BAHRAIN POLYTECHNIC بوليتكنك البحرين

Assessment Cover Sheet

Assessment	Project (Individual)		
Assessment	Uncontrolled	Individual	Not must-pass
Due Date	12-Dec-2024	**Course Code**	IT9130
Course Title	Datavisualization		
Internal Moderator's	Dr. Shomona Gracia Jacob		
External Examiner's			

Instructions:

1. This cover sheet must be completed and attached to your assessment before submission in soft copy.
2. The time allowed for this assessment is given in Moodle.
3. This assessment carries 50 marks distributed at the 3rd week of semester assessing CILO 1 and CILO 2.
4. The materials allowed for use in this assessment are Moodle, Internet resources.
5. The **use of generative AI tools is strictly prohibited.**
6. References consulted (if any) must be properly acknowledged and cited.
7. The assessment has a total of 6 pages.

Learner ID	202410231	**Date Submitted**	12/11/2024
Learner Name	Rawan Alqasimi		
Programme	ICT9010		
Programme	MSc in AI		
Lecturer's	D.Shomona Gracia Jacob		

By submitting this assessment for marking, I affirm that this assessment is my own work.

Learner	

Do not write beyond this line. For assessor use only

Assessor's Name	D.Shomona Gracia Jacob		
Marking Date		**Maks Obtained**	

Comments:

Table of Contents

Introduction to the Datasets- choice of data sets

Dataset 1: Water Quality

Problem Explanation: Water quality is a critical global issue that affects human health, ecosystems, and economic activities. Contaminated water can cause severe health problems and harm the environment. Monitoring and improving water quality is essential for sustainable development, ensuring safe drinking water, protecting ecosystems, and supporting economic activities like agriculture and tourism. Prioritizing water quality helps address global challenges and promotes a healthier, more sustainable future for all.

Attributes: The water quality dataset includes the following attributes:

- **pH:** Indicates the acidity or alkalinity of water.
- **Hardness:** Measures the concentration of calcium and magnesium ions.
- **Solids (Total Dissolved Solids - TDS):** Represents the concentration of dissolved substances in water.
- **Chloramines:** Measures the concentration of chloramines used for disinfection.
- **Sulfate:** Indicates the concentration of sulfate ions.
- **Conductivity:** Measures the water's ability to conduct electricity, related to ion concentration.
- **Organic Carbon:** Represents the amount of organic compounds in water.
- **Trihalomethanes:** Measures the concentration of trihalomethanes, which are by-products of chlorination.
- **Turbidity:** Measures the clarity of water.
- **Potability:** Indicates whether water is safe to drink (1 for potable, 0 for non-potable).

Justification for Choice: This dataset is chosen due to its global relevance and the critical need for clean water. It provides a comprehensive view of water quality parameters, making it suitable for machine learning applications. These applications can predict and classify water quality, identify pollution sources, and suggest remediation measures. This dataset supports efforts to ensure safe drinking water and protect ecosystems, contributing to sustainable development.

Reference: https://www.kaggle.com/datasets/adityakadiwal/water-potability

Dataset 2: School Bullying

Problem Explanation: Bullying in schools is a pervasive issue that affects students' mental health, academic performance, and overall well-being. Understanding the factors contributing to bullying can help in developing effective prevention and intervention strategies.

Attributes: The school bullying dataset includes:

- **Bullied on School Property in Past 12 Months:** Captures the prevalence of bullying incidents within the school environment, crucial for understanding school safety.
- **Bullied Not on School Property in Past 12 Months:** Identifies bullying incidents outside school premises, providing a broader perspective on students' experiences.
- **Cyber Bullied in Past 12 Months:** Captures the extent of online bullying, which has become significant with the rise of digital communication.
- **Custom Age:** influences the likelihood of bullying, helping analyze age-specific trends.
- **Sex:** understand gender-specific differences.
- **Physically Attacked:** Captures instances of physical violence, a severe form of bullying with significant impacts.
- **Physical Fighting:** Involvement in physical fights can be both a cause and consequence of bullying, indicating aggressive behaviors.
- **Felt Lonely:** Feelings of loneliness can result from and contribute to bullying, indicating emotional impact.
- **Close Friends:** The number of close friends influences vulnerability to bullying and coping ability, indicating social support.
- **Miss School No Permission:** Skipping school without permission can indicate bullying, as students may avoid school to escape it.
- **Other Students Kind and Helpful:** Perception of peer support influences school climate and bullying prevalence.
- **Parents Understand Problems:** Parental support is crucial for students dealing with bullying, indicating the level of parental involvement.
- **Most of the Time or Always Felt Lonely:** Chronic loneliness can indicate severe emotional distress, often associated with prolonged bullying.
- **Missed Classes or School Without Permission:** Similar to "Miss School No Permission," helps identify truancy patterns linked to bullying.
- **Were Underweight:** Physical appearance, including being underweight, can be a factor in bullying, indicating the role of body image.
- **Were Overweight:** Being overweight can make students target for bullying, indicating the impact of body weight.
- **Were Obese:** Obesity can also be a factor in bullying, providing further granularity in understanding body weight's influence.

Justification for Choice: This dataset is chosen because it addresses a significant social issue. Analyzing this data can help in identifying patterns and risk factors associated with bullying, which can inform policies and programs aimed at reducing bullying in schools.

Reference: https://www.kaggle.com/datasets/leomartinelli/bullying-in-schools

Choice of dataset: School Bullying

I selected this dataset to work on for this project because it deals with an important issue: bullying in schools. The data helps us understand how common bullying is, what types of bullying occur, and what effects it has on students. It includes detailed information about different aspects of bullying, which can help us create better ways to prevent and stop bullying, making schools safer and more supportive places for students.

Problem Statement Formulation and definition

Problem Statement: Bullying in schools is a widespread issue affecting students' mental health, academic performance, and well-being.

Objective: Use data-driven approaches to understand bullying patterns, identify risk factors, and develop prevention and intervention strategies to create safer school environments.

Significance: The research aims to reduce bullying, support affected students, and enhance school safety by leveraging insights from the dataset. This contributes to a positive and inclusive educational environment.

Data Preprocessing/Transformation- Data Cleaning: Handling Missing Values

Description:

Deleting records with null values is a common data cleaning technique when dealing with missing data. This approach helps to ensure that the analysis or model training is not affected by incomplete data. While this method is straightforward, it's important to consider the impact of removing data. If the dataset is large and the null values are sparsely distributed, deleting these records might not significantly impact the dataset's integrity. However, if null values are prevalent or the dataset is small, this could lead to the loss of valuable information. I chose to handle the null values in my dataset because there are a lot of them, however it won't affect the result because the dataset is huge.

The reason behind choosing this type of data cleaning:

- **Data Quality**: I want to make sure my dataset is complete and reliable.
- **Avoiding Bias**: By addressing the numerous missing data points, I prevent my analysis from being skewed.
- **Model Performance**: Many algorithms I use require a complete dataset to function properly.
- **Consistent Analysis**: Handling these null values helps me generate meaningful insights.

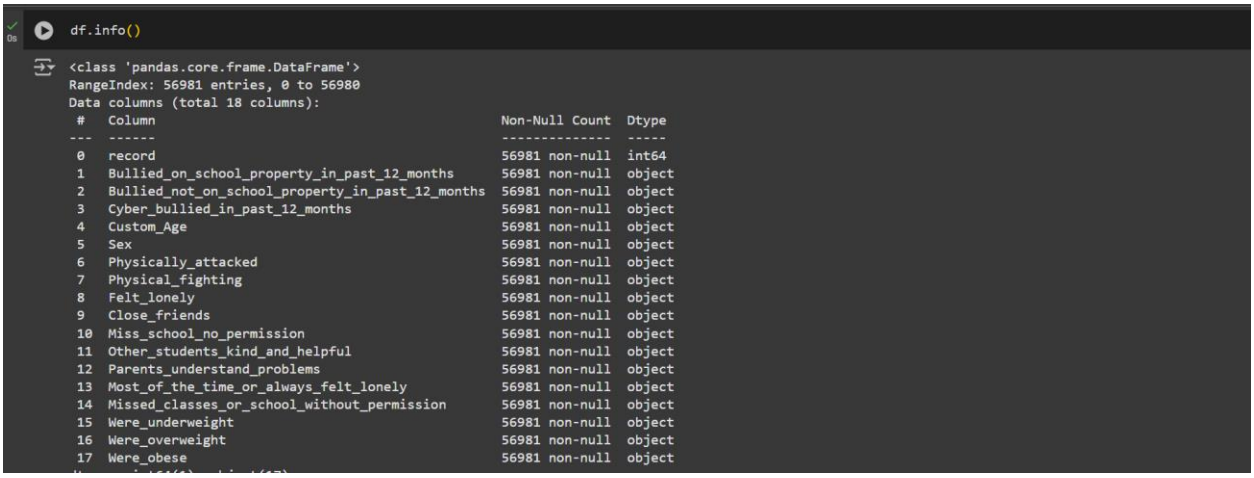
By addressing these null values, I ensure my analysis remains accurate and effective.

Handling missing values implementation.

```
[5] import pandas as pd
    import numpy as np
    from ipykernel import kernelapp as app
    import matplotlib.pyplot as plt
```

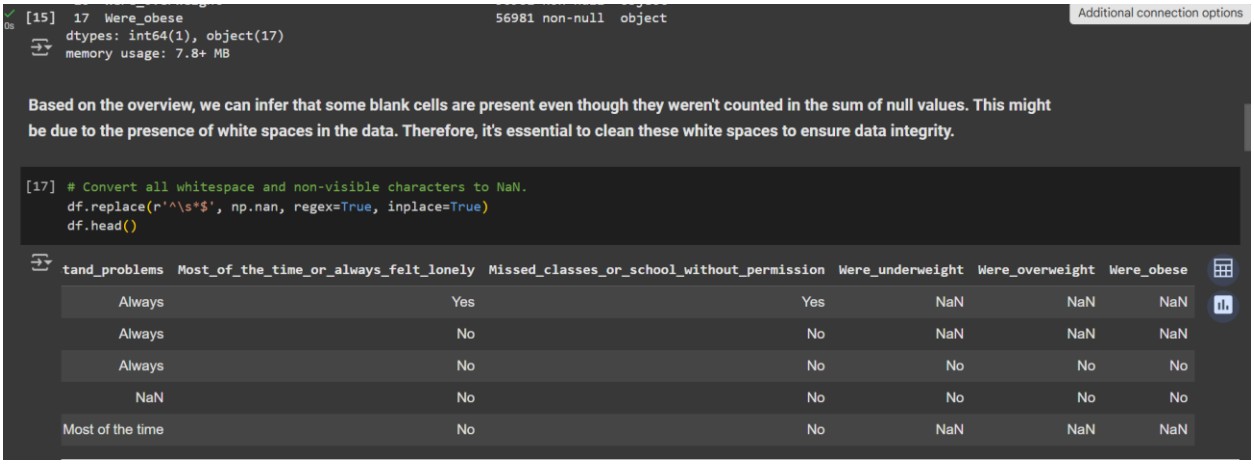
```
[14] #import the dataset and list the first 5 rows.
     df=pd.read_csv("/content/drive/MyDrive/DataVisualisation/project/bullyingInSchool.csv")
     df.head()
```

tand_problems	Most_of_the_time_or_always_felt_lonely	Missed_classes_or_school_without_permission	Were_underweight	Were_overweight	Were_obese
Always	Yes	Yes			
Always	No	No			
Always	No	No	No	No	No
	No	No	No	No	No
Most of the time	No	No			



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56981 entries, 0 to 56980
Data columns (total 18 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   record                                      56981 non-null  int64
 1   Bullied_on_school_property_in_past_12_months 56981 non-null  object
 2   Bullied_not_on_school_property_in_past_12_months 56981 non-null  object
 3   Cyber_bullied_in_past_12_months             56981 non-null  object
 4   Custom_Age                                  56981 non-null  object
 5   Sex                                         56981 non-null  object
 6   Physically_attacked                         56981 non-null  object
 7   Physical_fighting                           56981 non-null  object
 8   Felt_lonely                                 56981 non-null  object
 9   Close_friends                               56981 non-null  object
 10  Miss_school_no_permission                   56981 non-null  object
 11  Other_students_kind_and_helpful             56981 non-null  object
 12  Parents_understand_problems                 56981 non-null  object
 13  Most_of_the_time_or_always_felt_lonely      56981 non-null  object
 14  Missed_classes_or_school_without_permission 56981 non-null  object
 15  Were_underweight                            56981 non-null  object
 16  Were_overweight                             56981 non-null  object
 17  Were_obese                                  56981 non-null  object
```



```
[15]  17  Were_obese                                  56981 non-null  object
dtypes: int64(1), object(17)
memory usage: 7.8+ MB
```

Additional connection options

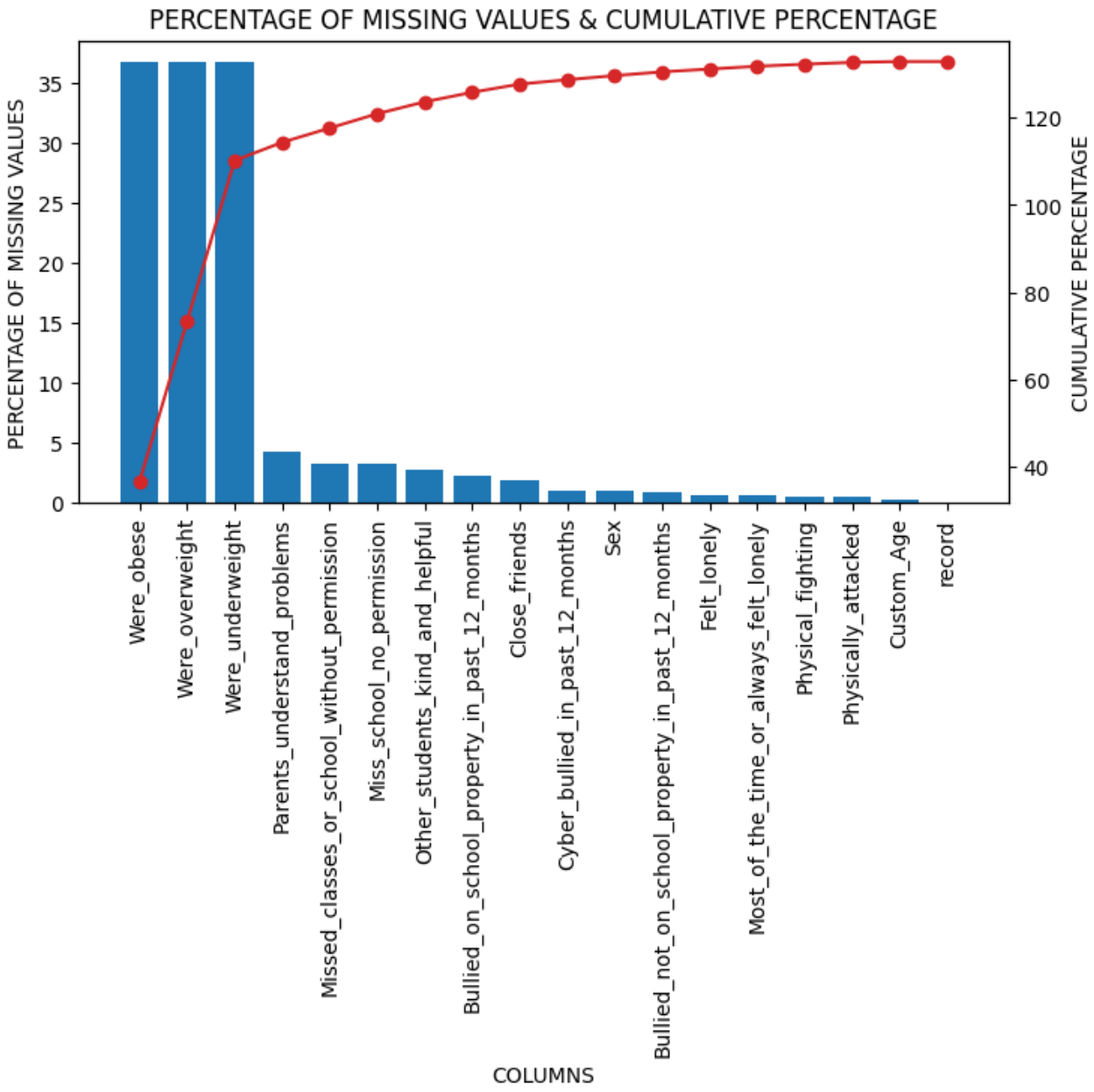
Based on the overview, we can infer that some blank cells are present even though they weren't counted in the sum of null values. This might be due to the presence of white spaces in the data. Therefore, it's essential to clean these white spaces to ensure data integrity.

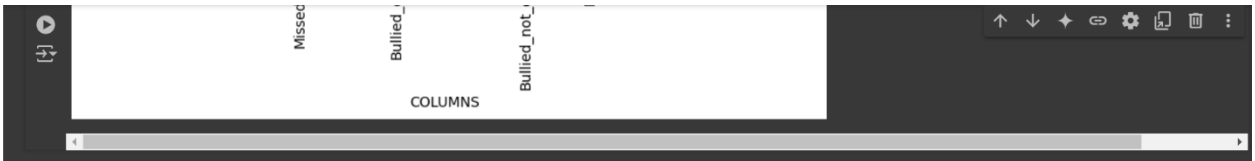
```
[17] # Convert all whitespace and non-visible characters to NaN.
     df.replace(r'^\s*$', np.nan, regex=True, inplace=True)
     df.head()
```

tand_problems	Most_of_the_time_or_always_felt_lonely	Missed_classes_or_school_without_permission	Were_underweight	Were_overweight	Were_obese
Always	Yes	Yes	NaN	NaN	NaN
Always	No	No	NaN	NaN	NaN
Always	No	No	No	No	No
NaN	No	No	No	No	No
Most of the time	No	No	NaN	NaN	NaN

```python
# Calculate the percentage of missing values in each column
missing_perc = (df.isnull().sum() / len(df)) * 100
# Sort the columns by percentage of missing values in descending order
missing_perc_sorted = missing_perc.sort_values(ascending=False)
# Calculate the cumulative percentage of missing values
cumulative_perc = missing_perc_sorted.cumsum()
# Create a Pareto chart
fig, ax1 = plt.subplots(figsize=(8, 4))
ax1.bar(missing_perc_sorted.index, missing_perc_sorted.values)
ax1.set_xlabel('COLUMNS')
ax1.set_ylabel('PERCENTAGE OF MISSING VALUES')
ax1.set_xticklabels(missing_perc_sorted.index, rotation=90)
ax2 = ax1.twinx()
ax2.plot(missing_perc_sorted.index, cumulative_perc.values, color='tab:red', marker='o')
ax2.set_ylabel('CUMULATIVE PERCENTAGE')
plt.title('PERCENTAGE OF MISSING VALUES & CUMULATIVE PERCENTAGE')
# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-29-9757f8355a31>:18: UserWarning: set_ticklabels() should only be used with a fixed number of ticks. i.e. after set_ticks() or using a
  ax1.set_xticklabels(missing_perc_sorted.index, rotation=90)
```


PERCENTAGE OF MISSING VALUES & CUMULATIVE PERCENTAGE
PERCENTAGE OF MISSING VALUES
CUMULATIVE PERCENTAGE
COLUMNS
0
5
10
15
20
25
30
35
40
60
80
100
120
Were_obese
Were_overweight
Were_underweight
Parents_understand_problems
Missed_classes_or_school_without_permission
Miss_school_no_permission
Other_students_kind_and_helpful
Bullied_on_school_property_in_past_12_months
Close_friends
Cyber_bullied_in_past_12_months
Sex
Bullied_not_on_school_property_in_past_12_months
Felt_lonely
Most_of_the_time_or_always_felt_lonely
Physical_fighting
Physically_attacked
Custom_Age
record



Were_underweight, Were_overweight, and Were_obese have the highest number of missing values at over 35%. Although imputing is possible, it could skew the model, so I'm dropping them.

```
# As discussed, drop columns with a high proportion of missing values
# Get the actual column names from the DataFrame
columns_to_drop = ['Were_underweight', 'Were_overweight', 'Were_obese']
existing_columns = df.columns.tolist()
# Filter the columns to drop to only those that exist in the DataFrame
valid_columns_to_drop = [col for col in columns_to_drop if col in existing_columns]
# Drop the valid columns
df.drop(valid_columns_to_drop, axis=1, inplace=True)
#dropping Na values
df = df.dropna()
```

[36]
```
# Get the count of non-null values for each column and check if all columns have the same count of non-null values
non_null_counts = df.count()
if non_null_counts.nunique() == 1:
    print("Count of null values:", df.isnull().sum().sum())
else:
    print("Columns have different counts of non-null values.")
```

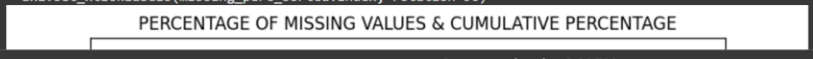
```
Count of null values: 0
```

```
# Save the DataFrame to a new CSV file and list the new, cleaned data set.
df.to_csv('/content/drive/MyDrive/DataVisualisation/project/bullyingInSchool_cleaned.csv', index=False)
df=pd.read_csv("/content/drive/MyDrive/DataVisualisation/project/bullyingInSchool_cleaned.csv")
df.head()
```

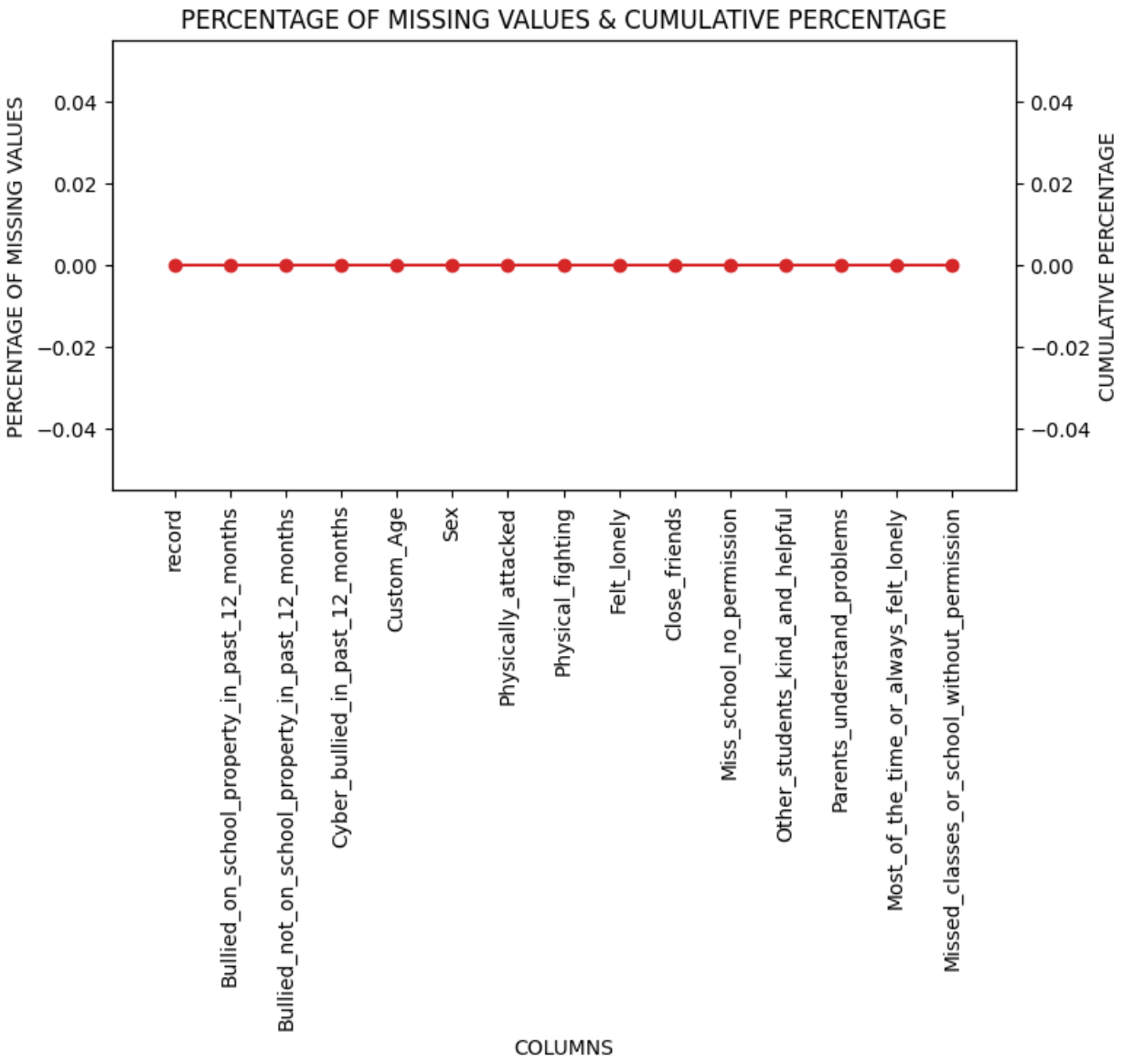
Other_students_kind_and_helpful	Parents_understand_problems	Most_of_the_time_or_always_felt_lonely	Missed_classes_or_school_without_permission
Sometimes	Always	No	No
Sometimes	Always	No	No
Most of the time	Most of the time	No	No
Most of the time	Always	No	No
Most of the time	Always	No	No

```
# Calculate the percentage of missing values in each column
missing_perc = (df.isnull().sum() / len(df)) * 100
# Sort the columns by percentage of missing values in descending order
missing_perc_sorted = missing_perc.sort_values(ascending=False)
# Calculate the cumulative percentage of missing values
cumulative_perc = missing_perc_sorted.cumsum()
# Create a Pareto chart
fig, ax1 = plt.subplots(figsize=(8, 4))
ax1.bar(missing_perc_sorted.index, missing_perc_sorted.values)
ax1.set_xlabel('COLUMNS')
ax1.set_ylabel('PERCENTAGE OF MISSING VALUES')
ax1.set_xticklabels(missing_perc_sorted.index, rotation=90)
ax2 = ax1.twinx()
ax2.plot(missing_perc_sorted.index, cumulative_perc.values, color='tab:red', marker='o')
ax2.set_ylabel('CUMULATIVE PERCENTAGE')
plt.title('PERCENTAGE OF MISSING VALUES & CUMULATIVE PERCENTAGE')
# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-58-689269f81004>:12: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a
  ax1.set_xticklabels(missing_perc_sorted.index, rotation=90)
```



✓ 0s completed at 9:00 PM



The data has now undergone meticulous cleaning, ensuring that all records with missing values have been thoroughly removed. This critical step guarantees that our dataset is not only complete but also primed for precise analysis and reliable modeling. By systematically eliminating any incomplete records, we have significantly enhanced the overall quality and dependability of our data. This comprehensive preparation is vital for deriving accurate and meaningful insights from subsequent analyses and models. Moreover, our dataset is now fully prepared for machine learning applications, providing a robust foundation for training algorithms and achieving credible, actionable results.

Choice of data visualization tools

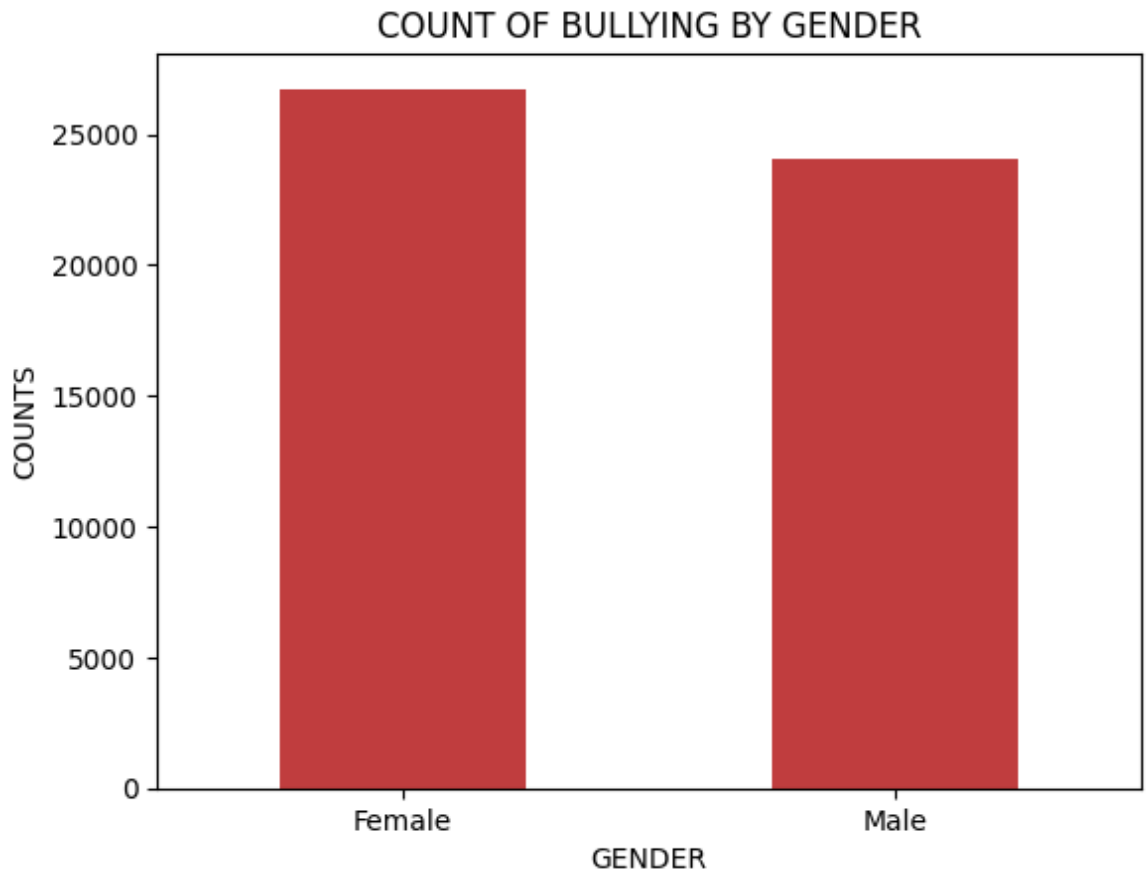
Data Visualization tool 1: Countplot

Packages:

- seaborn (imported as sns)
- matplotlib. pyplot (imported as plt)
- pandas (imported as pd)

Reason: These libraries together provide a comprehensive toolkit for visualizing and analyzing data efficiently. Seaborn library is great for creating statistical data visualizations, including the countplot function. And the matplotlib Provides functions for customizing plots, such as setting labels, titles, legends, and displaying the plot. Finally, pandas library used for data manipulation and analysis, including reading data from CSV files using the read_csv function.

Plot:



The bar graph titled **"COUNT OF BULLYING BY GENDER"** shows the number of bullying incidents categorized by gender.

- **X-axis (Gender)**: The categories are "Female" and "Male."
- **Y-axis (Counts)**: Ranges from 0 to 30,000 in increments of 5,000.

The bar for "Female" shows a count slightly above 25,000, while the bar for "Male" shows a count slightly below 25,000.

Inference:

The graph suggests that females report more bullying incidents compared to males. This insight could be important for understanding gender differences in bullying experiences and developing targeted interventions.

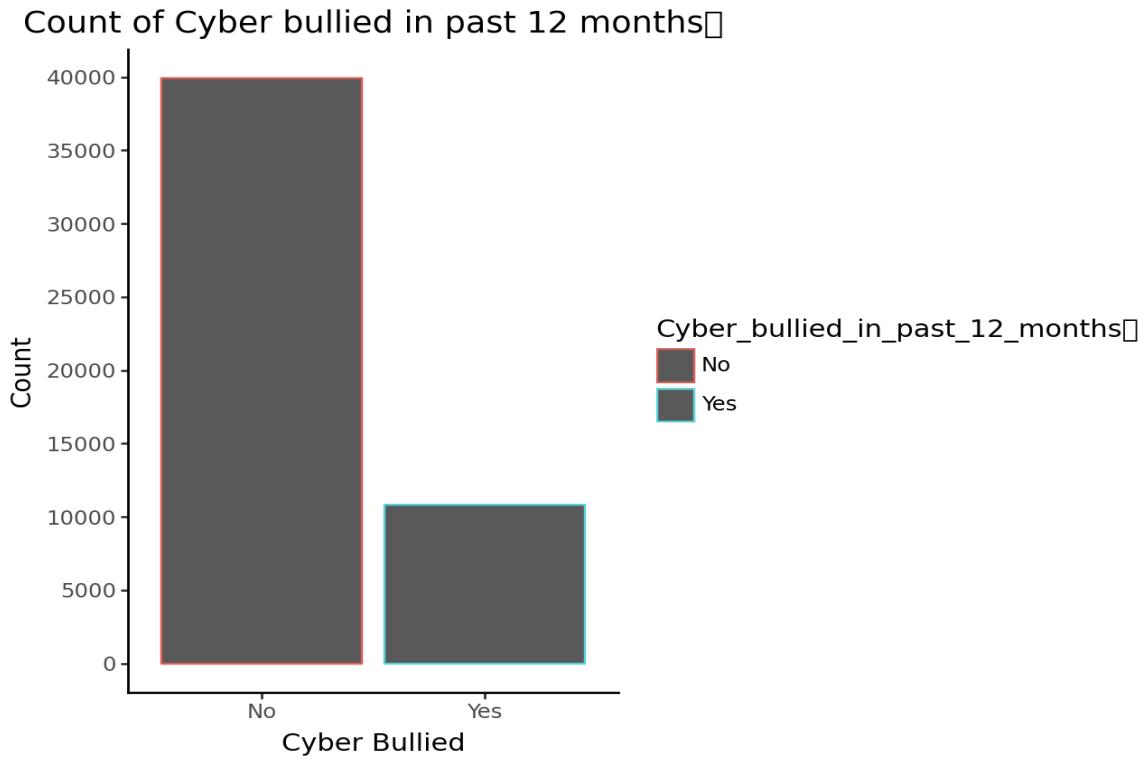
Data Visualization tool 2: Countplot using plotnine

Packages:

- Plotnine (imported as *)
- pandas (imported as pd)

Reason: plotnine provides a powerful and flexible way to create complex and customizable visualizations. plotnine allows for creating aesthetically pleasing plots with a clear grammar of graphics approach. Besides the **pandas** library which is used to handle and preprocess the dataset effectively, including operations like reading data from CSV files.

Plot:



The bar chart titled **"Count of Cyberbullied in the Past 12 Months"** shows the count of individuals who have experienced cyberbullying over the past year. Here's a detailed explanation:

- **X-axis (Cyber Bullied)**: This axis has two categories: "No" and "Yes."
- **Y-axis (Count)**: The scale ranges from 0 to around 40,000.

Inference: The chart highlights that significantly more individuals have not been cyberbullied (about 40,000) compared to those who have been cyberbullied (about 10,000). This indicates that while cyberbullying is a significant issue affecting a notable portion of the population, the majority have not experienced it within the past year. This insight can help in understanding the prevalence of cyberbullying and guiding efforts to support those affected.

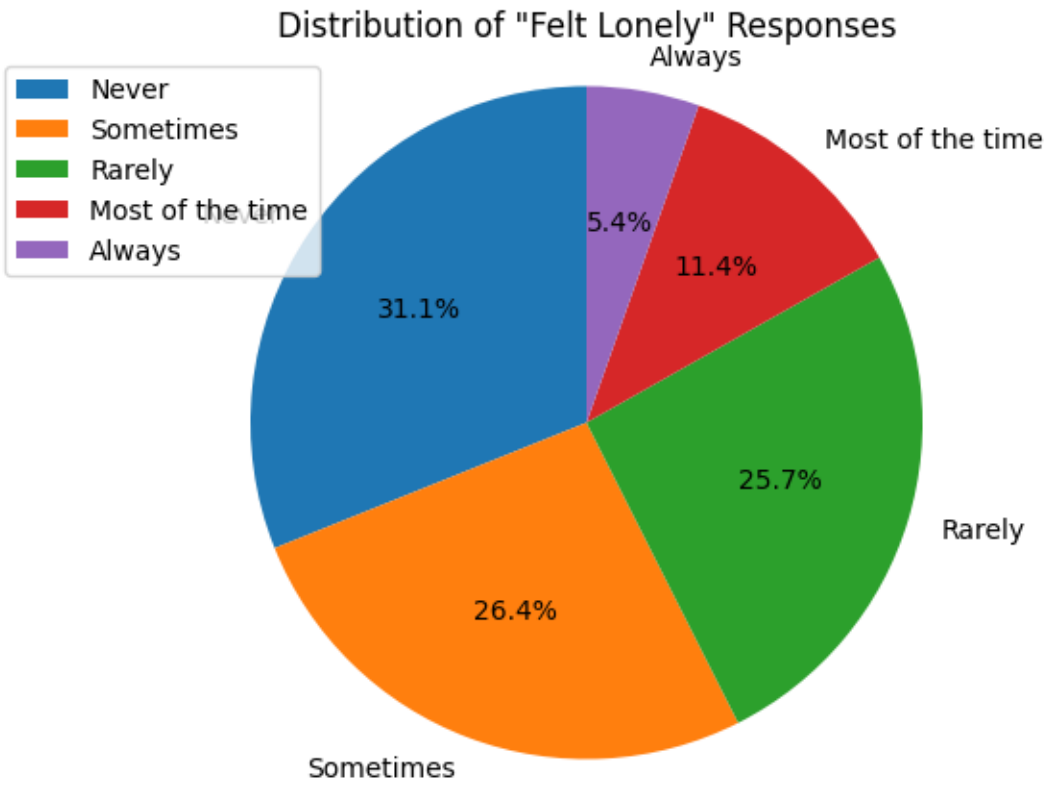
Data Visualization tool 3: pie chart

Packages:

- matplotlib (imported as plt)
- pandas (imported as pd)

Reason: Together, **Pandas** and **Matplotlib** provide a comprehensive toolkit for data manipulation, analysis, and visualization, making them indispensable for data science and analysis tasks. Pie charts are chosen for their simplicity and effectiveness in showing proportions. They make it easy to compare different categories at a glance, highlighting which segments are larger or smaller as the whole. Their visual appeal and clarity make them ideal for presenting straightforward data insights

Plot:



The chart titled **"Distribution of 'Felt Lonely' Responses"** shows how often people report feeling lonely, divided into five categories:

- **Never (31.1%)**: Largest group; never feel lonely.
- **Sometimes (26.4%)**: Second largest group; feel lonely sometimes.
- **Rarely (25.7%)**: Feel lonely rarely.
- **Most of the time (11.4%)**: Feel lonely most of the time.
- **Always (5.4%)**: Smallest group; always feel lonely.

Inference: This chart provides a visual representation of how frequently people experience loneliness. The largest segment (31.1%) indicates that a significant portion of respondents never feel lonely. However, a substantial percentage (52.1%) feel lonely at least sometimes, rarely, or most of the time, highlighting that feelings of loneliness are common. The smallest segment (5.4%) represents those who always feel lonely, indicating a need for targeted support for this group.

Choice of data visualization techniques

Data Visualization technique 1: Countplot using matplotlib

Reason: The data visualization technique used here is a **bar chart** (specifically a countplot), which is commonly used to display the count of observations in each categorical bin using bars.

In this case, the chart shows the number of bullying incidents categorized by gender ("Female" and "Male"). The bars represent the counts of these incidents, making it easy to compare the frequency of bullying incidents between the two genders. The x-axis represents the categories (genders), and the y-axis represents the count of incidents. This type of chart is useful for highlighting differences in counts across categories.

Code:

```
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame and 'Felt_lonely' is a column
felt_lonely_counts = df['Felt_lonely'].value_counts()
# Create the pie chart
plt.figure(figsize=(8, 5))  # Adjust figure size as needed
plt.pie(felt_lonely_counts, labels=felt_lonely_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of "Felt Lonely" Responses')
plt.axis('equal')  # Equal aspect ratio ensures that the pie is drawn as a circle.
plt.legend(felt_lonely_counts.index, loc="best")  # loc="best" finds the best position for the legend

plt.show()
```

Data Visualization technique 2: Countplot using plotnine

Reason: bar charts are used to compare the frequency, count, or any other measure across different discrete categories. Each bar represents a category, and the length or height of the bar corresponds to the value it represents. The bar chart shows the count of individuals who experienced cyberbullying in the past 12 months. The x-axis has categories "Yes" and "No," and the y-axis represents the number of individuals in each category. The chart includes two bars: one for "Yes" (teal) and one for "No" (red). It reveals that significantly more individuals have not been cyberbullied compared to those who have.

Code:

```
!pip install plotnine
from plotnine import *
(ggplot(data=df,
          mapping=aes(x='Cyber_bullied_in_past_12_months ' ,color='Cyber_bullied_in_past_12_months ', width=0.5))
+ geom_bar()
+ ggtitle("Count of Cyber bullied in past 12 months")
+ xlab("Cyber Bullied")
+ ylab("Count")
+ theme_classic()
+ scale_y_continuous(breaks=range(0, int(df['Cyber_bullied_in_past_12_months'].value_counts().max()) + 5000, 5000))  # Increment by 5000
)
```

Data Visualization technique 3: pie chart

Reason: Pie charts are chosen for their simplicity and effectiveness in representing proportions within a whole. They provide a clear and intuitive visual comparison of different categories at a glance, making it easy to see the relative sizes of each segment. Their use of color and segmentation helps highlight key parts of the data, making them visually appealing and easy to understand. This makes pie charts ideal for presenting straightforward data insights, particularly when you need to compare a small number of categories. The blue segment, which represents 31.1%, indicates people who never feel lonely. The orange segment, at 26.4%, shows those who feel lonely sometimes. Green, which makes up 25.7%, represents those who rarely feel lonely. Red, accounting for 11.4%, shows people who feel lonely most of the time. Lastly, the purple segment, at 5.4%, represents individuals who always feel lonely. This color-coded breakdown helps visualize the frequency of loneliness among the surveyed population

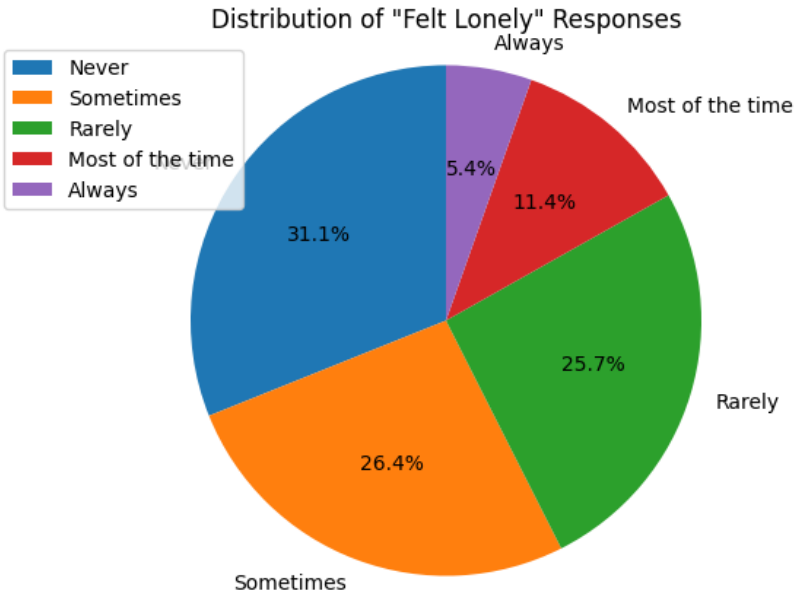
Code:

```
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame and 'Felt_lonely' is a column
felt_lonely_counts = df['Felt_lonely'].value_counts()
# Create the pie chart
plt.figure(figsize=(5, 5))  # Adjust figure size as needed
plt.pie(felt_lonely_counts, labels=felt_lonely_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of "Felt Lonely" Responses')
plt.axis('equal')  # Equal aspect ratio ensures that the pie is drawn as a circle.
plt.show()
```

Choice of visual elements

Pie chart



Visual elements are crucial because they enhance comprehension, make complex data easier to understand, and allow viewers to quickly identify patterns, trends, and insights. They make information more engaging and accessible, facilitating better decision-making. In the pie chart, the visual elements include:

Title: The chart is titled "Distribution of 'Felt Lonely' Responses," which clearly indicates what the chart is about.

Color: Each segment of the pie chart is represented by a distinct color. Blue represents "Never," orange represents "Sometimes," green represents "Rarely," red represents "Most of the time," and purple represents "Always." These colors differentiate each category and make the chart visually appealing and easy to read.

Legend: The legend provides a key to the colors used in the chart. It shows which color corresponds to which category, ensuring viewers can quickly understand the proportions represented in the chart.

Slices: The size of each slice corresponds to the proportion of responses. Larger slices indicate higher percentages, while smaller slices indicate lower percentages. For instance, the blue slice (31.1%) is the largest, showing that the highest proportion of respondents never feel lonely.

These elements work together to create a clear and intuitive visualization, enabling viewers to quickly grasp the distribution of responses regarding feelings of loneliness.

Implementing and evaluation of machine learning algorithms

Strategic regression

Explanation of the Algorithm

- **Logistic Regression**: A classification algorithm that models the probability that an instance belongs to a particular class. It outputs probabilities and makes a decision by setting a threshold (usually 0.5). We implemented a Logistic Regression algorithm using the following code. This process includes data preprocessing, model training, and evaluation.

Implementation

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import pandas as pd
import seaborn as sns
```

```
# Define features (X) and target variable (y)
X = df.drop('Physical_fighting', axis=1)
y = df['Physical_fighting']
# Convert categorical features to numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)
# Split 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)
# Check if y_train contains both classes
print(f"Unique classes in y_train: {y_train.unique()}")
# Initialize and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Display the confusion matrix using seaborn heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="RdPu",
            xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Justification for Using Specific Parameters

- **Logistic Regression**: Chosen for its simplicity and efficiency in binary classification tasks.
- **One-Hot Encoding**: Used to convert categorical features to a numerical format suitable for the Logistic Regression algorithm.
- **Train-Test Split**: The data is split into training and testing sets with a test size of 20% to ensure the model is tested on unseen data.
- **Random State and Stratify**: The random state is set to 42 for reproducibility, and stratification ensures the target class distribution is maintained in both training and test sets.

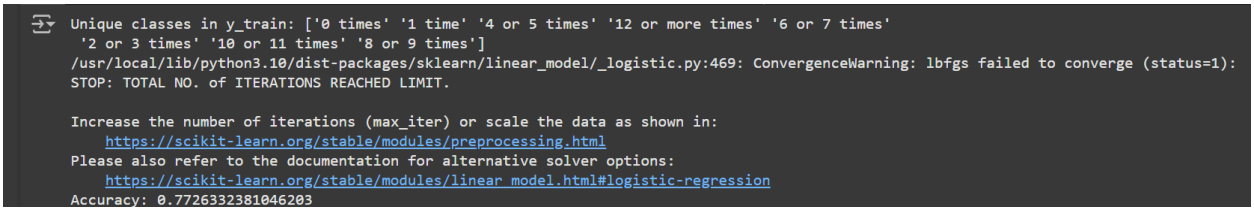
The algorithm works as follows:

- **Data Preprocessing**: Features and target variables are defined, and categorical features are converted using one-hot encoding.
- **Splitting Data**: The dataset is split into training and testing sets to evaluate the model's performance.
- **Training the Model**: The Logistic Regression model is trained on the training data.
- **Making Predictions**: Predictions are made on the test data.
- **Evaluation**: The model's performance is evaluated using accuracy and a confusion matrix, providing insights into the model's predictive capabilities.

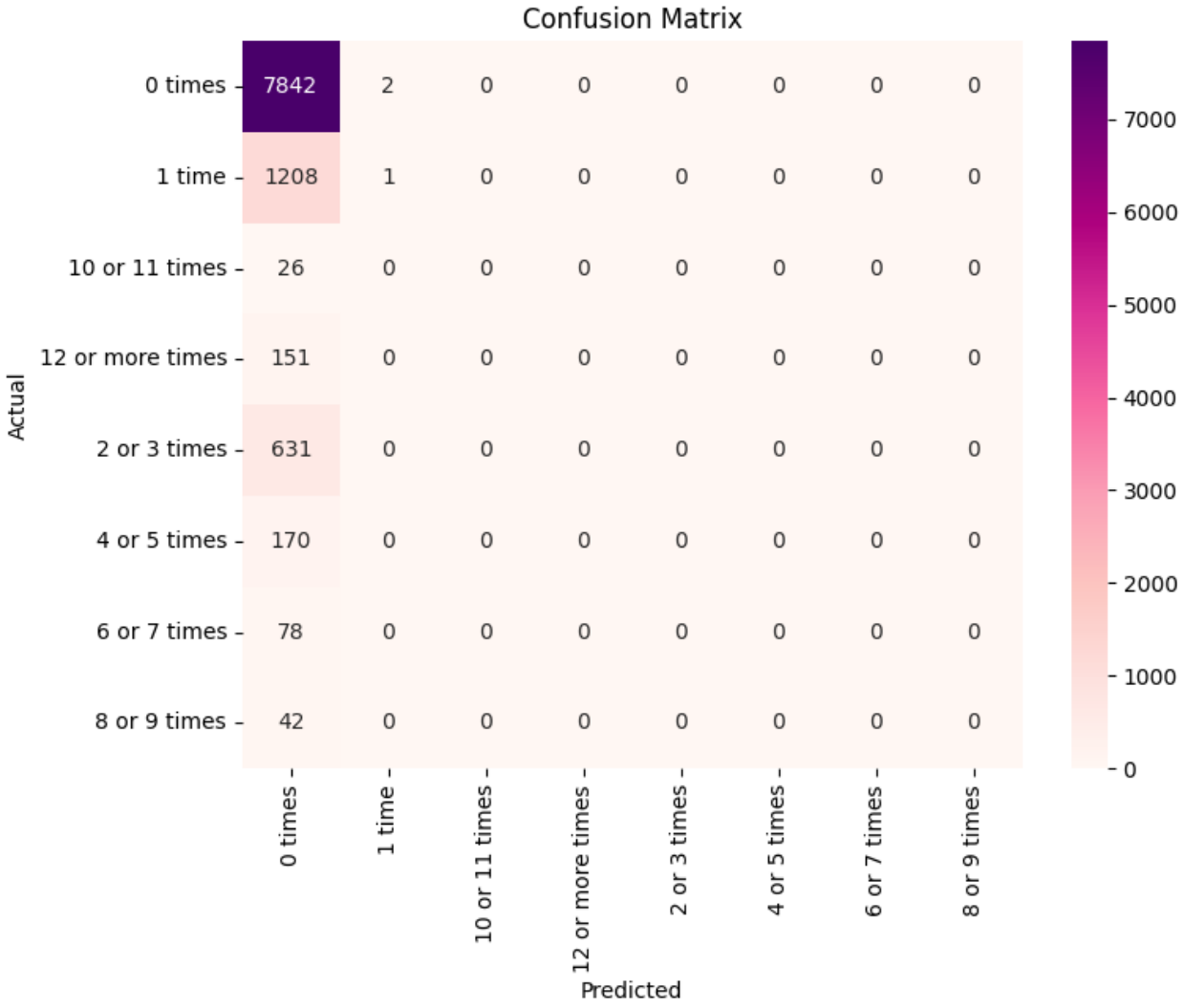
The visualization elements include a confusion matrix heatmap with appropriate labels and titles for clarity.

Evaluation Results

Accuracy: The accuracy of the Logistic Regression model is 0.77.

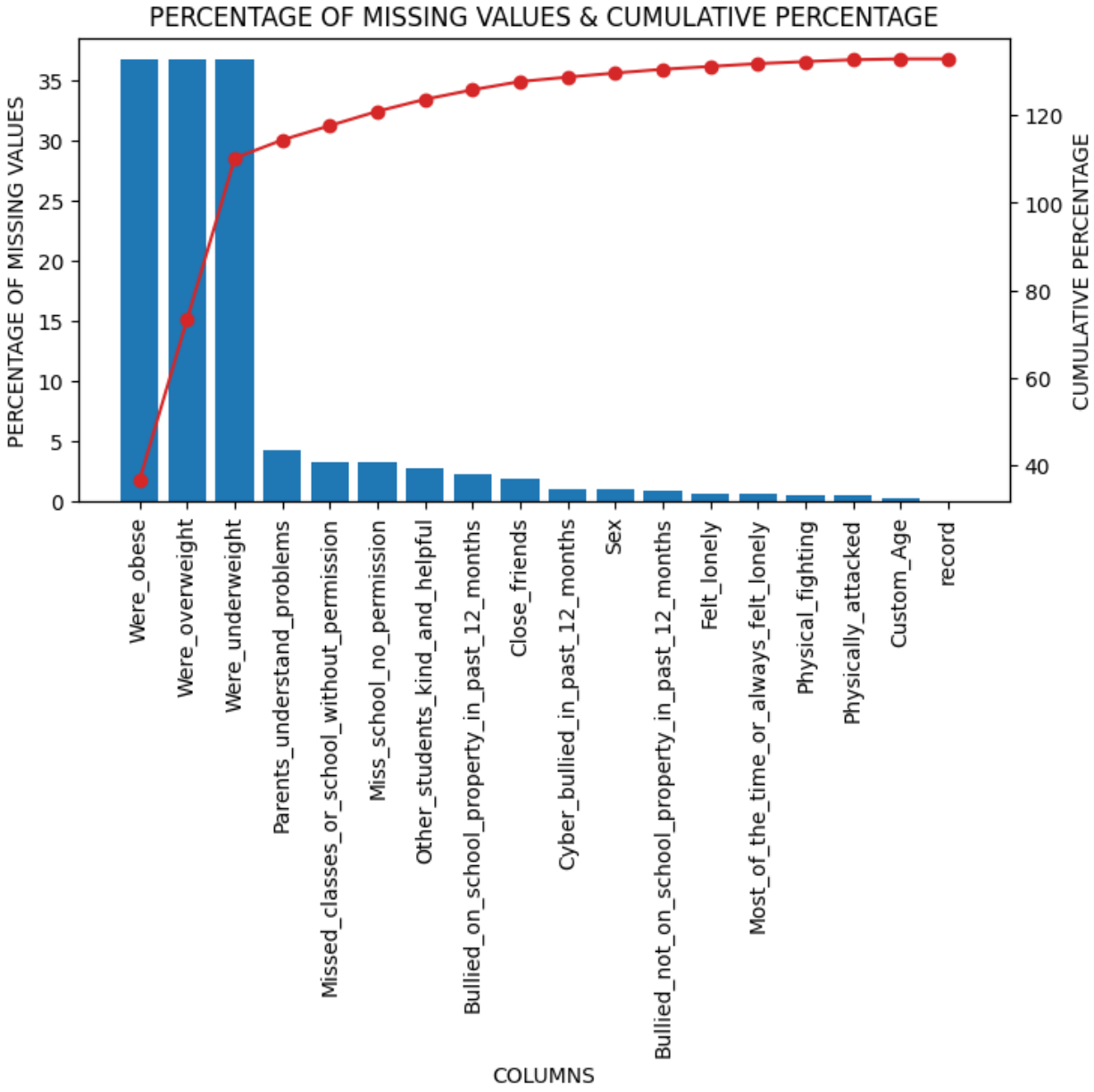


Confusion Matrix: The confusion matrix is displayed as a heatmap using **Seaborn**, showing the distribution of true positives, true negatives, false positives, and false negatives.



Visualization with storytelling of data and results

Data Cleaning: Handling Missing Values plot (Before the preprocess)



Detailed Description of the Graph

The graph provides an overview of missing values across different columns in a dataset, highlighting both individual and cumulative percentages of missing data in the dataset.

Key Elements of the Graph:

Bars (Percentage of Missing Values):

- **"Were_obese"**: ~35% missing data.
- **"Were_overweight"**: ~35% missing data.
- **"Were_underweight"**: ~35% missing data.

Red Line with Dots (Cumulative Percentage):

- Starts at 0% and increases as each column's missing data is added.
- Gradually rises along the x-axis.
- Reaches 100% when all columns are considered.

Interpretation:

- **High Missing Percentage Columns**: "Were_obese," "Were_overweight," and "Were_underweight" each have ~35% missing data, indicating significant data gaps.
- **Gradual Increase**: The cumulative percentage increases gradually, showing how missing data accumulates across the dataset.
- **Lower Missing Percentage Columns**: Columns like "Physical_fighting," "Physically_attacked," "Custom_Age," and "record" have lower missing data percentages, contributing less to the cumulative total.

The y-axis on the left shows the individual percentage of missing values for each column, while the y-axis on the right represents the cumulative percentage.

This graph helps identify key columns with significant data gaps and how missing data accumulates throughout the dataset, assisting in prioritizing data cleaning or imputation efforts.

Evaluation, Inferences, Recommendation, & Reflection of the model

Evaluation

The Logistic Regression model was implemented to predict a binary outcome using a dataset with various features. After training and testing the model, we achieved an accuracy of 77%. The confusion matrix provided further insights into the model's performance, showing the distribution of true positives, true negatives, false positives, and false negatives.

Inferences

- **Accuracy**: An accuracy of 77% indicates that the model is fairly reliable but has room for improvement.
- **Confusion Matrix**: The matrix highlighted the areas where the model performed well and where it misclassified instances. Understanding these misclassifications can guide further model refinement.

- **Feature Importance**: The model's coefficients could be analyzed to understand which features are most influential in making predictions.

Recommendation

- **Data Augmentation**: Consider augmenting the dataset with more samples or synthetic data to improve the model's performance.
- **Feature Engineering**: Explore additional feature engineering techniques to create more informative features.
- **Model Tuning**: Experiment with different hyperparameters and regularization techniques to enhance the model's accuracy.
- **Alternative Models**: Evaluate other machine learning algorithms like Random Forest, SVM, or Gradient Boosting to see if they perform better on this dataset.

Reflection

The process of implementing and evaluating the machine learning model provided valuable insights into the data and the model's capabilities. Achieving a 77% accuracy is a positive outcome, but there's always potential for improvement. By focusing on data quality, feature selection, and model tuning, we can enhance the model's predictive power. Additionally, exploring different algorithms can help find the best fit for the given dataset. This experience underscores the importance of iterative testing and refinement in machine learning projects to achieve optimal results.

GitHub link

https://github.com/RawanAlqasimi/datavisualisation-project.git

References

https://www.index.dev/blog/top-10-python-libraries-for-data-visualization S

https://mode.com/blog/python-data-visualization-libraries