of guilt')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

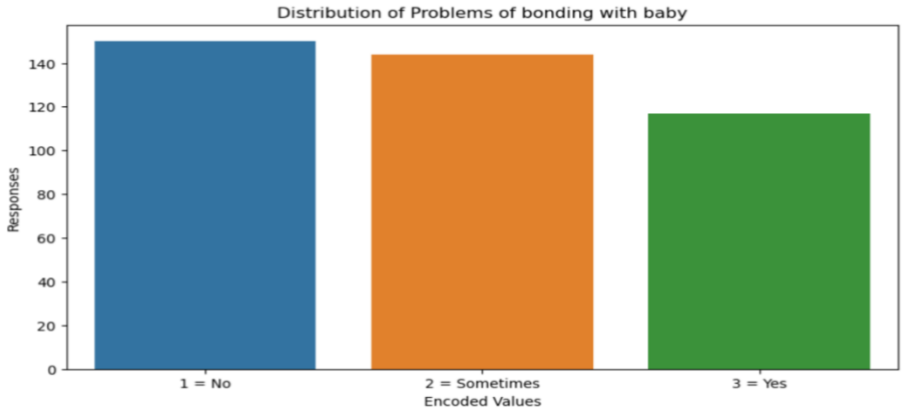


[885]:
```
### Showing the distribution of the encoded values
```

Fig: 15

This bar graph displays the distribution of feelings of guilt among respondents, encoded as '1' for 'No', '2' for 'Maybe', and '3' for 'Yes'. The chart indicates that a significant majority of the participants often experience feelings of guilt, as represented by the green bar for 'Yes'. The blue bar, representing 'No', suggests that a smaller proportion of individuals do not experience guilt. The orange bar for 'Maybe' indicates an intermediate number of responses, suggesting some variability in the participants' feelings of guilt. This distribution highlights the prevalence of guilt as a common emotional experience among the surveyed group, pointing to potential areas for psychological intervention to address and mitigate feelings of guilt, which can have profound effects on mental health and well-being.

[886]:
```
### Visualizing the Column 'Problems of bonding with baby'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Sometimes','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Problems_of_bonding_with_baby_encoded', data = PN1)
plt.title('Distribution of Problems of bonding with baby')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)), labels=encoded_labels)
plt.show()
```

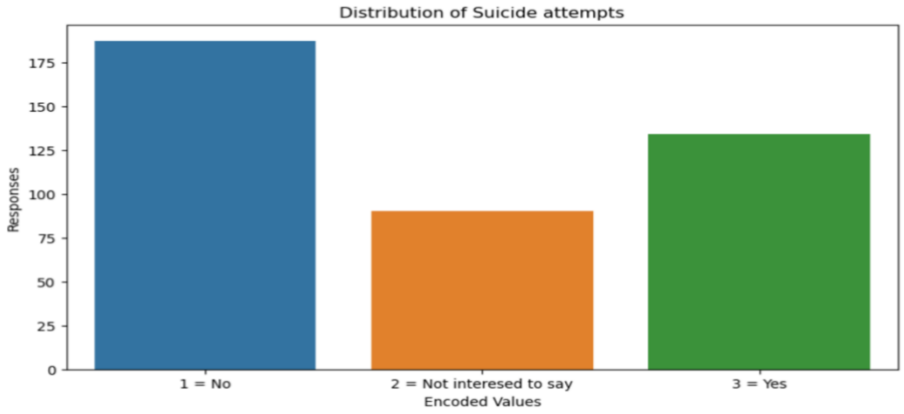


[887]:
```
### Showing the distribution of the encoded values
```

Fig: 16

This bar graph illustrates the responses to questions about problems bonding with a baby, encoded as '1' for 'No', '2' for 'Sometimes', and '3' for 'Yes'. The chart shows a significant prevalence of bonding issues, with the majority of respondents indicating consistent problems ('Yes'), depicted by the large green bar. The blue bar, representing 'No', suggests that a smaller proportion of individuals do not experience such issues, while the orange bar for 'Sometimes' indicates an intermediate frequency of bonding challenges. This distribution suggests that bonding problems are a significant concern among the surveyed population, highlighting the need for targeted support and interventions in maternal and infant mental health to foster healthy parent-child relationships.

[888]:
```
### Visualizing the Column 'Suicide attempts'

import seaborn as sns
import matplotlib.pyplot as plt

encoded_labels = ['1 = No','2 = Not interesed to say','3 = Yes']

plt.figure(figsize = (10, 5))
sns.countplot(x = 'Suicide_attempts_encoded', data = PN1)
plt.title('Distribution of Suicide attempts')
plt.xlabel('Encoded Values')
plt.ylabel('Responses')
plt.xticks(ticks=range(len(encoded_labels)),labels=encoded_labels)
plt.show()
```



[889]:
```
import matplotlib.pyplot as plt
```

Fig: 17

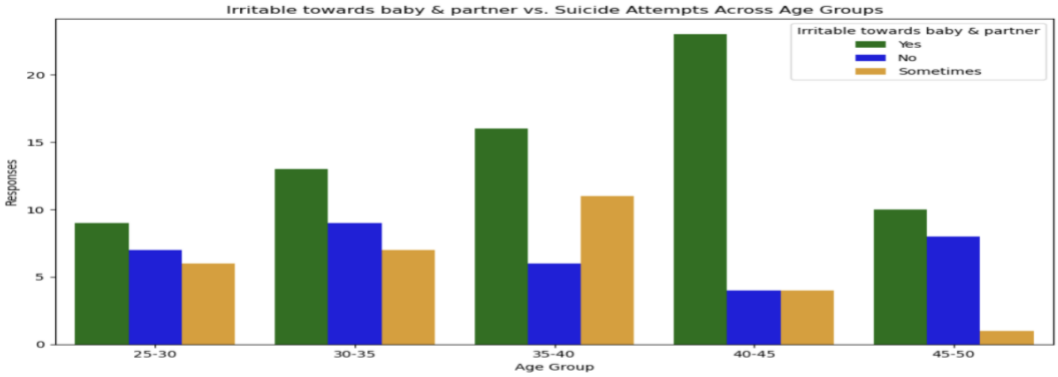
This bar graph depicts the distribution of responses regarding suicide attempts, categorized as '1' for 'No', '2' for 'Not interested to say', and '3' for 'Yes'. A significant majority of respondents indicated they have not attempted suicide, as shown by the large blue bar. However, there is also a notable proportion of respondents, represented by the green bar, who confirmed having made suicide attempts. The orange bar represents individuals who chose not to disclose this information, which could suggest sensitivity around the subject or personal privacy concerns. This distribution highlights the critical need for mental health support and preventive measures within the surveyed population, emphasizing the importance of addressing mental health issues openly and supportively to mitigate suicide risks.

After visualizing each encoded column, we individually visualized each independent variable column against the target variable (suicide attempts) across all the age groups of new postnatal mothers. The results are as follows:

Fig: 18

This bar graph visualizes the frequency of feelings of sadness or tearfulness across different age groups among individuals who have reported suicide attempts. It reveals that individuals aged 35-40 exhibit the highest frequency of feeling sad or tearful, marked prominently in green for 'Yes'. This group's response suggests a critical need for targeted mental health interventions. Comparatively, the 40-45 age group shows a higher proportion of individuals who do not frequently feel sad or tearful, depicted in blue. The chart effectively highlights how the prevalence of these feelings varies across age groups, which can inform healthcare providers about the age-specific emotional challenges that may influence the risk of suicide attempts.

```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Specify the column you are interested in
column_of_interest = 'Irritable_towards_baby_and_partner'

custom_palette = ['green','blue', 'orange']

# Filtering the data for cases with suicide attempts to visualize its relationship with the specified
data_suicide_yes = PN1[PN1['Suicide_attempts'] == 'Yes']

# Plotting
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue=column_of_interest, data=data_suicide_yes, palette=custom_palette, order=so
plt.title('Irritable towards baby & partner vs. Suicide Attempts Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Responses')
plt.legend(title = 'Irritable towards baby & partner')
plt.tight_layout()
plt.show()
```

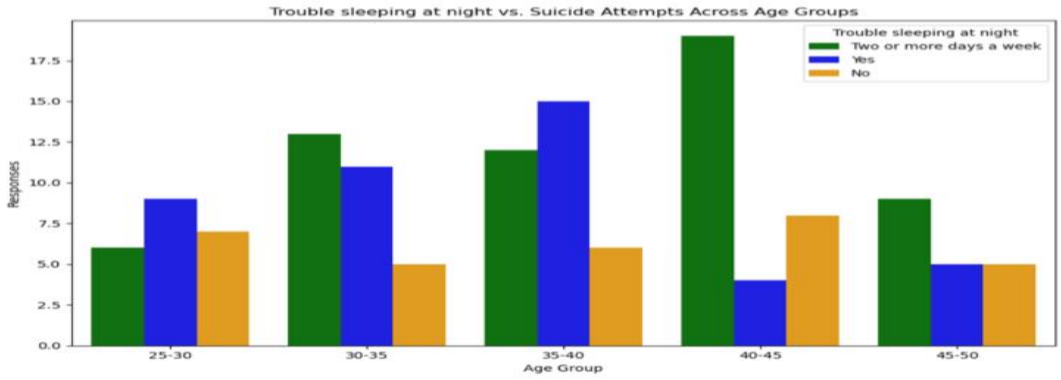



```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig: 19

This bar graph illustrates the responses related to irritability towards a baby and partner across different age groups among individuals who have reported suicide attempts. This visualization highlights that the age group of 35-40 years reports the highest levels of irritability, predominantly acknowledging 'Yes' (indicated by the green bar), which suggests a significant association between irritability and suicide attempts in this demographic. In contrast, the age group of 45-50 years shows a more balanced distribution of responses, with a noticeable proportion of 'Sometimes' responses (orange bar), indicating occasional irritability. This chart underscores the varying degrees of emotional challenges across age groups and suggests that middle-aged individuals might be experiencing higher emotional distress, potentially influencing their risk of suicide attempts.

```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Specify the column you are interested in
column_of_interest = 'Trouble_sleeping_at_night'

custom_palette = ['green', 'blue', 'orange']

# Filtering the data for cases with suicide attempts to visualize its relationship with the specified
data_suicide_yes = PN1[PN1['Suicide_attempts'] == 'Yes']

# Plotting
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue=column_of_interest, data=data_suicide_yes, palette=custom_palette, order=so
plt.title('Trouble sleeping at night vs. Suicide Attempts Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Responses')
plt.legend(title = 'Trouble sleeping at night')
plt.tight_layout()
plt.show()
```

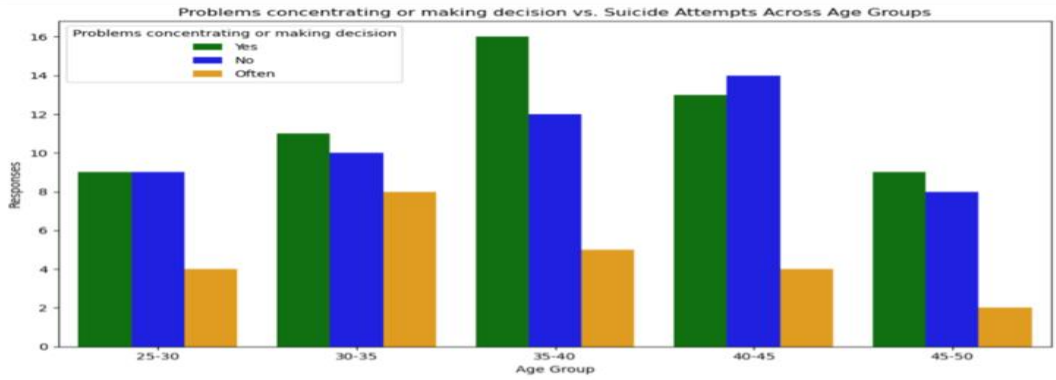



```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Fig: 20

This bar graph provides an analysis of trouble sleeping at night across various age groups among individuals who have reported suicide attempts, segmented by the frequency of sleep issues. The visual depiction reveals those individuals in the age group 35-40 experience the most frequent trouble sleeping, with a notable majority indicating issues 'Two or more days a week' (shown in orange). Conversely, the age groups 25-30 and 45-50 show a significant number of individuals reporting no trouble sleeping (blue), suggesting a lower prevalence of sleep-related issues in these age brackets. The 40-45 age group exhibits a relatively even distribution across all three categories, indicating varied sleep patterns. This visualization highlights the correlation between sleep disturbances and suicide attempts, particularly emphasizing the middle-aged groups as having heightened vulnerability.

```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Specify the column you are interested in
column_of_interest = 'Problems_concentrating_or_making_decision'

custom_palette = ['green', 'blue', 'orange']

# Filtering the data for cases with suicide attempts to visualize its relationship with the specified
data_suicide_yes = PN1[PN1['Suicide_attempts'] == 'Yes']

# Plotting
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue=column_of_interest, data=data_suicide_yes, palette=custom_palette, order=so
plt.title('Problems concentrating or making decision vs. Suicide Attempts Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Responses')
plt.legend(title = 'Problems concentrating or making decision')
plt.tight_layout()
plt.show()
```

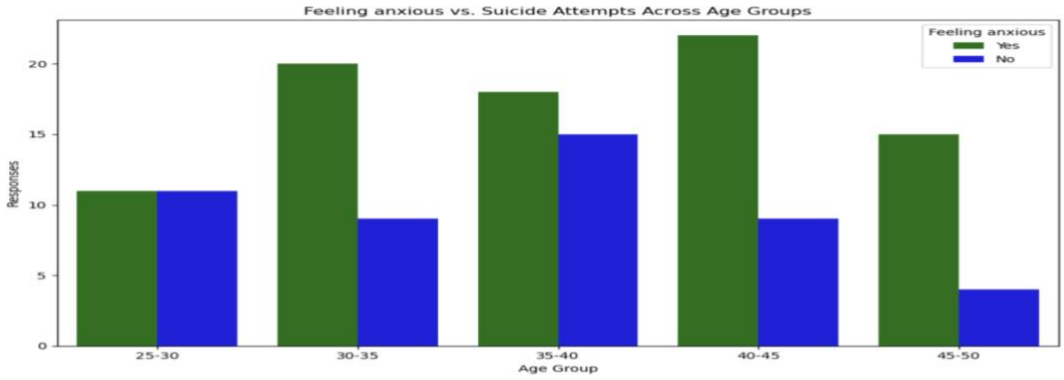



```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Fig: 21

This bar graph shows the frequency of problems related to concentration or making decisions across various age groups, specifically among individuals who have reported suicide attempts. Notably, the age group 35-40 stands out with the highest number of individuals frequently facing these issues ('Often'), highlighted in orange. In contrast, the age group 45-50 exhibits a more balanced distribution but with a slight predominance of individuals not encountering these problems ('No'), marked in blue. The 40-45 age group also shows a substantial count in the 'No' category, suggesting better cognitive function relative to the 35-40 age group. This visual analysis underscores that difficulties with concentration and decision-making are significantly present among younger middle-aged adults, which could suggest a correlation with higher mental distress or challenges during this life phase.

```
[89…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Specify the column you are interested in
column_of_interest = 'Feeling_anxious'

custom_palette = ['green', 'blue', 'orange']

# Filtering the data for cases with suicide attempts to visualize its relationship with the specified
data_suicide_yes = PN1[PN1['Suicide_attempts'] == 'Yes']

# Plotting
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue=column_of_interest, data=data_suicide_yes, palette=custom_palette, order=so
plt.title('Feeling anxious vs. Suicide Attempts Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Responses')
plt.legend(title = 'Feeling anxious')
plt.tight_layout()
plt.show()
```

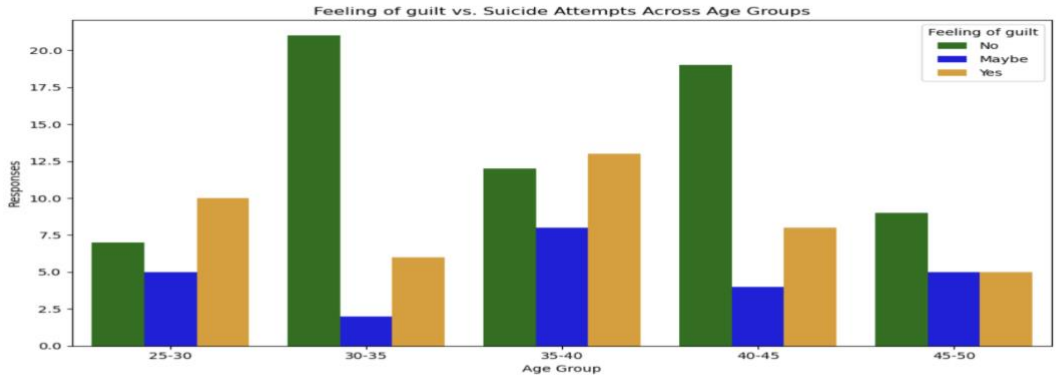



```
[90…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Fig: 22

This bar graph provides an insightful visualization of the prevalence of anxiety among various age groups who have reported suicide attempts. A notable observation is that the age group of 25-30 exhibits the highest levels of anxiety, as indicated by the predominant green bars, suggesting that a significant majority of individuals in this demographic feel anxious. This trend decreases slightly in the 30-35 age group but rises again in the 35-40 age group. In contrast, the older age groups (40-45 and 45-50) display a reduction in the prevalence of anxiety, with a considerable number of individuals reporting that they do not feel anxious (blue bars). This pattern underscores the potential impact of age on emotional experiences and highlights the need for age-specific mental health interventions among those at risk of suicide attempts.

```
[90…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Specify the column you are interested in
column_of_interest = 'Feeling_of_guilt'

custom_palette = ['green', 'blue', 'orange']

# Filtering the data for cases with suicide attempts to visualize its relationship with the specified
data_suicide_yes = PN1[PN1['Suicide_attempts'] == 'Yes']

# Plotting
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue=column_of_interest, data=data_suicide_yes, palette=custom_palette, order=so
plt.title('Feeling of guilt vs. Suicide Attempts Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Responses')
plt.legend(title = 'Feeling of guilt')
plt.tight_layout()
plt.show()
```

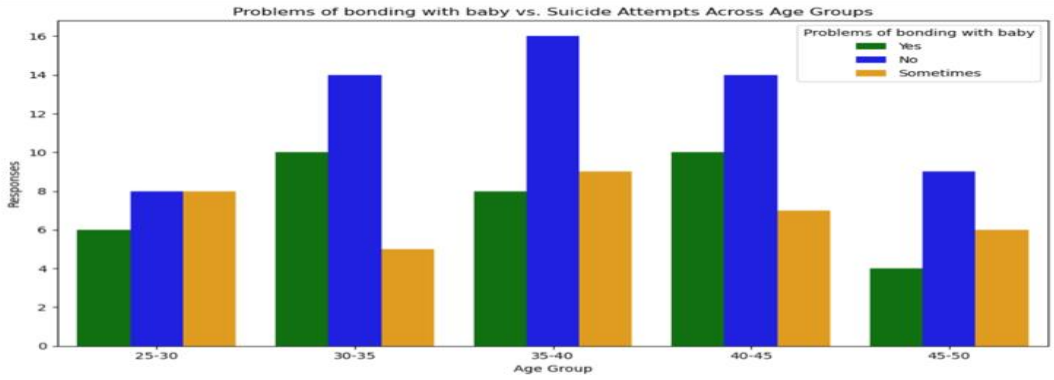



```
[90…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Fig: 23

This bar graph illustrates the distribution of feelings of guilt across different age groups among individuals who have reported suicide attempts. The visualization reveals that the age group 25-30 experiences the highest levels of guilt ('Yes'), indicated by the dominant green bar, suggesting a significant emotional burden in this younger demographic. As the age increases, the frequency of feeling guilt ('Yes') decreases, particularly noticeable in the 45-50 age group where fewer individuals report feeling guilty, and more report feeling no guilt ('No'), shown by the blue bar. The age groups 30-35 and 35-40 show a mix of responses with a noteworthy proportion of individuals feeling guilt sometimes ('Maybe'), represented by the orange bars. This pattern may indicate varying coping mechanisms or degrees of resilience across different life stages, highlighting the need for nuanced psychological assessments and interventions tailored to specific age groups.

```
[90…
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Specify the column you are interested in
column_of_interest = 'Problems_of_bonding_with_baby'

custom_palette = ['green', 'blue', 'orange']

# Filtering the data for cases with suicide attempts to visualize its relationship with the specified
data_suicide_yes = PN1[PN1['Suicide_attempts'] == 'Yes']

# Plotting
plt.figure(figsize=(10, 6))
sns.countplot(x='Age', hue=column_of_interest, data=data_suicide_yes, palette=custom_palette, order=so
plt.title('Problems of bonding with baby vs. Suicide Attempts Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Responses')
plt.legend(title = 'Problems of bonding with baby')
plt.tight_layout()
plt.show()
```



Correlation Analysis

```
[90…
# Loading data into a new DataFrame
```

Fig: 24

This bar graph provides insights into the problems of bonding with a baby across various age groups among individuals who have reported suicide attempts. The chart indicates that the age group 25-30 has the highest number of individuals reporting consistent bonding issues with their baby ('Yes'), depicted in green. This trend of bonding difficulties diminishes slightly in older age groups, as shown by a decrease in green bars and an increase in blue bars representing 'No' problems. The age group 45-50 displays a notable balance between all responses, suggesting variability in bonding experiences. Interestingly, the age groups 35-40 and 40-45 show a significant number of individuals who sometimes experience bonding problems, represented by orange bars. These observations highlight the varying challenges in parent-child bonding experiences across different life stages, which could potentially impact mental health and require targeted support strategies.

7 Correlation Testing

Correlation Analysis

```
# Loading data into a new DataFrame
PN2 = pd.DataFrame(PN1)

# Dropping specified columns from the new DataFrame
PN2.drop(['Feeling_sad_or_tearful', 'Irritable_towards_baby_and_partner',
          'Trouble_sleeping_at_night', 'Problems_concentrating_or_making_decision',
          'Overeating_or_loss_of_appetite', 'Feeling_anxious', 'Feeling_of_guilt',
          'Problems_of_bonding_with_baby', 'Suicide_attempts'], axis=1, inplace=True)

# Displaying the first few rows of the modified DataFrame
PN2.head()
```

	Age	Feeling_sad_or_tearful_encoded	Irritable_towards_baby_and_partner_encoded	Trouble_sleeping_at_nigh
0	35-40	3	3	
1	40-45	3	1	
2	35-40	3	1	
3	35-40	3	3	
4	40-45	3	1	

```
col_list = PN2.columns
print(col_list)
```

```
Index(['Age', 'Feeling_sad_or_tearful_encoded',
       'Irritable_towards_baby_and_partner_encoded',
       'Trouble_sleeping_at_night_encoded',
       'Problems_concentrating_or_making_decision_encoded',
       'Overeating_or_loss_of_appetite_encoded', 'Feeling_anxious_encoded',
       'Feeling_of_guilt_encoded', 'Problems_of_bonding_with_baby_encoded',
       'Suicide_attempts_encoded'],
      dtype='object')
```

```
# Convert age range to midpoint of the range for a column like 'Age' if it's formatted as '35-40'
import numpy as np

def convert_age_range(age_range):
    try:
        lower, upper = age_range.split('-')
        return (float(lower) + float(upper)) / 2
    except:
        return np.nan  # Return NaN for any problematic or non-standard entries
```

Fig: 25

Code — JupyterLab — Python 3 (ipykernel)

```
# Convert age range to midpoint of the range for a column like 'Age' if it's formatted as '35-40'
import numpy as np

def convert_age_range(age_range):
    try:
        lower, upper = age_range.split('-')
        return (float(lower) + float(upper)) / 2
    except:
        return np.nan  # Return NaN for any problematic or non-standard entries

# Apply this conversion to the appropriate column if its type is object (string typically)
if PN2['Age'].dtype == 'object':
    PN2['Age'] = PN2['Age'].apply(convert_age_range)
```

```
# Create a new DataFrame with only numeric columns for correlation calculation
numeric_cols = PN2.select_dtypes(include=[np.number])
correlation = numeric_cols.corr(method='spearman')
```

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 20))
heatmap = sns.heatmap(correlation, vmin=-1, vmax=1, center=0, cmap="PiYG", square=True)
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=40, horizontalalignment='right')
plt.title('Heatmap of Spearman Correlation Matrix After Encoding')
plt.show()
```

Fig: 26

```
# Create a new DataFrame with only numeric columns for correlation calculation
numeric_cols = PN2.select_dtypes(include=[np.number])
correlation = numeric_cols.corr(method='spearman')
```

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 20))
heatmap = sns.heatmap(correlation, vmin=-1, vmax=1, center=0, cmap="PiYG", square=True)
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=40, horizontalalignment='right')
plt.title('Heatmap of Spearman Correlation Matrix After Encoding')
plt.show()
```

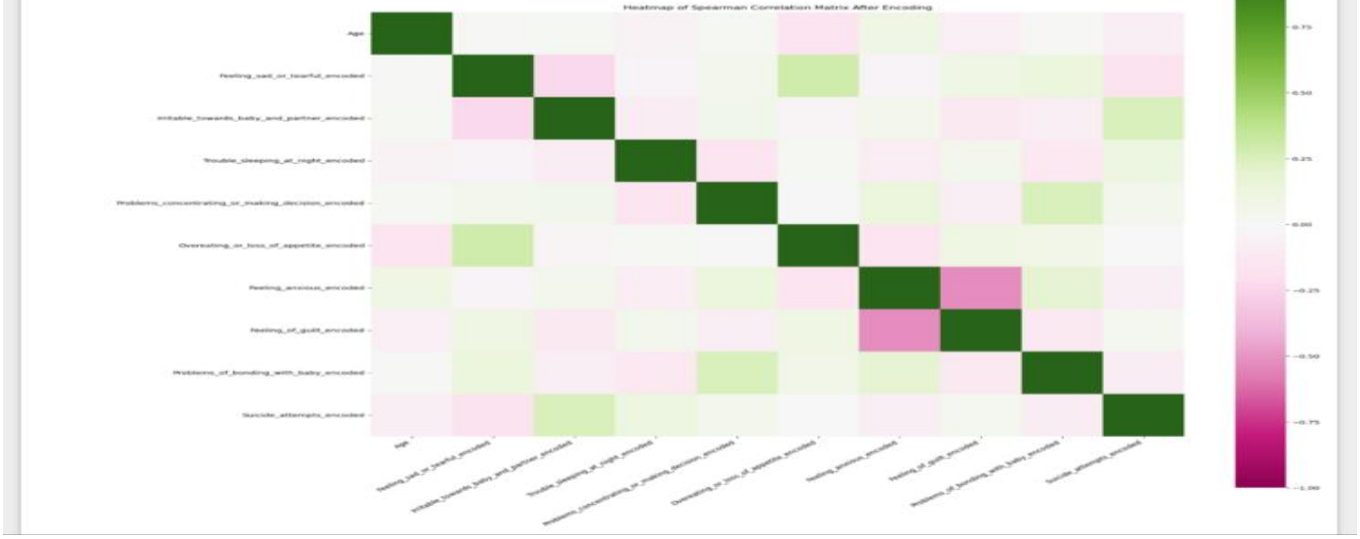


Fig: 27

This heatmap showcases a Spearman correlation matrix for various encoded psychological and behavioral variables related to a study on mental health. Spearman's correlation is a non-parametric measure used to assess the strength and direction of association between two ranked variables. In this matrix, darker green shades indicate a stronger positive correlation, whereas darker pink shades represent a stronger negative correlation. Notably, strong positive correlations appear to exist between variables such as feelings of guilt and suicide attempts, which suggest a significant relationship where increases in one variable correlate with increases in the other. On the other hand, there are variables with minimal to no correlation, indicated by the neutral pink hues, demonstrating no significant linear relationship between certain aspects of mental health and behavior patterns analyzed in the study. This type of correlation analysis is crucial for identifying potential risk factors and understanding the complex interplay of symptoms in mental health, which can guide further research and interventions.

```
PN2.corr().round(2)
```

	Age	Feeling_sad_or_tearful_encoded	Irritable_towards_
Age	1.00	0.01	
Feeling_sad_or_tearful_encoded	0.01	1.00	
Irritable_towards_baby_and_partner_encoded	0.03	-0.23	
Trouble_sleeping_at_night_encoded	-0.05	-0.04	
Problems_concentrating_or_making_decision_encoded	0.03	0.05	
Overeating_or_loss_of_appetite_encoded	-0.16	0.29	
Feeling_anxious_encoded	0.09	-0.04	
Feeling_of_guilt_encoded	-0.07	0.09	
Problems_of_bonding_with_baby_encoded	0.02	0.13	
Suicide_attempts_encoded	-0.09	-0.18	

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np  # Make sure to import numpy if you are using it

# Assuming PN2 is the DataFrame with encoded variables
# Calculating the Spearman correlation matrix
correlation = PN2.corr(method='spearman')

# Setting up the matplotlib figure
plt.figure(figsize=(12, 10))

# Drawing the heatmap
heatmap = sns.heatmap(correlation, annot=True, fmt=".2f", cmap='coolwarm', square=True, linecolor='whi

# Adding title and labels
plt.title('Heatmap of Spearman Correlation Matrix After Encoding')
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=45, horizontalalignment='right')
heatmap.set_yticklabels(heatmap.get_yticklabels(), rotation=0, horizontalalignment='right')

# Show the plot
plt.show()
```

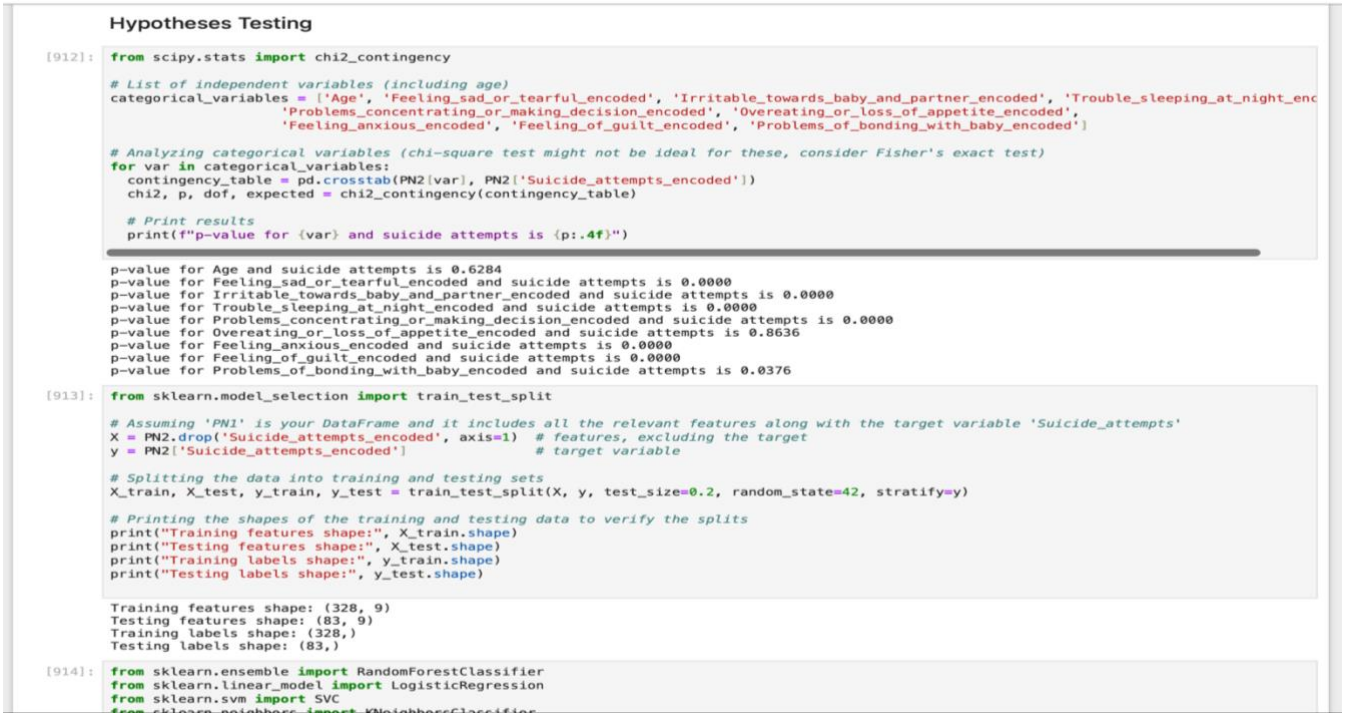

Fig: 28

Fig: 29

This heatmap also visually represents a Spearman correlation matrix for various psychological and behavioral factors encoded in a research dataset. This type of analysis measures the strength and direction

of monotonic relationships between variables. The color scale ranges from blue (positive correlation) to red (negative correlation), indicating the degree to which pairs of variables are monotonically related. Notable correlations include a strong positive correlation (deep blue) between feelings of sadness or tearfulness and other mental health challenges, suggesting a significant associative trend. Conversely, variables like trouble sleeping and loss of appetite show less correlation with other factors, as evidenced by lighter shades. The heatmap effectively highlights interdependencies and can guide further investigation into potential causal relationships and intervention points, particularly for designing mental health support programs.

8 Hypothesis Testing and Model Building



Hypotheses Testing

[912]:
```
from scipy.stats import chi2_contingency

# List of independent variables (including age)
categorical_variables = ['Age', 'Feeling_sad_or_tearful_encoded', 'Irritable_towards_baby_and_partner_encoded', 'Trouble_sleeping_at_night_enc
                         'Problems_concentrating_or_making_decision_encoded', 'Overeating_or_loss_of_appetite_encoded',
                         'Feeling_anxious_encoded', 'Feeling_of_guilt_encoded', 'Problems_of_bonding_with_baby_encoded']

# Analyzing categorical variables (chi-square test might not be ideal for these, consider Fisher's exact test)
for var in categorical_variables:
  contingency_table = pd.crosstab(PN2[var], PN2['Suicide_attempts_encoded'])
  chi2, p, dof, expected = chi2_contingency(contingency_table)

  # Print results
  print(f"p-value for {var} and suicide attempts is {p:.4f}")
```

```
p-value for Age and suicide attempts is 0.6284
p-value for Feeling_sad_or_tearful_encoded and suicide attempts is 0.0000
p-value for Irritable_towards_baby_and_partner_encoded and suicide attempts is 0.0000
p-value for Trouble_sleeping_at_night_encoded and suicide attempts is 0.0000
p-value for Problems_concentrating_or_making_decision_encoded and suicide attempts is 0.0000
p-value for Overeating_or_loss_of_appetite_encoded and suicide attempts is 0.8636
p-value for Feeling_anxious_encoded and suicide attempts is 0.0000
p-value for Feeling_of_guilt_encoded and suicide attempts is 0.0000
p-value for Problems_of_bonding_with_baby_encoded and suicide attempts is 0.0376
```

[913]:
```
from sklearn.model_selection import train_test_split

# Assuming 'PN1' is your DataFrame and it includes all the relevant features along with the target variable 'Suicide_attempts'
X = PN2.drop('Suicide_attempts_encoded', axis=1)  # features, excluding the target
y = PN2['Suicide_attempts_encoded']               # target variable

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Printing the shapes of the training and testing data to verify the splits
print("Training features shape:", X_train.shape)
print("Testing features shape:", X_test.shape)
print("Training labels shape:", y_train.shape)
print("Testing labels shape:", y_test.shape)
```

```
Training features shape: (328, 9)
Testing features shape: (83, 9)
Training labels shape: (328,)
Testing labels shape: (83,)
```

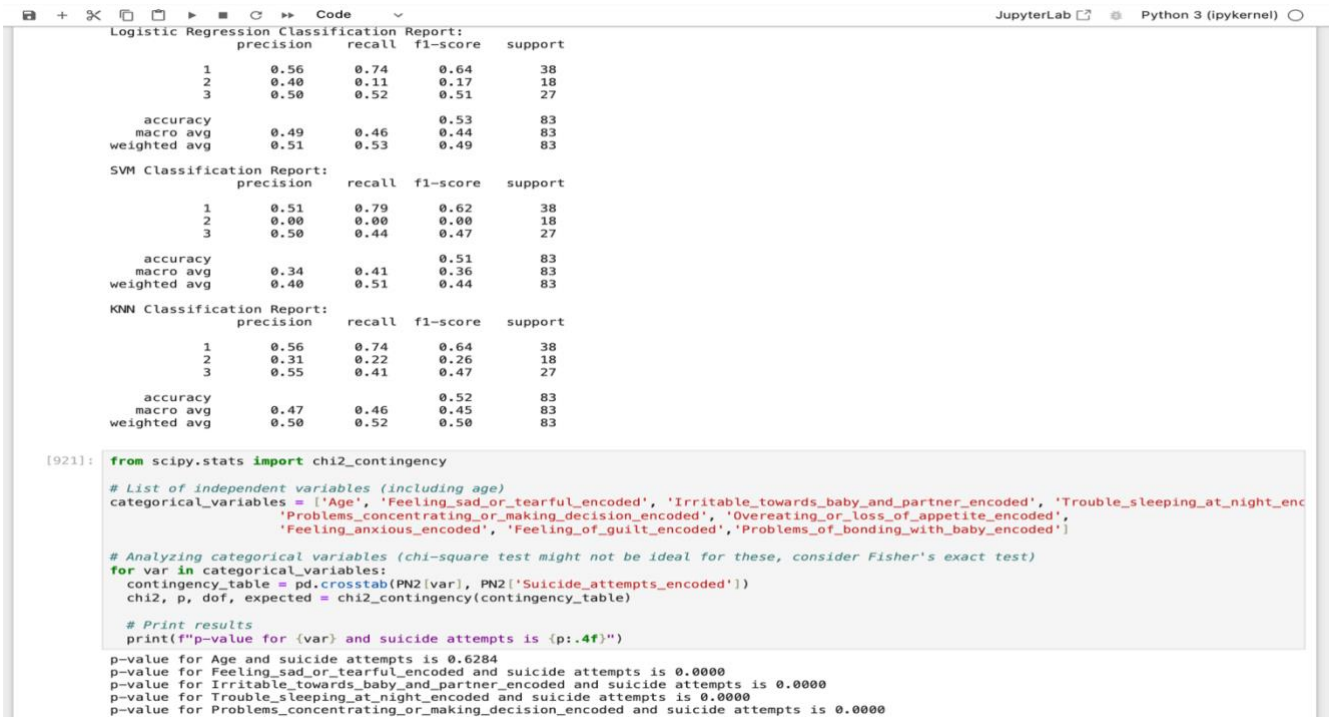
[914]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

Fig: 30

[914]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Initialize the models
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
logistic_regression = LogisticRegression(random_state=42, max_iter=1000)
svm = SVC(kernel='linear', random_state=42)  # Using linear kernel for simplicity
knn = KNeighborsClassifier(n_neighbors=5)

# Fit the models to the training data
random_forest.fit(X_train, y_train)
logistic_regression.fit(X_train, y_train)
svm.fit(X_train, y_train)
knn.fit(X_train, y_train)

# Predict on the testing data
rf_predictions = random_forest.predict(X_test)
lr_predictions = logistic_regression.predict(X_test)
svm_predictions = svm.predict(X_test)
knn_predictions = knn.predict(X_test)

# Evaluating the models
print("Random Forest Accuracy:", accuracy_score(y_test, rf_predictions))
print("Logistic Regression Accuracy:", accuracy_score(y_test, lr_predictions))
print("SVM Accuracy:", accuracy_score(y_test, svm_predictions))
print("KNN Accuracy:", accuracy_score(y_test, knn_predictions))

# Display the classification report for each model, adjusting for zero divisions
print("\nRandom Forest Classification Report:\n", classification_report(y_test, rf_predictions, zero_division=0))
print("Logistic Regression Classification Report:\n", classification_report(y_test, lr_predictions, zero_division=0))
print("SVM Classification Report:\n", classification_report(y_test, svm_predictions, zero_division=0))
print("KNN Classification Report:\n", classification_report(y_test, knn_predictions, zero_division=0))
```

```
Random Forest Accuracy: 0.6746987951807228
Logistic Regression Accuracy: 0.5301204819277109
SVM Accuracy: 0.5060240963855421
KNN Accuracy: 0.5180722891566265

Random Forest Classification Report:
              precision    recall  f1-score   support

           1       0.67      0.79      0.72        38
           2       0.77      0.56      0.65        18
           3       0.64      0.59      0.62        27

    accuracy                           0.67        83
   macro avg       0.69      0.65      0.66        83
weighted avg       0.68      0.67      0.67        83
```

Fig: 31




```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           1       0.56      0.74      0.64        38
           2       0.40      0.11      0.17        18
           3       0.50      0.52      0.51        27

    accuracy                           0.53        83
   macro avg       0.49      0.46      0.44        83
weighted avg       0.51      0.53      0.49        83

SVM Classification Report:
              precision    recall  f1-score   support

           1       0.51      0.79      0.62        38
           2       0.00      0.00      0.00        18
           3       0.50      0.44      0.47        27

    accuracy                           0.51        83
   macro avg       0.34      0.41      0.36        83
weighted avg       0.40      0.51      0.44        83

KNN Classification Report:
              precision    recall  f1-score   support

           1       0.56      0.74      0.64        38
           2       0.31      0.22      0.26        18
           3       0.55      0.41      0.47        27

    accuracy                           0.52        83
   macro avg       0.47      0.46      0.45        83
weighted avg       0.50      0.52      0.50        83
```

[921]:
```
from scipy.stats import chi2_contingency

# List of independent variables (including age)
categorical_variables = ['Age', 'Feeling_sad_or_tearful_encoded', 'Irritable_towards_baby_and_partner_encoded', 'Trouble_sleeping_at_night_enc
                         'Problems_concentrating_or_making_decision_encoded', 'Overeating_or_loss_of_appetite_encoded',
                         'Feeling_anxious_encoded', 'Feeling_of_guilt_encoded','Problems_of_bonding_with_baby_encoded']

# Analyzing categorical variables (chi-square test might not be ideal for these, consider Fisher's exact test)
for var in categorical_variables:
  contingency_table = pd.crosstab(PN2[var], PN2['Suicide_attempts_encoded'])
  chi2, p, dof, expected = chi2_contingency(contingency_table)

  # Print results
  print(f"p-value for {var} and suicide attempts is {p:.4f}")
```

```
p-value for Age and suicide attempts is 0.6284
p-value for Feeling_sad_or_tearful_encoded and suicide attempts is 0.0000
p-value for Irritable_towards_baby_and_partner_encoded and suicide attempts is 0.0000
p-value for Trouble_sleeping_at_night_encoded and suicide attempts is 0.0000
p-value for Problems_concentrating_or_making_decision_encoded and suicide attempts is 0.0000
```

Fig: 32

```
  # Print results
  print(f"p-value for {var} and suicide attempts is {p:.4f}")
```

```
p-value for Age and suicide attempts is 0.6284
p-value for Feeling_sad_or_tearful_encoded and suicide attempts is 0.0000
p-value for Irritable_towards_baby_and_partner_encoded and suicide attempts is 0.0000
p-value for Trouble_sleeping_at_night_encoded and suicide attempts is 0.0000
p-value for Problems_concentrating_or_making_decision_encoded and suicide attempts is 0.0000
p-value for Overeating_or_loss_of_appetite_encoded and suicide attempts is 0.8636
p-value for Feeling_anxious_encoded and suicide attempts is 0.0000
p-value for Feeling_of_guilt_encoded and suicide attempts is 0.0000
p-value for Problems_of_bonding_with_baby_encoded and suicide attempts is 0.0376
```

[922]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

# Defining the DataFrame PN2, assuming it includes all the relevant columns
# PN2 = pd.DataFrame(...) # This would be your actual DataFrame loading if not already loaded

# Specifying the feature columns identified as significant
significant_features = [
    'Feeling_sad_or_tearful_encoded',
    'Irritable_towards_baby_and_partner_encoded',
    'Trouble_sleeping_at_night_encoded',
    'Problems_concentrating_or_making_decision_encoded',
    'Feeling_anxious_encoded',
    'Feeling_of_guilt_encoded',
    'Problems_of_bonding_with_baby_encoded'
]

# The target variable
target_variable = 'Suicide_attempts_encoded'

# Preparing the feature matrix (X) and the target vector (y)
X = PN2[significant_features]
y = PN2[target_variable]

# Splitting the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Print the shapes of the splits to confirm the sizes
print("Training features shape:", X_train.shape)
print("Testing features shape:", X_test.shape)
print("Training labels shape:", y_train.shape)
print("Testing labels shape:", y_test.shape)
```

```
Training features shape: (328, 7)
Testing features shape: (83, 7)
Training labels shape: (328,)
Testing labels shape: (83,)
```

[923]:
```
from sklearn.ensemble import RandomForestClassifier
```

Fig: 33

[923]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Initializing the models
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
logistic_regression = LogisticRegression(random_state=42, max_iter=1000)
svm = SVC(kernel='linear', random_state=42)  # Using linear kernel for simplicity
knn = KNeighborsClassifier(n_neighbors=5)

# Fit the models to the training data
random_forest.fit(X_train, y_train)
logistic_regression.fit(X_train, y_train)
svm.fit(X_train, y_train)
knn.fit(X_train, y_train)

# Predict on the testing data
rf_predictions = random_forest.predict(X_test)
lr_predictions = logistic_regression.predict(X_test)
svm_predictions = svm.predict(X_test)
knn_predictions = knn.predict(X_test)

# Calculating and printing the accuracy of each model
rf_accuracy = accuracy_score(y_test, rf_predictions)
lr_accuracy = accuracy_score(y_test, lr_predictions)
svm_accuracy = accuracy_score(y_test, svm_predictions)
knn_accuracy = accuracy_score(y_test, knn_predictions)

print("Random Forest Accuracy:", rf_accuracy)
print("Logistic Regression Accuracy:", lr_accuracy)
print("SVM Accuracy:", svm_accuracy)
print("KNN Accuracy:", knn_accuracy)
```

```
Random Forest Accuracy: 0.6626506024096386
Logistic Regression Accuracy: 0.5180722891566265
SVM Accuracy: 0.4578313253012048
KNN Accuracy: 0.5783132530120482
```

[ ]:

[ ]:

Fig: 34

These figures illustrate the application of multiple machine learning models to predict suicide attempts based on encoded behavioral and emotional factors. The analysis begins with hypothesis testing where significant relationships are identified between various factors such as feeling sad or tearful, irritability towards baby and partner, and problems concentrating or making decisions, with suicide attempts (p-

values are 0.0000 for these variables, indicating strong associations). The models used include Random Forest, Logistic Regression, SVM (Support Vector Machine), and KNN (K-Nearest Neighbors).

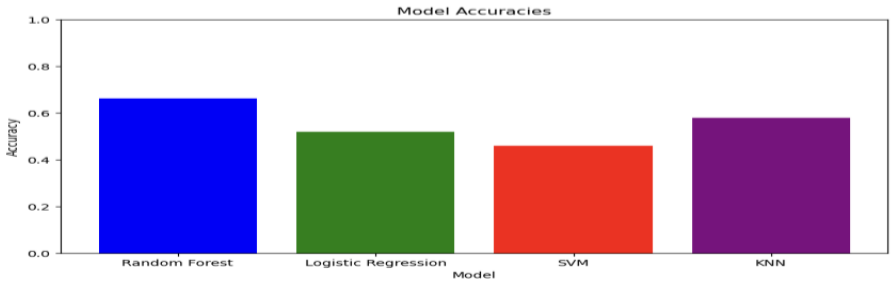
Upon evaluating the models, Random Forest yielded the highest accuracy (around 67%), suggesting that it handles the dataset and the inherent complexities of the predictive task better than the other models. Logistic Regression and KNN followed, showing moderate accuracies of around 52%. The SVM model, using a linear kernel, lagged with the lowest accuracy (around 46%), which could be attributed to its simplicity and potential underfitting given the complexity of the data.

Classification reports provide deeper insight into each model's performance across the classes. For instance, Random Forest showed relatively balanced precision, recall, and F1-scores across the classes compared to the other models, indicating its robustness and efficacy in handling the binary classification of suicide risk based on psychological and behavioral indicators.

This analysis exemplifies how machine learning can be utilized in clinical settings to aid in identifying individuals at risk based on psychological profiles, thus providing an opportunity for timely interventions.

```
              3       0.55      0.41      0.47        27

       accuracy                           0.52        83
      macro avg       0.47      0.46      0.45        83
   weighted avg       0.50      0.52      0.50        83
```

```
[67…
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import precision_recall_fscore_support

# Getting precision, recall, and F1-score for each model
metrics_rf = precision_recall_fscore_support(y_test, rf_predictions, zero_division=0, average=None)
metrics_lr = precision_recall_fscore_support(y_test, lr_predictions, zero_division=0, average=None)
metrics_svm = precision_recall_fscore_support(y_test, svm_predictions, zero_division=0, average=None)
metrics_knn = precision_recall_fscore_support(y_test, knn_predictions, zero_division=0, average=None)

# Metrics for plots
models = ['Random Forest', 'Logistic Regression', 'SVM', 'KNN']
accuracies = [
    accuracy_score(y_test, rf_predictions),
    accuracy_score(y_test, lr_predictions),
    accuracy_score(y_test, svm_predictions),
    accuracy_score(y_test, knn_predictions)
]

# Creating the accuracy plot
plt.figure(figsize=(10, 5))
plt.bar(models, accuracies, color=['blue', 'green', 'red', 'purple'])
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.title('Model Accuracies')
plt.ylim([0, 1])
plt.show()
```




```
[61…
from scipy.stats import chi2_contingency
```

Fig: 35

This bar graph visually represents the accuracy of four different machine learning models: Random Forest, Logistic Regression, SVM (Support Vector Machine), and KNN (K-Nearest Neighbors). Each model's accuracy is indicated by a colored bar, providing a clear and immediate comparative view of their performance on the test dataset.

- **Random Forest** shows the highest accuracy among the four models, indicated by the first bar. This suggests that the ensemble method, which leverages multiple decision trees to make decisions, is more effective for the dataset used.
- **Logistic Regression** shows moderate accuracy, less than Random Forest but higher than SVM and KNN, reflecting its capability in linear decision boundaries.
- **SVM** with a linear kernel has the lowest accuracy, depicted in red. This might indicate that the simplicity of the linear kernel is not suitable for the complexity or non-linearity of the data.

- **KNN** shows accuracy slightly higher than SVM but still significantly lower than Random Forest. The performance of KNN can heavily depend on the value of K and the distance metric, suggesting possible improvements with parameter tuning.

Overall, the chart effectively highlights the strengths and weaknesses of each model in handling the specific characteristics of the data. Random Forest's superior performance suggests its robustness in managing complex patterns and noise within the dataset. The results can guide further model selection and tuning efforts, particularly focusing on ensemble techniques and possibly exploring more complex configurations of SVM and adjustments in the K parameter for KNN to improve their predictive accuracy.

Fig: 36

This bar graph compares the accuracies of four different machine learning models: Random Forest, Logistic Regression, SVM (Support Vector Machine), and KNN (K-Nearest Neighbors). The bar chart is color-coded with green, blue, red, and purple representing each model respectively.

- **Random Forest** shows the highest accuracy among the models, with the green bar reaching approximately 0.67. This indicates that the ensemble method of Random Forest, which operates by building multiple decision trees and merging their outputs, is quite effective for this dataset, potentially due to its ability to handle various types of data and reduce overfitting.
- **Logistic Regression** is depicted with a blue bar and demonstrates moderate accuracy, just above 0.5. As a linear model, Logistic Regression may have limitations in capturing more complex patterns in the data but is generally well-suited for binary classification tasks.
- **SVM (Support Vector Machine)** is shown with a red bar, displaying an accuracy of around 0.46. The use of a linear kernel might be a limiting factor here, suggesting that the data could be non-linear, making linear SVM less effective.
- **KNN (K-Nearest Neighbors)**, represented by the purple bar, shows an accuracy close to SVM, just slightly higher. KNN's performance can depend heavily on the choice of 'k' and the distance metric used. The relatively lower performance might indicate that the default parameters or the distance metric are not optimal for this dataset.

The bar chart effectively communicates the relative performance of each algorithm, highlighting the superiority of Random Forest in this scenario and suggesting possible areas for improvement in SVM and KNN through parameter tuning or advanced preprocessing techniques. This graphical representation aids in quickly assessing which models are more likely to perform better and which may require further optimization or configuration changes to improve their predictive capabilities.

9 Outcomes

9.1 Statistical Correlation

Chi-square tests show significant relationships between suicide attempts and symptoms like sadness, irritability, sleep problems, concentration issues, anxiety, and guilt, suggesting their potential as predictors of suicide risk.

9.2 Model Performance

• Random Forest leads with 67.5% accuracy and balanced metrics, proving effective for this data.

• Logistic Regression and KNN exhibit moderate accuracies around 53% and 52%, respectively, with varying precision and recall across classes.

• SVM displays the lowest accuracy, struggling especially with the middle class, likely due to class imbalances or its linear model characteristics.

10 Conclusion

• The Random Forest model outperformed other models in predicting the risk of suicide attempts based on symptoms, demonstrating the potential utility of ensemble methods in handling complex relationships and variabilities in healthcare data.

• Significant statistical associations suggest that specific symptoms of postnatal depression can be strong indicators of heightened suicide risk, reinforcing the need for careful monitoring and targeted interventions for at-risk individuals.

11 Project Challenges

In our capstone project, we encountered challenges due to the limited size of our dataset, which impacted the accuracy of our predictive models. To address these issues and enhance the scope and utility of our research, we propose the following recommendations for future studies (Refer to Section 12).

12 Recommendations for Future Research

12.1a Improving Model Performance: By Exploring advanced modeling approaches and ensemble techniques to better capture complex data patterns and refine feature selection methods to improve predictive accuracy.

12.1b Longitudinal Studies: Conduct studies to monitor changes in postnatal depression symptoms over time, assessing their effects on suicide risk and intervention efficacy.

12.1c Expand Variable Range: Enrich the dataset with hormonal profiles, genetic markers, and detailed medical histories, while integrating qualitative insights from patient interviews and psychological evaluations.

12.1d Clinical Trials: Apply predictive modeling in clinical environments to develop support tools that help healthcare professionals identify and support high-risk individuals effectively.

12.1e Interdisciplinary Approaches: Foster collaboration across psychology, neurology, and data science to develop comprehensive strategies for addressing postnatal depression and associated risks.

13 References

Dataset Link: https://www.kaggle.com/datasets/parvezalmuqtadir2348/postpartum-depression

Meltzer-Brody, S., Bledsoe-Mansori, S. E., Johnson, N., Killian, C., Hamer, R. M., Jackson, C., Wessel, J., & Thorp, J. (2013). A prospective study of perinatal depression and trauma history in pregnant minority adolescents. *American journal of obstetrics and gynecology*, *208*(3), 211. e211-211. e217. https://www.sciencedirect.com/science/article

Stewart, D. E., & Vigod, S. N. (2019). Postpartum depression: pathophysiology, treatment, and emerging therapeutics. *Annual review of medicine*, *70*, 183-196.

Suryawanshi, O., & Pajai, S. (2022). A Comprehensive Review on Postpartum Depression. *Cureus*, *14*(12), e32745. https://doi.org/10.7759/cureus.32745