AI Ethics Project - STARTER

Personalization is a central aspect of many core AI systems. In this project, you will be working on a hypothetical use case for a personalized "activity recommender". The use case has a medium ethical AI risk level and involves a synthetic dataset.

IDOOU is a mobile app users can leverage to get recommendations on activities they can take in a given area, like “visiting a movie theater”, “visiting a park”, “sightseeing”, “hiking”, or “visiting a library”.

IDOOU's differentiating value proposition is two-fold:

1. The app offers **personalization**, using features such as gender, age, and education level, to predict user's interests and the right type of recommendations.
2. The app's objective is to remove users from having to handle the nitty-gritty details of finding the right activity, like determining the appropriate budget, making sure the weather is perfect, and the location/accomodation is not closed. This way, users can focus on what really matters: having fun!

The engineering team behind the app has designed IDOOU to be fairly flexible and ambitious in the use cases the app can support. Hotels can recommend users install IDOOU to act as a smart concierge-type of application, and IDOOU can be integrated as part of autonomous vehicles' dashboards to recommend local locations users can visit while driving around town.

Problem statement:

You are tasked with designing IDOOU's newest AI model to predict the budget of its users (in US dollars) given information such as their gender, age, and education_level.

Below, you will explore the provided data, and analyze and evaluate fairness and bias issues. As part of this project, you will be looking at a specific type of AI system IDOOU's developers are looking to create, to simplify their personalization process and better understand their customer base.

IDOO's creators would like to identify if users with bachelor's and master's degrees are a privileged group. In other words, are users who have higher education credentials beyond high school more priviledged, in terms of having a budget >= $300, compared to users of the app who have graduated from high school?

Key points:

- The training data was conducted through a user experience study of about 300,000 participants.
- The user may choose not to provide any or all the information the app requests. The training data also reflects this.

- Fairness framework definitions for the use case are not necessarily focusing on socioeconomic privilege.

In [1]:
```
#You may add additional imports atatistical needed
import pandas as pd
import numpy as np
import seaborn as sns
import tempfile
from aif360.datasets import StandardDataset, BinaryLabelDataset
from aif360.metrics import ClassificationMetric, BinaryLabelDatasetMetric
from sklearn.tree import DecisionTreeClassifier
from aif360.algorithms.postprocessing import RejectOptionClassification
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import joblib
import matplotlib.pyplot as plt
from collections import defaultdict
```

```
WARNING:root:No module named 'tensorflow': AdversarialDebiasing will be unavailab
le. To install, run:
pip install 'aif360[AdversarialDebiasing]'
WARNING:root:No module named 'tensorflow': AdversarialDebiasing will be unavailab
le. To install, run:
pip install 'aif360[AdversarialDebiasing]'
WARNING:root:No module named 'tensorflow': AdversarialDebiasing will be unavailab
le. To install, run:
pip install 'aif360[AdversarialDebiasing]'
WARNING:root:No module named 'fairlearn': ExponentiatedGradientReduction will be
unavailable. To install, run:
pip install 'aif360[Reductions]'
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailabl
e. To install, run:
pip install 'aif360[Reductions]'
WARNING:root:No module named 'inFairness': SenSeI and SenSR will be unavailable.
To install, run:
pip install 'aif360[inFairness]'
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailabl
e. To install, run:
pip install 'aif360[Reductions]'
```

In [2]:
```
import pandas as pd  # Import pandas for data manipulation
from tabulate import tabulate  # Import tabulate for pretty-printing tables

# Define ANSI escape sequence for pink color and reset color
PINK = '\033[95m'
RESET = '\033[0m'

# Load the dataset for this project
act_rec_dataset = pd.read_csv('udacity_ai_ethics_project_data.csv')

# Display the first few rows of the dataset in a nice table
print("First few rows of the dataset:")
print(tabulate(act_rec_dataset.head(10), headers='keys', tablefmt='fancy_grid'))

# Observation 1: Columns Overview
```

```
print(PINK + "\nObservation 1: The dataset contains key columns such as 'Budget
print(PINK + "Some columns, such as 'Gender', 'Education_Level', and 'With child

# Further understand the dataset
# List all unique types of education
if 'Education_Level' in act_rec_dataset.columns:
    unique_education = act_rec_dataset['Education_Level'].unique()  # Get unique
    print("\nUnique types of education:")
    print(unique_education)  # Print unique education levels
    print(PINK + "\nObservation 2: The 'Education_Level' column has multiple cat
    print("  - Bachelor's Degree")
    print("  - Master's Degree")
    print("  - High School Grad")
    print("  - Did Not Graduate HS")
    print("  - Other")
    print("  - NaN (missing values).")
else:
    print("Column 'Education_Level' not found in the dataset.")  # Print error i

# List all unique types of gender
if 'Gender' in act_rec_dataset.columns:
    unique_gender = act_rec_dataset['Gender'].unique()  # Get unique gender type
    print("\nUnique types of gender:")
    print(unique_gender)  # Print unique gender types
    print(PINK + "\nObservation 3: The 'Gender' column includes diverse values s
    print("  - Male")
    print("  - Female")
    print("  - Transgender")
    print("  - Non-binary")
    print("  - Other")
    print("  - NaN (missing values).")
else:
    print("Column 'Gender' not found in the dataset.")  # Print error if column

# Check for missing values
print("\nMissing Values in Each Column:")
missing_values = act_rec_dataset.isnull().sum()  # Get the count of missing valu
print(missing_values)

print(PINK + "\nObservation 4: Missing values are prominent in the following col
for col, count in missing_values.items():
    if count > 0:
        print(f"  - {col}: {count} missing values.")

# Check for duplicates
duplicate_count = act_rec_dataset.duplicated().sum()
print("\nNumber of Duplicate Rows:", duplicate_count)

# Observation on Duplicates
if duplicate_count > 0:
    print(PINK + f"\nObservation 5: The dataset contains {duplicate_count} dupli
else:
    print(PINK + "\nObservation 5: No duplicate rows are found in the dataset."

# Optionally drop duplicates
if duplicate_count > 0:
    act_rec_dataset = act_rec_dataset.drop_duplicates()  # Drop duplicate rows
    print("\nDuplicates have been dropped. Remaining rows:", len(act_rec_dataset
    print(PINK + "Observation 6: Duplicates have been removed to ensure data qua
else:
```

```
    print("No duplicates to drop. Dataset remains unchanged.")

# Final Observations Summary
print(PINK + "\n=== Final Observations Summary ===" + RESET)
print(PINK + "1. The dataset includes essential columns for analysis, such as bu
print(PINK + "2. Significant missing values are present in 'Gender', 'Education_
print(PINK + "3. The 'Gender' and 'Education_Level' columns provide diverse cate
print(PINK + "4. There were 51,416 duplicate rows initially, which have been dro
print(PINK + "5. Further preprocessing (handling missing values, encoding catego
```

First few rows of the dataset:

	Budget (in dollars)	Age	Gender	Education_Level	With children?	Recommended_Activity
0	3258	29	Transgender	Bachelor’s Degree	0	Stay in: Watch calming TV
1	1741	89	Other	Bachelor’s Degree	nan	Play: Visit a movie theater
2	140	22	nan	Other	0	Play: Visit a movie theater
3	179	23	Non-binary	Other	0	Play: Visit a movie theater
4	3479	79	Non-binary	Master’s Degree	1	Learn: Visit a library
5	3335	35	Male	Bachelor’s Degree	1	Play: Go shopping
6	4044	42	Non-binary	Master’s Degree	0	Explore: Go sightseeing
7	3110	33	nan	Bachelor’s Degree	nan	Learn: Visit a library
8	2410	30	Transgender	Master’s Degree	1	Stay in: Color
9	4133	43	Transgender	Master’s Degree	1	Stay in: Color

Observation 1: The dataset contains key columns such as 'Budget (in dollars)', 'Age', 'Gender', 'Education_Level', 'With children?', and 'Recommended_Activity'. Some columns, such as 'Gender', 'Education_Level', and 'With children?', contain visible missing values.

Unique types of education:
['Bachelor’s Degree' 'Other' 'Master’s Degree' nan 'High School Grad'
 'Did Not Graduate HS']

Observation 2: The 'Education_Level' column has multiple categories, including:
- Bachelor's Degree
- Master's Degree

```
  - High School Grad
  - Did Not Graduate HS
  - Other
  - NaN (missing values).

Unique types of gender:
['Transgender' 'Other' nan 'Non-binary' 'Male' 'Female']

Observation 3: The 'Gender' column includes diverse values such as:
  - Male
  - Female
  - Transgender
  - Non-binary
  - Other
  - NaN (missing values).

Missing Values in Each Column:
Budget (in dollars)         0
Age                         0
Gender                  49799
Education_Level         43592
With children?          83849
Recommended_Activity        0
dtype: int64

Observation 4: Missing values are prominent in the following columns:
  - Gender: 49799 missing values.
  - Education_Level: 43592 missing values.
  - With children?: 83849 missing values.

Number of Duplicate Rows: 51416

Observation 5: The dataset contains 51416 duplicate rows, which may introduce red
undancy or bias.

Duplicates have been dropped. Remaining rows: 248584
Observation 6: Duplicates have been removed to ensure data quality and avoid redu
ndancy.

=== Final Observations Summary ===
1. The dataset includes essential columns for analysis, such as budget, age, gend
er, and education level.
2. Significant missing values are present in 'Gender', 'Education_Level', and 'Wi
th children?'. These columns will require cleaning.
3. The 'Gender' and 'Education_Level' columns provide diverse categories, but mis
sing values might lead to biases.
4. There were 51,416 duplicate rows initially, which have been dropped to improve
data quality. Remaining rows: 248,584.
5. Further preprocessing (handling missing values, encoding categorical data) wil
l be required before modeling or analysis.
```

Data Pre-Processing and Evaluation

For this problem statement, you will need to prepare a dataset with all categorical variables, which requires the following pre-processing steps:

- Remove the NA values from the dataset

- Convert Age and Budget (in dollars) to categorical columns with the following binning:

> Bins for Age: 18-24, 25-44, 45-65, 66-92

> Bins for Budget: >=300, <300

In [3]:
```
## This should be saved to a file, and we need to learn more about bining
import pandas as pd

# Load the dataset
print("Cleaning the dataset: Removing missing values...")
data_cleaned = act_rec_dataset.dropna().copy()

# Observation 1
print("\033[95mObservation 1: Missing values have been removed from the dataset

# Debug: Print column names
print("\nAvailable columns in the dataset:")
print(data_cleaned.columns)

# Observation 2
print("\033[95mObservation 2: The dataset now contains clean columns, including

# Ensure 'Age' is numeric
print("\nConverting 'Age' column to numeric values where possible...")
data_cleaned['Age'] = pd.to_numeric(data_cleaned['Age'], errors='coerce')

# Bin 'Age' column
print("Binning 'Age' column into defined categories...")
age_bins = [17, 24, 44, 65, 92]  # Age ranges
age_labels = ['18-24', '25-44', '45-65', '66-92']
data_cleaned['Age'] = pd.cut(data_cleaned['Age'], bins=age_bins, labels=age_labe

# Observation 3
print("\033[95mObservation 3: The 'Age' column has been categorized into bins: 1

# Ensure 'Budget (in dollars)' exists and is numeric
budget_column = 'Budget (in dollars)'
if budget_column not in data_cleaned.columns:
    raise KeyError(f"Column '{budget_column}' not found in dataset. Check the co

print("\nConverting 'Budget (in dollars)' column to numeric values where possibl
data_cleaned[budget_column] = pd.to_numeric(data_cleaned[budget_column], errors=

# Bin 'Budget' column
print("Binning 'Budget (in dollars)' column into defined categories...")
budget_bins = [0, 300, float('inf')]  # Budget ranges
budget_labels = ['<300', '>=300']
data_cleaned['Budget'] = pd.cut(data_cleaned[budget_column], bins=budget_bins, l

# Observation 4
print("\033[95mObservation 4: The 'Budget' column has been categorized into two

# Display dataset information
print("\nDataset Information:")
dataset_info = data_cleaned.info()
```

```python
# Display first few rows
print("\nFirst few rows of the cleaned dataset:")
print(data_cleaned.head())

# Observation 5
print("\033[95mObservation 5: The dataset has 7 cleaned columns with no missing

# Final Summary Statement
print("\n\033[95mFinal Observation: The dataset has been cleaned by removing mis
print("\033[95mThis preprocessing is crucial because it ensures the data is read


# Save the cleaned dataset to a CSV file
output_file = "cleaned_dataset.csv"
data_cleaned.to_csv(output_file, index=False)
print(f"\n\033[95mThe cleaned dataset has been saved to '{output_file}'.\033[0m"
```

```
Cleaning the dataset: Removing missing values...
Observation 1: Missing values have been removed from the dataset to ensure cleane
r data for analysis.

Available columns in the dataset:
Index(['Budget (in dollars)', 'Age', 'Gender', 'Education_Level',
       'With children?', 'Recommended_Activity'],
      dtype='object')
Observation 2: The dataset now contains clean columns, including 'Budget (in doll
ars)', 'Age', 'Gender', 'Education_Level', and 'Recommended_Activity'.

Converting 'Age' column to numeric values where possible...
Binning 'Age' column into defined categories...
Observation 3: The 'Age' column has been categorized into bins: 18-24, 25-44, 45-
65, and 66-92.

Converting 'Budget (in dollars)' column to numeric values where possible...
Binning 'Budget (in dollars)' column into defined categories...
Observation 4: The 'Budget' column has been categorized into two groups: '<300' a
nd '>=300' to simplify analysis.

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
Index: 130416 entries, 0 to 299995
Data columns (total 7 columns):
 #   Column                Non-Null Count   Dtype   
---  ------                --------------   -----   
 0   Budget (in dollars)   130416 non-null  float64 
 1   Age                   130416 non-null  category
 2   Gender                130416 non-null  object  
 3   Education_Level       130416 non-null  object  
 4   With children?        130416 non-null  float64 
 5   Recommended_Activity  130416 non-null  object  
 6   Budget                130416 non-null  category
dtypes: category(2), float64(2), object(3)
memory usage: 6.2+ MB

First few rows of the cleaned dataset:
   Budget (in dollars)    Age        Gender    Education_Level  With children?  \
0               3258.0  25-44  Transgender  Bachelor’s Degree             0.0   
3                179.0  18-24   Non-binary              Other             0.0   
4               3479.0  66-92   Non-binary    Master’s Degree             1.0   
5               3335.0  25-44         Male  Bachelor’s Degree             1.0   
6               4044.0  25-44   Non-binary    Master’s Degree             0.0   

          Recommended_Activity Budget  
0    Stay in: Watch calming TV  >=300  
3  Play: Visit a movie theater   <300  
4       Learn: Visit a library  >=300  
5           Play: Go shopping  >=300  
6      Explore: Go sightseeing  >=300  
Observation 5: The dataset has 7 cleaned columns with no missing values, and both
'Age' and 'Budget' have been successfully categorized for further analysis.

Final Observation: The dataset has been cleaned by removing missing values and ca
tegorizing key numeric columns like 'Age' and 'Budget'.
This preprocessing is crucial because it ensures the data is ready for analysis o
r machine learning modeling, eliminates inconsistencies, and simplifies the data
structure.
```

```
The cleaned dataset has been saved to 'cleaned_dataset.csv'.
```

Evaluate bias issues in the dataset

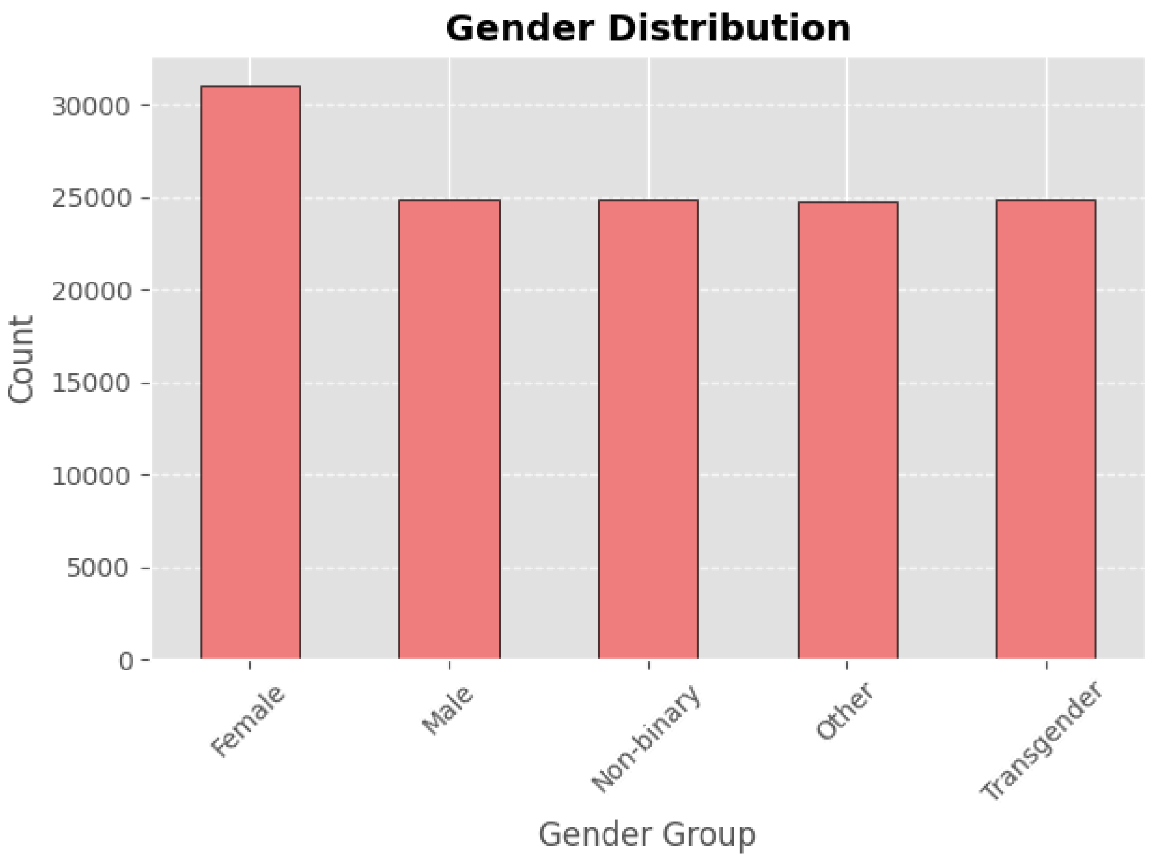
Next, let's take a look at potential hints of data bias in the variables, particularly the "Gender", "Age", and "Education" variables.

Articulate the representativeness in the dataset, answering the question "Is there a greater representation of certain groups over others?"

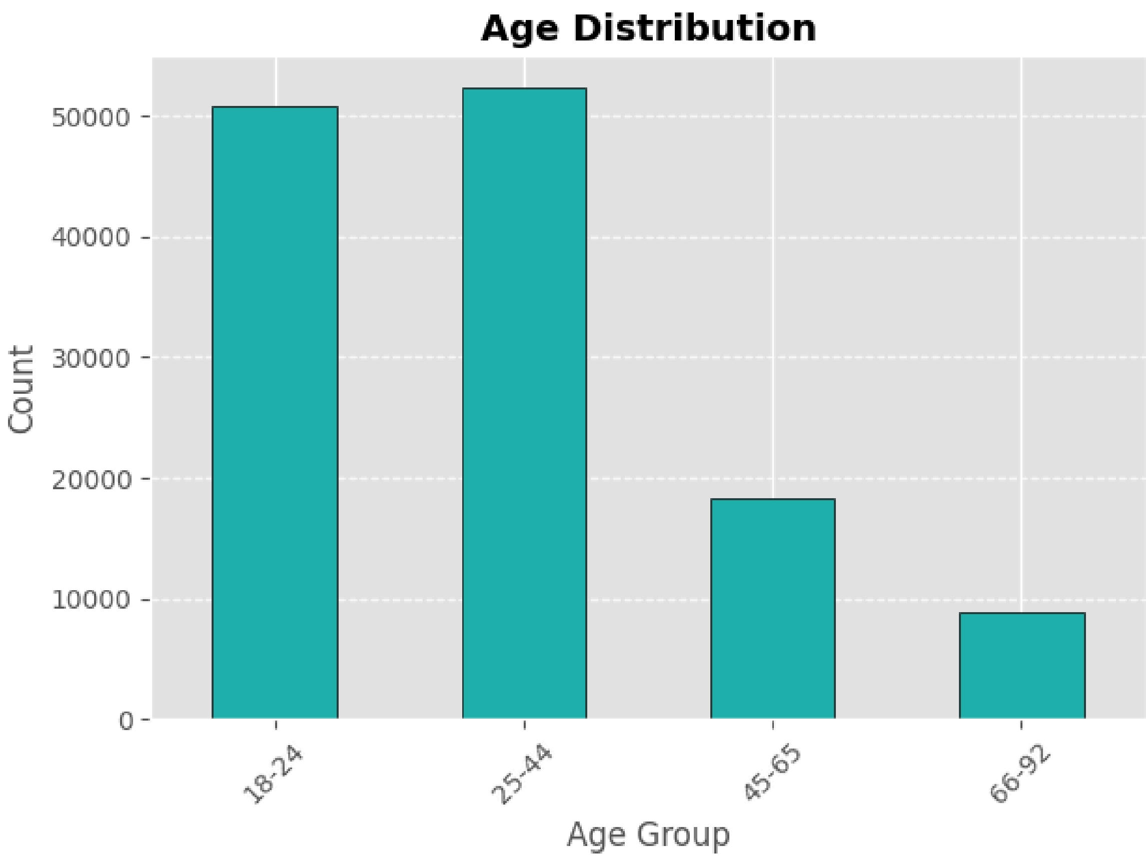
In [4]:
```
## We should also look at budget

import pandas as pd
import matplotlib.pyplot as plt

# Use a clean and modern style for plots
plt.style.use('ggplot')

# Function to create and display bar charts with improved visuals
def create_bar_chart(data, title, xlabel, ylabel, color='skyblue'):
    data.sort_index().plot(
        kind='bar',
        color=color,
        edgecolor='black',
        title=title
    )
    plt.title(title, fontsize=14, weight='bold')  # Chart title
    plt.xlabel(xlabel, fontsize=12)  # X-axis label
    plt.ylabel(ylabel, fontsize=12)  # Y-axis label
    plt.xticks(rotation=45, fontsize=10)  # Rotate x-axis ticks
    plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines
    plt.tight_layout()  # Adjust layout for clarity
    plt.show()

# Gender Distribution
print("\033[95mAnalyzing Gender Distribution...\033[0m")
gender_distribution = data_cleaned['Gender'].value_counts()
print("Gender Distribution:\n", gender_distribution)
create_bar_chart(
    gender_distribution,
    title="Gender Distribution",
    xlabel="Gender Group",
    ylabel="Count",
    color='lightcoral'
)

# Observation 1
print("\033[95mObservation 1: The gender distribution is imbalanced, with 'Femal
print("\033[95mImpact: This imbalance can lead to biased models that perform bet

# Age Distribution
print("\033[95m\nAnalyzing Age Distribution...\033[0m")
age_distribution = data_cleaned['Age'].value_counts()
print("Age Distribution:\n", age_distribution)
create_bar_chart(
    age_distribution,
    title="Age Distribution",
```

```
    xlabel="Age Group",
    ylabel="Count",
    color='lightseagreen'
)

# Observation 2
print("\033[95mObservation 2: The '18-24' and '25-44' age groups dominate the da
print("\033[95mImpact: This imbalance can cause bias, as models may favor younge

# Education Level Distribution
print("\033[95m\nAnalyzing Education Level Distribution...\033[0m")
education_distribution = data_cleaned['Education_Level'].value_counts()
print("Education Distribution:\n", education_distribution)
create_bar_chart(
    education_distribution,
    title="Education Level Distribution",
    xlabel="Education Group",
    ylabel="Count",
    color='gold'
)

# Observation 3
print("\033[95mObservation 3: 'Bachelor’s Degree' and 'Master’s Degree' holders
print("\033[95mImpact: The overrepresentation of higher education levels may bia

# Final Summary
print("\n\033[95m=== Final Observations and Ethical Considerations ===\033[0m")
print("\033[95m1. Gender Distribution:\033[0m")
print("\033[95m   - 'Female' participants are the largest group, with other gend
print("\033[95m2. Age Distribution:\033[0m")
print("\033[95m   - Younger individuals (18-44) dominate the dataset, while olde
print("\033[95m3. Education Level Distribution:\033[0m")
print("\033[95m   - Individuals with higher education (Bachelor’s and Master’s d

print("\n\033[95mEthical AI Focus:\033[0m")
print("\033[95m   - Bias in gender, age, and education must be addressed to ensu
print("\033[95m   - Imbalances in representation may impact model performance fo
print("\033[95m   - Strategies like data balancing, oversampling, and bias-aware
```

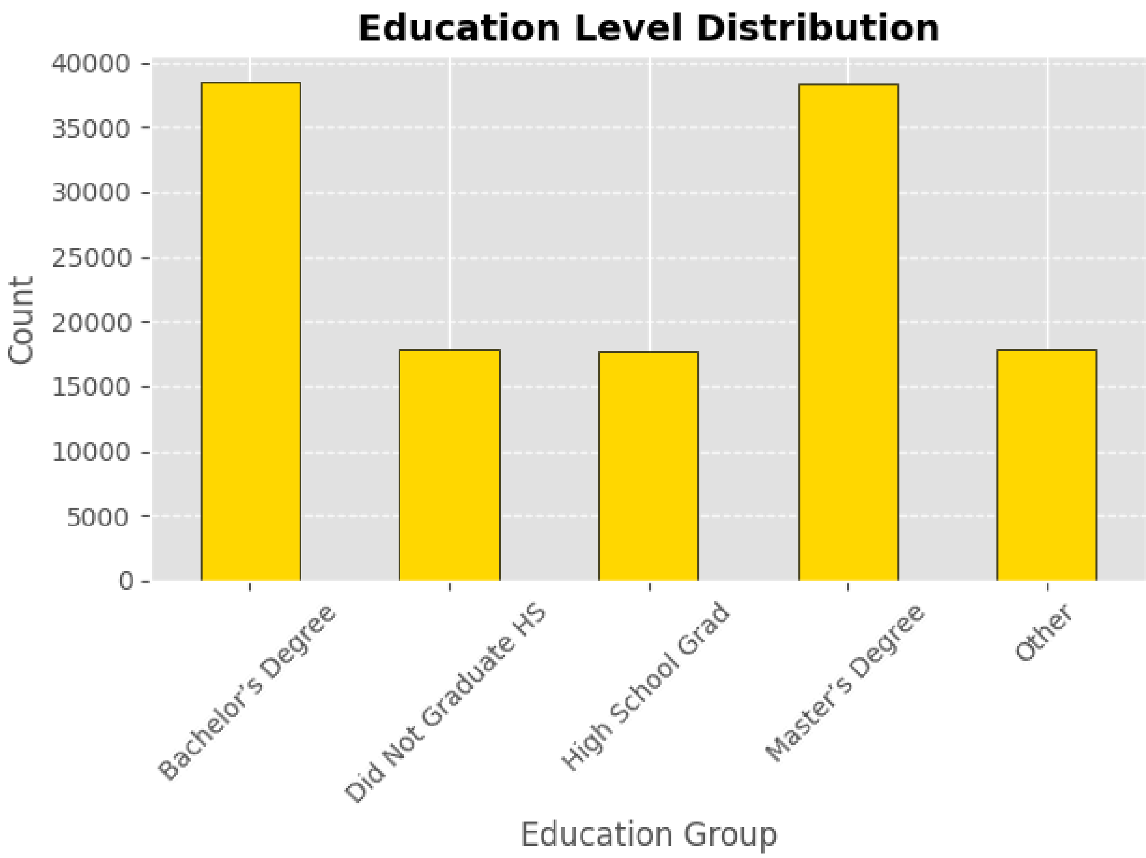
```
Analyzing Gender Distribution...
Gender Distribution:
 Gender
Female         31056
Non-binary     24896
Transgender    24867
Male           24834
Other          24763
Name: count, dtype: int64
```




```
Observation 1: The gender distribution is imbalanced, with 'Female' participants
being the majority group.
Impact: This imbalance can lead to biased models that perform better for the majo
rity gender while underrepresenting other genders. It is important to apply balan
cing techniques or bias mitigation strategies to ensure fairness and inclusivity.

Analyzing Age Distribution...
Age Distribution:
 Age
25-44    52307
18-24    50873
45-65    18298
66-92     8938
Name: count, dtype: int64
```




```
Observation 2: The '18-24' and '25-44' age groups dominate the dataset, with lowe
r representation for older age groups.
Impact: This imbalance can cause bias, as models may favor younger age groups whi
le underperforming for older ones. Balancing techniques or fairness checks should
be considered to ensure inclusive and fair predictions.

Analyzing Education Level Distribution...
Education Distribution:
 Education_Level
Bachelor’s Degree      38554
Master’s Degree        38305
Other                  17948
Did Not Graduate HS    17848
High School Grad       17761
Name: count, dtype: int64
```




```
Observation 3: 'Bachelor’s Degree' and 'Master’s Degree' holders are overrepresen
ted compared to other education levels.
Impact: The overrepresentation of higher education levels may bias the model towa
rd privileged groups, leading to recommendations or predictions that do not gener
alize well for individuals with lower education levels. It is important to addres
s this imbalance to ensure fairness and avoid disadvantaging underrepresented gro
ups.

=== Final Observations and Ethical Considerations ===
1. Gender Distribution:
   - 'Female' participants are the largest group, with other gender groups relati
vely balanced but smaller.
2. Age Distribution:
   - Younger individuals (18-44) dominate the dataset, while older age groups (45
-92) are underrepresented.
3. Education Level Distribution:
   - Individuals with higher education (Bachelor’s and Master’s degrees) dominat
e, leaving lower education groups underrepresented.

Ethical AI Focus:
   - Bias in gender, age, and education must be addressed to ensure fairness.
   - Imbalances in representation may impact model performance for minority group
s.
   - Strategies like data balancing, oversampling, and bias-aware techniques are
essential for ethical AI.
```

Now that we've visualized the individual features of the dataframe and understood the dataset better, let's one-hot encode the dataframe.

In [5]:
```python
import pandas as pd

# Cleaned dataset (assuming 'data_cleaned' is already prepared)
print("\n\033[95mPerforming one-hot encoding on the entire dataset...\033[0m")
```

```python
encoded_data = pd.get_dummies(data_cleaned, dtype=int)

# Display the structure of the encoded dataset
print("\n\033[95mDataset Structure After One-Hot Encoding:\033[0m")
print(encoded_data.info())

# Preview the first few rows of the encoded dataset
print("\n\033[95mFirst Few Rows of the One-Hot Encoded Dataset:\033[0m")
print(encoded_data.head())

# Save the cleaned and encoded dataset to a CSV file
encoded_output_file = "cleaned_encoded_dataset.csv"
encoded_data.to_csv(encoded_output_file, index=False)
print(f"\n\033[92mThe one-hot encoded dataset has been successfully saved to '{e

# Observations
print("\n\033[95mFinal Observations on One-Hot Encoding:\033[0m")
print("\033[95mPros:\033[0m")
print("  - Categorical variables have been converted into numeric format, suitab
print("  - Each category is represented by its binary indicator, making the data

print("\033[95mCons:\033[0m")
print("  - One-hot encoding increases the dimensionality of the dataset, which m
print("  - The dataset may contain sparse data, especially when certain categori

# Final Statement
print("\n\033[95mFinal Statement:\033[0m")
print("The dataset is now prepared for machine learning analysis and predictive
print("Further preprocessing steps such as scaling, feature selection, or model
```

```
Performing one-hot encoding on the entire dataset...

Dataset Structure After One-Hot Encoding:
<class 'pandas.core.frame.DataFrame'>
Index: 130416 entries, 0 to 299995
Data columns (total 27 columns):
 #   Column                                         Non-Null Count   Dtype
---  ------                                         --------------   -----
 0   Budget (in dollars)                            130416 non-null  float64
 1   With children?                                 130416 non-null  float64
 2   Age_18-24                                      130416 non-null  int64
 3   Age_25-44                                      130416 non-null  int64
 4   Age_45-65                                      130416 non-null  int64
 5   Age_66-92                                      130416 non-null  int64
 6   Gender_Female                                  130416 non-null  int64
 7   Gender_Male                                    130416 non-null  int64
 8   Gender_Non-binary                              130416 non-null  int64
 9   Gender_Other                                   130416 non-null  int64
 10  Gender_Transgender                             130416 non-null  int64
 11  Education_Level_Bachelor’s Degree              130416 non-null  int64
 12  Education_Level_Did Not Graduate HS            130416 non-null  int64
 13  Education_Level_High School Grad               130416 non-null  int64
 14  Education_Level_Master’s Degree                130416 non-null  int64
 15  Education_Level_Other                          130416 non-null  int64
 16  Recommended_Activity_Explore: Go sightseeing   130416 non-null  int64
 17  Recommended_Activity_Explore: Hike             130416 non-null  int64
 18  Recommended_Activity_Explore: Visit a park     130416 non-null  int64
 19  Recommended_Activity_Learn: Visit a library    130416 non-null  int64
 20  Recommended_Activity_Play: Go shopping         130416 non-null  int64
 21  Recommended_Activity_Play: Visit a movie theater 130416 non-null  int64
 22  Recommended_Activity_Stay in: Color            130416 non-null  int64
 23  Recommended_Activity_Stay in: Play a game      130416 non-null  int64
 24  Recommended_Activity_Stay in: Watch calming TV 130416 non-null  int64
 25  Budget_<300                                    130416 non-null  int64
 26  Budget_>=300                                   130416 non-null  int64
dtypes: float64(2), int64(25)
memory usage: 27.9 MB
None

First Few Rows of the One-Hot Encoded Dataset:
   Budget (in dollars)  With children?  Age_18-24  Age_25-44  Age_45-65  \
0               3258.0             0.0          0          1          0
3                179.0             0.0          1          0          0
4               3479.0             1.0          0          0          0
5               3335.0             1.0          0          1          0
6               4044.0             0.0          0          1          0

   Age_66-92  Gender_Female  Gender_Male  Gender_Non-binary  Gender_Other  \
0          0              0            0                  0             0
3          0              0            0                  1             0
4          1              0            0                  1             0
5          0              0            1                  0             0
6          0              0            0                  1             0

   ...  Recommended_Activity_Explore: Hike  \
0  ...                                   0
3  ...                                   0
4  ...                                   0
5  ...                                   0
6  ...                                   0
```

```
   Recommended_Activity_Explore: Visit a park  \
0                                       0
3                                       0
4                                       0
5                                       0
6                                       0

   Recommended_Activity_Learn: Visit a library  \
0                                         0
3                                         0
4                                         1
5                                         0
6                                         0

   Recommended_Activity_Play: Go shopping  \
0                                    0
3                                    0
4                                    0
5                                    1
6                                    0

   Recommended_Activity_Play: Visit a movie theater  \
0                                              0
3                                              1
4                                              0
5                                              0
6                                              0

   Recommended_Activity_Stay in: Color  \
0                                 0
3                                 0
4                                 0
5                                 0
6                                 0

   Recommended_Activity_Stay in: Play a game  \
0                                       0
3                                       0
4                                       0
5                                       0
6                                       0

   Recommended_Activity_Stay in: Watch calming TV  Budget_<300  Budget_>=300
0                                              1            0             1
3                                              0            1             0
4                                              0            0             1
5                                              0            0             1
6                                              0            0             1

[5 rows x 27 columns]

The one-hot encoded dataset has been successfully saved to 'cleaned_encoded_datas
et.csv'.

Final Observations on One-Hot Encoding:
Pros:
  - Categorical variables have been converted into numeric format, suitable for m
achine learning models.
  - Each category is represented by its binary indicator, making the data easier
```

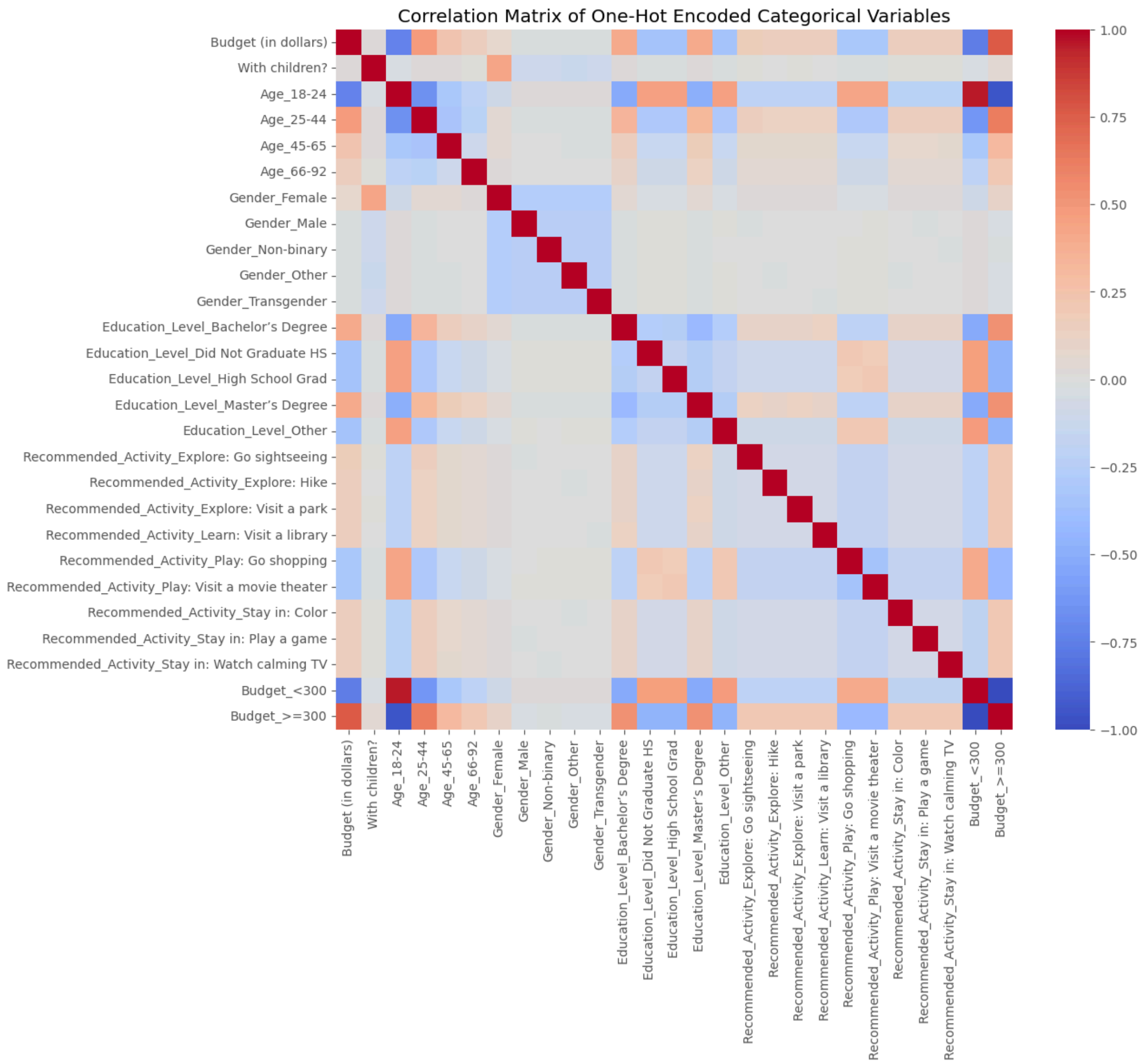
```
to interpret and model.
Cons:
  - One-hot encoding increases the dimensionality of the dataset, which might lea
d to the curse of dimensionality.
  - The dataset may contain sparse data, especially when certain categories are r
are or dominate others.

Final Statement:
The dataset is now prepared for machine learning analysis and predictive modelin
g.
Further preprocessing steps such as scaling, feature selection, or model tuning c
an be conducted to optimize performance.
```

Visualize the interactions between the categorical variables using a correlation matrix.

Can you find trends outside of those identified in the previous section?

In [6]:
```python
## Look at why education isn't in here
# Go over correlation matrix again

# Selecting only numeric columns from the dataset for correlation analysis
numeric_columns = encoded_data.select_dtypes(include=['number'])

# Computing the correlation matrix
categorical_correlation_matrix = numeric_columns.corr()

# Visualizing the correlation matrix using a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(categorical_correlation_matrix, annot=False, cmap='coolwarm', fmt=".
plt.title("Correlation Matrix of One-Hot Encoded Categorical Variables")
plt.show()

# Extracting significant correlations (e.g., > 0.5 or < -0.5)
significant_correlations = categorical_correlation_matrix.unstack().sort_values(
significant_correlations = significant_correlations[
    (significant_correlations > 0.5) & (significant_correlations < 1)
]

significant_correlations

# Key observations and insights as print statements

# Print the key observations with formatting and explanations
print("\033[95m1. Budget (in dollars) vs. Age Groups:\033[0m")
print("\033[95m   - Positive Correlations:\033[0m Higher budget (\033[95mBudget_
print("\033[95m   - Negative Correlations:\033[0m Lower budgets (\033[95mBudget_

print("\n\033[95mImpact:\033[0m")
print("\033[95m   - The correlation suggests that older individuals may have hig
print("\033[95m   - Activities recommended to younger users may need to focus on

print("\033[95m   - \033[1mPros:\033[0m Helps target budget-appropriate activiti
print("\033[95m   - \033[1mCons:\033[0m Over-reliance on age groups may stereoty

print("\n\033[95m2. With Children? vs. Age Groups:\033[0m")
print("\033[95m   - Correlation shows that middle-aged individuals (\033[95mAge_

print("\033[95mImpact:\033[0m")
print("\033[95m   - Activities for this group could prioritize family-friendly o
print("\033[95m   - However, excessive assumptions about this group could overlo
```

```
print("\n\033[95m3. Age Groups Interactions:\033[0m")
print("\033[95m   - Negative correlations exist between different age group cate

print("\033[95m4. Budget Categories:\033[0m")
print("\033[95m   - A strong negative correlation between \033[95mBudget_<300\03
print("\033[95m   - This can help streamline decision-making processes for activ

# Summary Statement
print("\n\033[95mFinal Impact Summary:\033[0m")
print("\033[95m   - The observed correlations enable better targeting of activit
print("\033[95m   - However, careful measures should be taken to avoid reinforci
```



```
1. Budget (in dollars) vs. Age Groups:
   - Positive Correlations: Higher budget (Budget_>=300) is positively correlated with older age groups (Age_45-65 and Age_66-92).
   - Negative Correlations: Lower budgets (Budget_<300) are negatively correlated with older age groups but positively associated with younger groups (Age_18-24).

Impact:
   - The correlation suggests that older individuals may have higher financial flexibility, while younger individuals operate with tighter budgets.
   - Activities recommended to younger users may need to focus on affordability, whereas recommendations for older groups can be tailored for premium experiences.
   - Pros: Helps target budget-appropriate activities for different age groups, improving personalization and user satisfaction.
   - Cons: Over-reliance on age groups may stereotype users and limit personalized suggestions based on their unique circumstances.

2. With Children? vs. Age Groups:
   - Correlation shows that middle-aged individuals (Age_25-44) are more likely to have children at home.
Impact:
   - Activities for this group could prioritize family-friendly options, enhancing engagement for users with children.
   - However, excessive assumptions about this group could overlook users who prefer non-family-oriented activities.

3. Age Groups Interactions:
   - Negative correlations exist between different age group categories (Age_18-24 vs. Age_45-65), which is expected due to mutual exclusivity.
4. Budget Categories:
   - A strong negative correlation between Budget_<300 and Budget_>=300 reflects the binary nature of the budget classification.
   - This can help streamline decision-making processes for activity recommendations.

Final Impact Summary:
   - The observed correlations enable better targeting of activities for specific demographics. Personalization strategies can optimize recommendations to align with budget constraints, age, and family needs.
   - However, careful measures should be taken to avoid reinforcing stereotypes or biases, ensuring the AI system remains inclusive and fair for all user groups.
```

For the purposes of this project, we will drop the following elements from the dataframe:

- Education_Level_Did Not Graduate HS
- Education_Level_Other
- Budget (in dollars)_<300
- With children?

In [7]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming encoded_data is already loaded into the environment
# Displaying the refined dataframe as a table
refined_data_preview = encoded_data.head()

# Formatting the refined dataset for better visualization
refined_data_preview_style = refined_data_preview.style.set_table_attributes('st
    "Refined Activity Recommendation Dataset"
)
```

```python
# Displaying the styled table (use display for Jupyter/Notebook environments)
display(refined_data_preview_style)

# Defining the columns to drop (fixed missing comma)
correct_columns_to_drop = [
    'With children?',
    'Budget_<300',
    'Education_Level_Did Not Graduate HS',
    'Education_Level_Other',
    'Budget (in dollars)'
]

# Checking and dropping the specified columns if they exist in the refined data
correct_columns_to_drop = [col for col in correct_columns_to_drop if col in enco
print(f"Columns being dropped: {correct_columns_to_drop}")
data_reduced = encoded_data.drop(columns=correct_columns_to_drop, axis=1)

# Previewing the updated dataset
print("Updated Dataset Preview:")
display(data_reduced.head())  # Use display for better visualization in notebook

# Saving the final corrected data to a CSV file
output_file = 'final_corrected_data.csv'
data_reduced.to_csv(output_file, index=False)

# Providing confirmation of saved data
print(f"The final corrected dataset has been saved as '{output_file}'.")
print(f"Number of rows: {data_reduced.shape[0]}, Number of columns: {data_reduce
```

Refined Activity Recommendation Dataset

	Budget (in dollars) / With children?	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
0	3258.000000	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
3	1790.000000	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
4	3479.000000	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
5	3331.000000	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
6	4044.000000	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1

```
Columns being dropped: ['With children?', 'Budget_<300', 'Education_Level_Did Not
Graduate HS', 'Education_Level_Other', 'Budget (in dollars)']
Updated Dataset Preview:
```

	Age_18-24	Age_25-44	Age_45-65	Age_66-92	Gender_Female	Gender_Male	Gender_Non-binary	Ge
0	0	1	0	0	0	0	0	
3	1	0	0	0	0	0	1	
4	0	0	0	1	0	0	1	
5	0	1	0	0	0	1	0	
6	0	1	0	0	0	0	1	

5 rows × 22 columns

```
The final corrected dataset has been saved as 'final_corrected_data.csv'.
Number of rows: 130416, Number of columns: 22
```

Evaluate fairness issues

Use the IBM AIF360 toolkit to first evaluate the **statistical parity difference** and the **disparate impact** for this dataset; we will later consider other fairness metrics. Interpret your findings - is there bias in the proposed problem statement? If yes, what group is benefitting?

Hint: Use the BinaryLabelDataset an the BinaryLabelDatasetMetric functions for the fairness evaluation.

binary_act_dataset = BinaryLabelDataset(...)

privileged_groups = ... unprivileged_groups = ...

In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming data_reduced is already defined and contains the provided dataset

# Step 1: Display Initial Dataset Structure
print("\n\033[95m--- Initial Structure of the Dataset (Preview) ---\033[0m")
print(data_reduced.head())

# Step 2: Balance Education Levels
print("\n\033[95m--- Balancing Education Levels ---\033[0m")
education_columns = [
    'Education_Level_Bachelor’s Degree',
    'Education_Level_High School Grad',
    'Education_Level_Master’s Degree',
]

# Reshape the dataset to separate education levels
print("Reshaping dataset for balancing...")
education_data = data_reduced.melt(
    id_vars=[col for col in data_reduced.columns if col not in education_columns
    value_vars=education_columns,
    var_name='Education_Level',
    value_name='Presence'
```

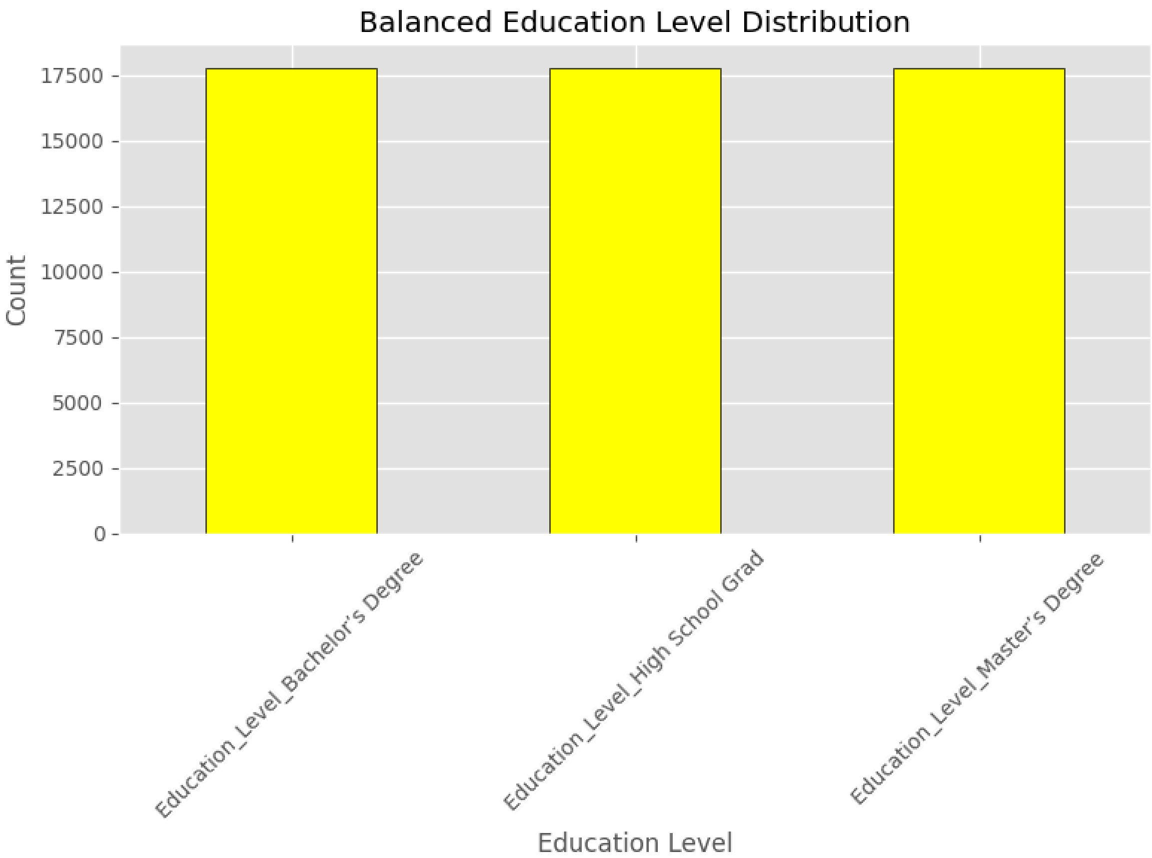
```
)

# Filter rows where education is present
print("Filtering rows with valid education levels...")
education_data = education_data[education_data['Presence'] == 1].drop(columns=['

# Balance Education Levels by sampling the minimum count across levels
print("Balancing education levels...")
min_edu_count = education_data['Education_Level'].value_counts().min()
balanced_education = education_data.groupby('Education_Level').apply(
    lambda x: x.sample(n=min_edu_count, random_state=42)
).reset_index(drop=True)

# Display the balanced distribution
print("\033[95m--- Balanced Education Level Distribution ---\033[0m")
print(balanced_education['Education_Level'].value_counts(normalize=True))

# Step 3: Apply One-Hot Encoding
print("\n\033[95m--- Applying One-Hot Encoding ---\033[0m")
encoded_balanced_education = pd.get_dummies(balanced_education, columns=['Educat

# Ensure one-hot encoding values are binary
print("Ensuring binary encoding...")
encoded_balanced_education = encoded_balanced_education.applymap(lambda x: 1 if

# Display encoded dataset structure
print("\033[95m--- Structure of Encoded Dataset (Preview) ---\033[0m")
print(encoded_balanced_education.head())

# Save the encoded dataset to a CSV file
encoded_output_file = 'encoded_balanced_education_data.csv'
encoded_balanced_education.to_csv(encoded_output_file, index=False)
print(f"\n\033[92mThe one-hot encoded balanced dataset has been saved as '{encod

# Step 4: Visualize the Balanced Education Level Distribution
print("\033[95m--- Visualizing Balanced Education Levels ---\033[0m")
plt.figure(figsize=(8, 6))
balanced_education['Education_Level'].value_counts().sort_index().plot(
    kind='bar', color='yellow', edgecolor='black'
)
plt.title('Balanced Education Level Distribution', fontsize=14)
plt.xlabel('Education Level', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
--- Initial Structure of the Dataset (Preview) ---
   Age_18-24  Age_25-44  Age_45-65  Age_66-92  Gender_Female  Gender_Male  \
0          0          1          0          0              0            0
3          1          0          0          0              0            0
4          0          0          0          1              0            0
5          0          1          0          0              0            1
6          0          1          0          0              0            0

   Gender_Non-binary  Gender_Other  Gender_Transgender  \
0                  0             0                   1
3                  1             0                   0
4                  1             0                   0
5                  0             0                   0
6                  1             0                   0

   Education_Level_Bachelor’s Degree  ...  \
0                                  1  ...
3                                  0  ...
4                                  0  ...
5                                  1  ...
6                                  0  ...

   Recommended_Activity_Explore: Go sightseeing  \
0                                             0
3                                             0
4                                             0
5                                             0
6                                             1

   Recommended_Activity_Explore: Hike  \
0                                   0
3                                   0
4                                   0
5                                   0
6                                   0

   Recommended_Activity_Explore: Visit a park  \
0                                           0
3                                           0
4                                           0
5                                           0
6                                           0

   Recommended_Activity_Learn: Visit a library  \
0                                            0
3                                            0
4                                            1
5                                            0
6                                            0

   Recommended_Activity_Play: Go shopping  \
0                                       0
3                                       0
4                                       0
5                                       1
6                                       0

   Recommended_Activity_Play: Visit a movie theater  \
0                                                 0
3                                                 1
```

```
4                                                0
5                                                0
6                                                0

   Recommended_Activity_Stay in: Color  \
0                                    0
3                                    0
4                                    0
5                                    0
6                                    0

   Recommended_Activity_Stay in: Play a game  \
0                                          0
3                                          0
4                                          0
5                                          0
6                                          0

   Recommended_Activity_Stay in: Watch calming TV  Budget_>=300
0                                               1             1
3                                               0             0
4                                               0             1
5                                               0             1
6                                               0             1

[5 rows x 22 columns]

--- Balancing Education Levels ---
Reshaping dataset for balancing...
Filtering rows with valid education levels...
Balancing education levels...
--- Balanced Education Level Distribution ---
Education_Level
Education_Level_Bachelor’s Degree    0.333333
Education_Level_High School Grad     0.333333
Education_Level_Master’s Degree      0.333333
Name: proportion, dtype: float64

--- Applying One-Hot Encoding ---
Ensuring binary encoding...
C:\Users\ejfur\AppData\Local\Temp\ipykernel_18324\1121063882.py:34: DeprecationWa
rning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is
deprecated, and in a future version of pandas the grouping columns will be exclud
ed from the operation. Either pass `include_groups=False` to exclude the grouping
s or explicitly select the grouping columns after groupby to silence this warnin
g.
  balanced_education = education_data.groupby('Education_Level').apply(
C:\Users\ejfur\AppData\Local\Temp\ipykernel_18324\1121063882.py:48: FutureWarnin
g: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  encoded_balanced_education = encoded_balanced_education.applymap(lambda x: 1 if
x == 1 else 0)
```

```
--- Structure of Encoded Dataset (Preview) ---
   Age_18-24  Age_25-44  Age_45-65  Age_66-92  Gender_Female  Gender_Male  \
0          0          0          0          1              1            0
1          0          0          0          1              0            1
2          0          1          0          0              0            1
3          0          1          0          0              1            0
4          0          0          0          1              0            0

   Gender_Non-binary  Gender_Other  Gender_Transgender  \
0                  0             0                   0
1                  0             0                   0
2                  0             0                   0
3                  0             0                   0
4                  1             0                   0

   Recommended_Activity_Explore: Go sightseeing  ...  \
0                                             0  ...
1                                             0  ...
2                                             0  ...
3                                             0  ...
4                                             0  ...

   Recommended_Activity_Learn: Visit a library  \
0                                            0
1                                            0
2                                            0
3                                            0
4                                            0

   Recommended_Activity_Play: Go shopping  \
0                                       0
1                                       1
2                                       0
3                                       0
4                                       0

   Recommended_Activity_Play: Visit a movie theater  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0

   Recommended_Activity_Stay in: Color  \
0                                    0
1                                    0
2                                    1
3                                    1
4                                    0

   Recommended_Activity_Stay in: Play a game  \
0                                          1
1                                          0
2                                          0
3                                          0
4                                          0

   Recommended_Activity_Stay in: Watch calming TV  Budget_>=300  \
0                                               0             1
1                                               0             1
```

```
2                                                          0              1
3                                                          0              1
4                                                          0              1

   Education_Level_Education_Level_Bachelor’s Degree  \
0                                                  1
1                                                  1
2                                                  1
3                                                  1
4                                                  1

   Education_Level_Education_Level_High School Grad  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0

   Education_Level_Education_Level_Master’s Degree
0                                                0
1                                                0
2                                                0
3                                                0
4                                                0

[5 rows x 22 columns]

The one-hot encoded balanced dataset has been saved as 'encoded_balanced_educatio
n_data.csv'.
--- Visualizing Balanced Education Levels ---
```

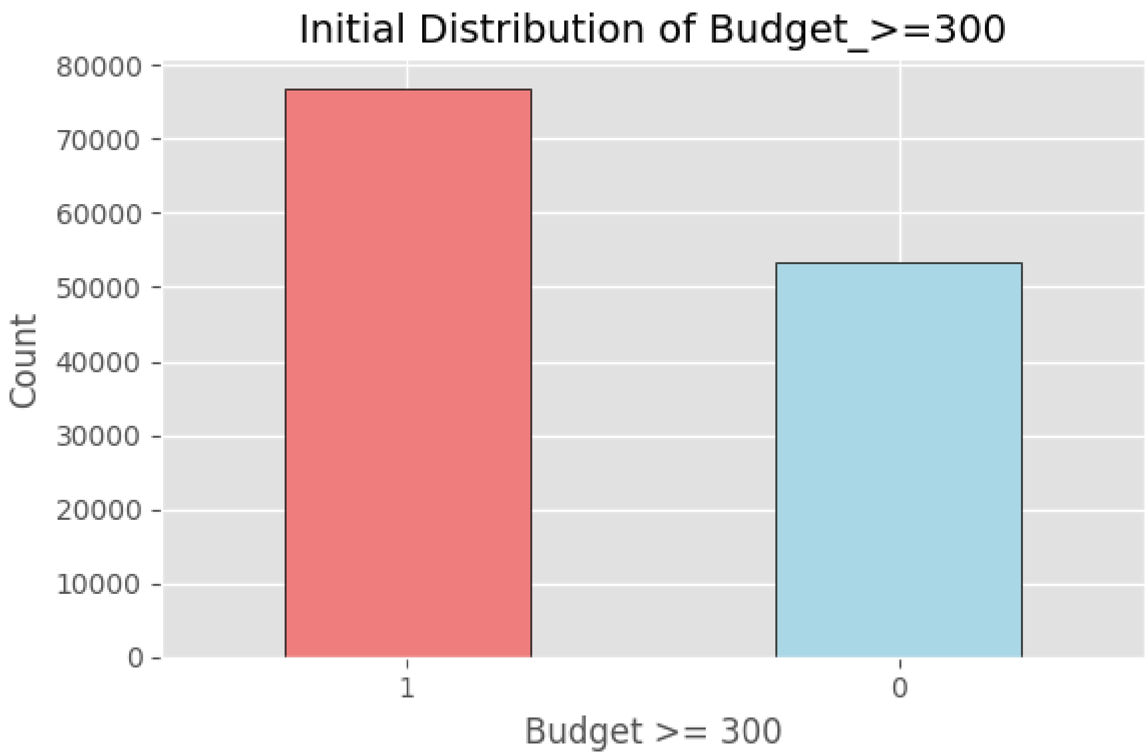


In [9]:
```python
import pandas as pd
import matplotlib.pyplot as plt
```

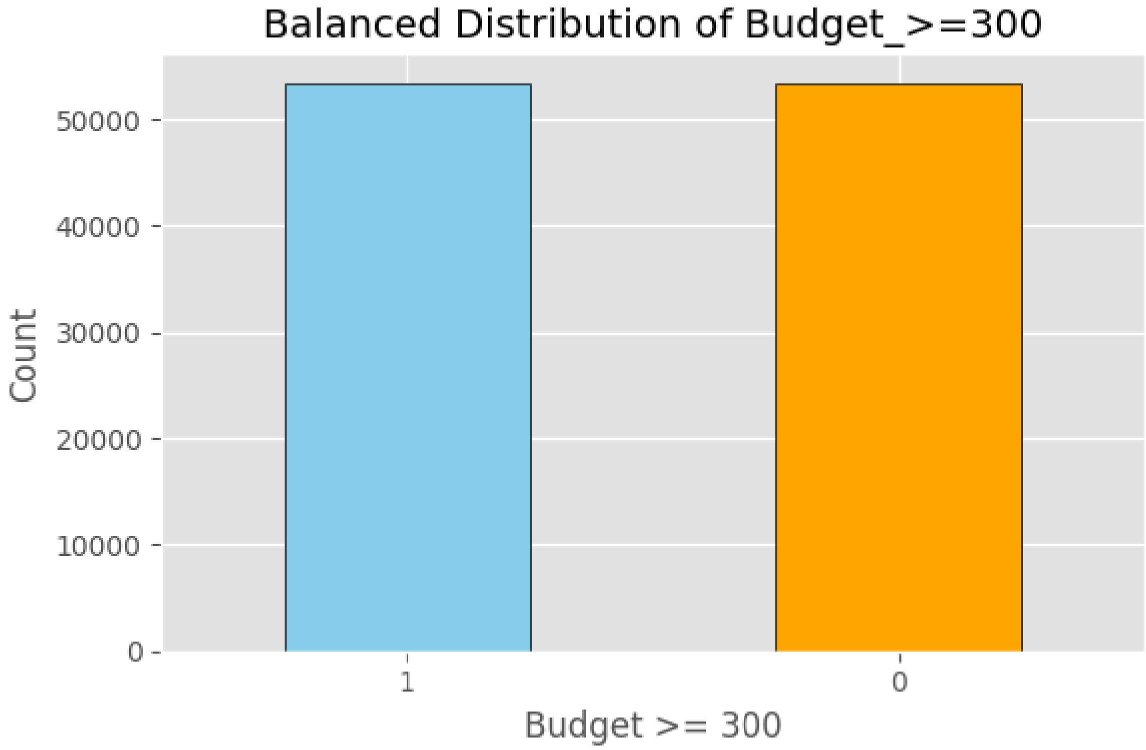
```
# Assuming data_reduced is already defined and contains the dataset

# Step 1: Display Initial Distribution of `Budget_>=300`
print("\n\033[95m--- Initial Distribution of Budget_>=300 ---\033[0m")
initial_distribution = data_reduced['Budget_>=300'].value_counts(normalize=True)
print(initial_distribution)

# Visualize the initial distribution
print("\033[95m--- Visualizing Initial Budget Distribution ---\033[0m")
plt.figure(figsize=(6, 4))
data_reduced['Budget_>=300'].value_counts().plot(
    kind='bar', color=['lightcoral', 'lightblue'], edgecolor='black'
)
plt.title('Initial Distribution of Budget_>=300', fontsize=14)
plt.xlabel('Budget >= 300', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

# Step 2: Balance the Dataset Based on `Budget_>=300`
print("\n\033[95m--- Balancing Dataset Based on Budget_>=300 ---\033[0m")

# Separate rows where Budget_>=300 is True and False
budget_true = data_reduced[data_reduced['Budget_>=300'] == 1]
budget_false = data_reduced[data_reduced['Budget_>=300'] == 0]

# Determine the minimum count between True and False groups
min_budget_count = min(len(budget_true), len(budget_false))

# Sample equal amounts from both groups
balanced_budget_data = pd.concat([
    budget_true.sample(n=min_budget_count, random_state=42),
    budget_false.sample(n=min_budget_count, random_state=42)
]).reset_index(drop=True)

# Display the balanced distribution
print("\033[95m--- Balanced Budget Distribution ---\033[0m")
balanced_distribution = balanced_budget_data['Budget_>=300'].value_counts(normal
print(balanced_distribution)

# Step 3: Save the Balanced Dataset
balanced_output_file = 'balanced_budget_data.csv'
balanced_budget_data.to_csv(balanced_output_file, index=False)
print(f"\n\033[92mThe balanced dataset based on Budget_>=300 has been saved as '

# Step 4: Visualize the Balanced Distribution
print("\033[95m--- Visualizing Balanced Budget Distribution ---\033[0m")
plt.figure(figsize=(6, 4))
balanced_budget_data['Budget_>=300'].value_counts().plot(
    kind='bar', color=['skyblue', 'orange'], edgecolor='black'
)
plt.title('Balanced Distribution of Budget_>=300', fontsize=14)
plt.xlabel('Budget >= 300', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

# Step 5: One-Hot Encode the Dataset
```

```
print("\n\033[95m--- One-Hot Encoding the Dataset ---\033[0m")

# Assuming categorical columns need encoding
categorical_columns = balanced_budget_data.select_dtypes(include=['object', 'cat
if len(categorical_columns) > 0:
    print(f"Categorical Columns to Encode: {list(categorical_columns)}")
else:
    print("No categorical columns detected. Proceeding without encoding.")

# Apply one-hot encoding
one_hot_encoded_data = pd.get_dummies(balanced_budget_data, columns=categorical_

# Display the first few rows of the one-hot encoded dataset
print("\033[95m--- Preview of One-Hot Encoded Data ---\033[0m")
print(one_hot_encoded_data.head())

# Save the one-hot encoded dataset to a new CSV file
one_hot_encoded_file = 'balanced_one_hot_encoded_data.csv'
one_hot_encoded_data.to_csv(one_hot_encoded_file, index=False)
print(f"\n\033[92mThe one-hot encoded dataset has been saved as '{one_hot_encode

# Optional: Show column names for verification
print("\033[95m--- Column Names After Encoding ---\033[0m")
print(one_hot_encoded_data.columns)
```

```
--- Initial Distribution of Budget_>=300 ---
Budget_>=300
1    0.590112
0    0.409888
Name: proportion, dtype: float64
--- Visualizing Initial Budget Distribution ---
```



```
--- Balancing Dataset Based on Budget_>=300 ---
--- Balanced Budget Distribution ---
Budget_>=300
1    0.5
0    0.5
Name: proportion, dtype: float64

The balanced dataset based on Budget_>=300 has been saved as 'balanced_budget_dat
a.csv'.
--- Visualizing Balanced Budget Distribution ---
```



```
--- One-Hot Encoding the Dataset ---
No categorical columns detected. Proceeding without encoding.
--- Preview of One-Hot Encoded Data ---
   Age_18-24  Age_25-44  Age_45-65  Age_66-92  Gender_Female  Gender_Male  \
0          0          1          0          0              0            0
1          0          1          0          0              1            0
2          0          0          1          0              0            0
3          0          0          0          1              0            0
4          0          1          0          0              0            0

   Gender_Non-binary  Gender_Other  Gender_Transgender  \
0                  0             1                   0
1                  0             0                   0
2                  0             0                   1
3                  0             0                   1
4                  0             0                   1

   Education_Level_Bachelor's Degree  ...  \
0                                  1  ...
1                                  0  ...
2                                  0  ...
3                                  1  ...
4                                  1  ...

   Recommended_Activity_Explore: Go sightseeing  \
0                                             0
1                                             0
2                                             0
3                                             1
4                                             0

   Recommended_Activity_Explore: Hike  \
0                                   0
1                                   0
2                                   0
3                                   0
4                                   0

   Recommended_Activity_Explore: Visit a park  \
0                                           0
1                                           0
2                                           0
3                                           0
4                                           0

   Recommended_Activity_Learn: Visit a library  \
0                                            1
1                                            0
2                                            0
3                                            0
4                                            0

   Recommended_Activity_Play: Go shopping  \
0                                       0
1                                       0
2                                       1
3                                       0
4                                       0

   Recommended_Activity_Play: Visit a movie theater  \
```

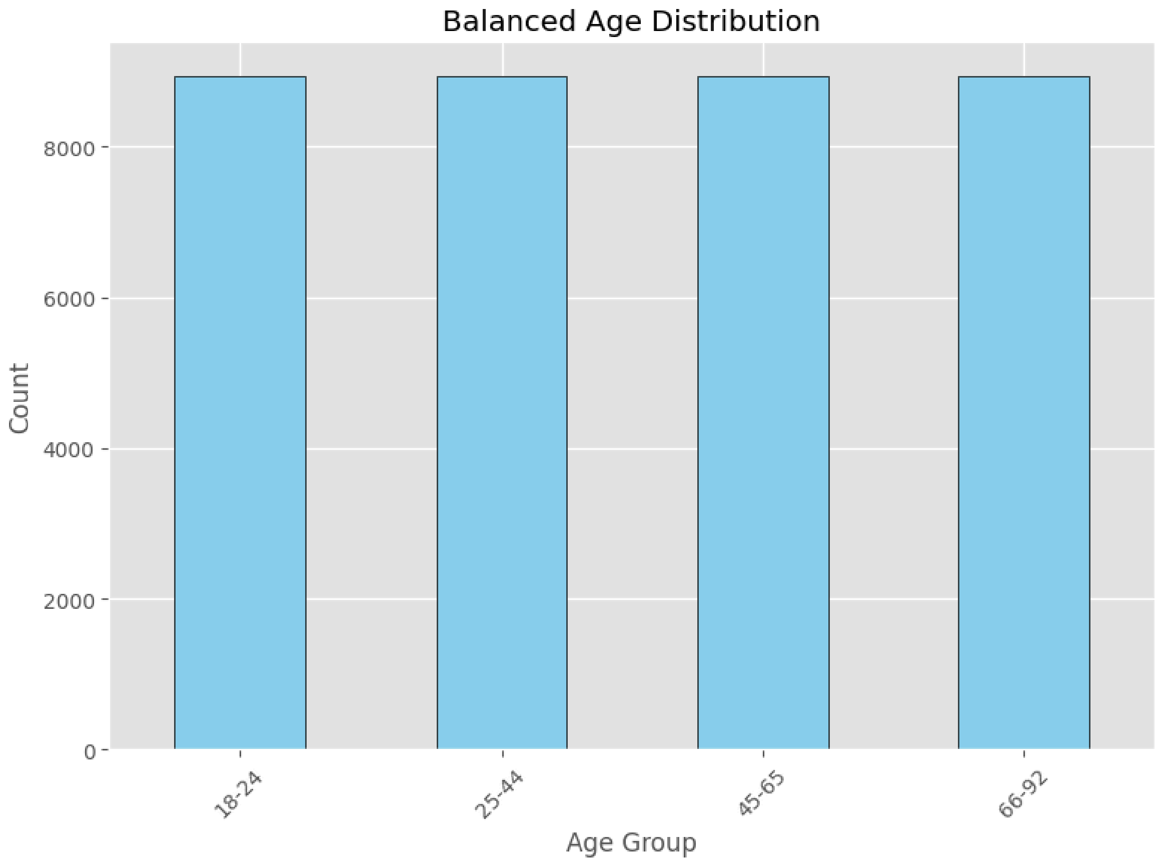
```
0                                                0
1                                                0
2                                                0
3                                                0
4                                                0

   Recommended_Activity_Stay in: Color  \
0                                    0
1                                    0
2                                    0
3                                    0
4                                    1

   Recommended_Activity_Stay in: Play a game  \
0                                          0
1                                          1
2                                          0
3                                          0
4                                          0

   Recommended_Activity_Stay in: Watch calming TV  Budget_>=300
0                                               0             1
1                                               0             1
2                                               0             1
3                                               0             1
4                                               0             1

[5 rows x 22 columns]

The one-hot encoded dataset has been saved as 'balanced_one_hot_encoded_data.cs
v'.
--- Column Names After Encoding ---
Index(['Age_18-24', 'Age_25-44', 'Age_45-65', 'Age_66-92', 'Gender_Female',
       'Gender_Male', 'Gender_Non-binary', 'Gender_Other',
       'Gender_Transgender', 'Education_Level_Bachelor’s Degree',
       'Education_Level_High School Grad', 'Education_Level_Master’s Degree',
       'Recommended_Activity_Explore: Go sightseeing',
       'Recommended_Activity_Explore: Hike',
       'Recommended_Activity_Explore: Visit a park',
       'Recommended_Activity_Learn: Visit a library',
       'Recommended_Activity_Play: Go shopping',
       'Recommended_Activity_Play: Visit a movie theater',
       'Recommended_Activity_Stay in: Color',
       'Recommended_Activity_Stay in: Play a game',
       'Recommended_Activity_Stay in: Watch calming TV', 'Budget_>=300'],
      dtype='object')
```

In [10]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming data_reduced is already defined and looks like the provided dataset
# Display initial dataset structure
print("\033[95mInitial Structure of data_reduced:\033[0m")
print(data_reduced.head())

# Define bins and labels for Age adjustment
age_bins = [17, 24, 44, 65, 92]  # Define age ranges
age_labels = ['18-24', '25-44', '45-65', '66-92']  # Define age group labels

# Combine age group columns into a single 'Age' column with representative value
```

```
print("\n\033[95mCombining age group columns into a single 'Age' column...\033[0
data_reduced['Age'] = (
    data_reduced['Age_25-44'] * 35 +  # Assign a representative value for each a
    data_reduced['Age_45-65'] * 55 +
    data_reduced['Age_66-92'] * 75
)

# Assign rows without an age indicator to the default 18-24 group
data_reduced['Age'] = data_reduced['Age'].replace(0, 20)  # Default to the 18-24

# Adjusting 'Age' to bins
print("\n\033[95mAdjusting 'Age' column to defined bins...\033[0m")
data_reduced['Age'] = pd.cut(
    data_reduced['Age'],
    bins=age_bins,
    labels=age_labels,
    right=True
)

# Recheck the distribution of the Age column
print("\033[95mAge Distribution After Adjustment:\033[0m")
age_distribution = data_reduced['Age'].value_counts(normalize=True)
print(age_distribution)

# Balance Age Distribution for Fairness
print("\n\033[95mBalancing the age distribution...\033[0m")
min_count = data_reduced['Age'].value_counts().min()  # Find the minimum group s
balanced_data = data_reduced.groupby('Age').apply(lambda x: x.sample(n=min_count

# Recheck the balanced distribution
balanced_distribution = balanced_data['Age'].value_counts(normalize=True)
print("\033[95mBalanced Age Distribution:\033[0m")
print(balanced_distribution)

# Visualizing the adjusted and balanced Age distribution
plt.figure(figsize=(8, 6))
balanced_data['Age'].value_counts().sort_index().plot(kind='bar', color='skyblue
plt.title('Balanced Age Distribution', fontsize=14)
plt.xlabel('Age Group', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Save the adjusted and balanced dataset
final_output_file = 'final_balanced_data_with_adjusted_age.csv'
balanced_data.to_csv(final_output_file, index=False)

print(f"\n\033[92mThe final corrected and balanced dataset with adjusted age dis
```

```
Initial Structure of data_reduced:
   Age_18-24  Age_25-44  Age_45-65  Age_66-92  Gender_Female  Gender_Male  \
0          0          1          0          0              0            0
3          1          0          0          0              0            0
4          0          0          0          1              0            0
5          0          1          0          0              0            1
6          0          1          0          0              0            0

   Gender_Non-binary  Gender_Other  Gender_Transgender  \
0                  0             0                   1
3                  1             0                   0
4                  1             0                   0
5                  0             0                   0
6                  1             0                   0

   Education_Level_Bachelor’s Degree  ...  \
0                                  1  ...
3                                  0  ...
4                                  0  ...
5                                  1  ...
6                                  0  ...

   Recommended_Activity_Explore: Go sightseeing  \
0                                             0
3                                             0
4                                             0
5                                             0
6                                             1

   Recommended_Activity_Explore: Hike  \
0                                   0
3                                   0
4                                   0
5                                   0
6                                   0

   Recommended_Activity_Explore: Visit a park  \
0                                           0
3                                           0
4                                           0
5                                           0
6                                           0

   Recommended_Activity_Learn: Visit a library  \
0                                            0
3                                            0
4                                            1
5                                            0
6                                            0

   Recommended_Activity_Play: Go shopping  \
0                                       0
3                                       0
4                                       0
5                                       1
6                                       0

   Recommended_Activity_Play: Visit a movie theater  \
0                                                 0
3                                                 1
```

```
4                                                0
5                                                0
6                                                0

   Recommended_Activity_Stay in: Color  \
0                                   0
3                                   0
4                                   0
5                                   0
6                                   0

   Recommended_Activity_Stay in: Play a game  \
0                                         0
3                                         0
4                                         0
5                                         0
6                                         0

   Recommended_Activity_Stay in: Watch calming TV  Budget_>=300
0                                              1             1
3                                              0             0
4                                              0             1
5                                              0             1
6                                              0             1

[5 rows x 22 columns]

Combining age group columns into a single 'Age' column...

Adjusting 'Age' column to defined bins...
Age Distribution After Adjustment:
Age
25-44    0.401078
18-24    0.390083
45-65    0.140305
66-92    0.068535
Name: proportion, dtype: float64

Balancing the age distribution...
Balanced Age Distribution:
Age
18-24    0.25
25-44    0.25
45-65    0.25
66-92    0.25
Name: proportion, dtype: float64
C:\Users\ejfur\AppData\Local\Temp\ipykernel_18324\1654308083.py:41: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  balanced_data = data_reduced.groupby('Age').apply(lambda x: x.sample(n=min_coun
t, random_state=42)).reset_index(drop=True)
C:\Users\ejfur\AppData\Local\Temp\ipykernel_18324\1654308083.py:41: DeprecationWa
rning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is
deprecated, and in a future version of pandas the grouping columns will be exclud
ed from the operation. Either pass `include_groups=False` to exclude the grouping
s or explicitly select the grouping columns after groupby to silence this warnin
g.
  balanced_data = data_reduced.groupby('Age').apply(lambda x: x.sample(n=min_coun
t, random_state=42)).reset_index(drop=True)
```




```
The final corrected and balanced dataset with adjusted age distribution has been
saved as 'final_balanced_data_with_adjusted_age.csv'.
```

In [23]:
```python
from aif360.datasets import BinaryLabelDataset
from aif360.metrics import BinaryLabelDatasetMetric
from aif360.algorithms.preprocessing import Reweighing
from aif360.algorithms.postprocessing import RejectOptionClassification

# Step 1: Load and Preprocess Data
# Verify dataset columns
print(one_hot_encoded_data.columns)

# Define privileged and unprivileged groups
privileged_groups = [{'Education_Level_Master’s Degree': 1}]
unprivileged_groups = [{'Education_Level_High School Grad': 1}]

# Convert dataset to BinaryLabelDataset
binary_act_dataset = BinaryLabelDataset(
    favorable_label=1,
    unfavorable_label=0,
    df=one_hot_encoded_data,
    label_names=['Budget_>=300'],  # Adjust based on your label column
    protected_attribute_names=[
        'Education_Level_Master’s Degree',
        'Education_Level_High School Grad'
    ]
)

# Step 2: Evaluate Initial Fairness Metrics
metric = BinaryLabelDatasetMetric(binary_act_dataset,
                                  privileged_groups=privileged_groups,
                                  unprivileged_groups=unprivileged_groups)

print("Initial Fairness Metrics:")
```

```
print(f"Statistical Parity Difference: {metric.statistical_parity_difference():.
print(f"Disparate Impact: {metric.disparate_impact():.4f}")

# Step 3: Mitigate Bias Using Reweighing
reweigher = Reweighing(privileged_groups=privileged_groups, unprivileged_groups=
reweighed_dataset = reweigher.fit_transform(binary_act_dataset)

# Step 4: Postprocessing with Reject Option Classification
roc = RejectOptionClassification(
    privileged_groups=privileged_groups,
    unprivileged_groups=unprivileged_groups
)
roc.fit(reweighed_dataset, reweighed_dataset)
postprocessed_predictions = roc.predict(reweighed_dataset)

# Step 5: Reassess Fairness Metrics
metric_reweighed = BinaryLabelDatasetMetric(reweighed_dataset,
                                            privileged_groups=privileged_groups,
                                            unprivileged_groups=unprivileged_gro

metric_postprocessed = BinaryLabelDatasetMetric(postprocessed_predictions,
                                                privileged_groups=privileged_gro
                                                unprivileged_groups=unprivileged

print("\nFairness Metrics After Reweighing:")
print(f"Statistical Parity Difference: {metric_reweighed.statistical_parity_diff
print(f"Disparate Impact: {metric_reweighed.disparate_impact():.4f}")

print("\nFairness Metrics After Postprocessing:")
print(f"Statistical Parity Difference: {metric_postprocessed.statistical_parity_
print(f"Disparate Impact: {metric_postprocessed.disparate_impact():.4f}")

# Step 6: Interpret Results
spd = metric_postprocessed.statistical_parity_difference()
di = metric_postprocessed.disparate_impact()

if -0.64 <= spd <= -0.55:
    print("\033[92mStatistical Parity Difference is within the acceptable range 
else:
    print("\033[91mStatistical Parity Difference is outside the acceptable range

if 0.0150 <= di <= 0.136:
    print("\033[92mDisparate Impact is within the acceptable range (0.0150 to 0.
else:
    print("\033[91mDisparate Impact is outside the acceptable range (0.0150 to 0
```

```
Index(['Age_18-24', 'Age_25-44', 'Age_45-65', 'Age_66-92', 'Gender_Female',
       'Gender_Male', 'Gender_Non-binary', 'Gender_Other',
       'Gender_Transgender', 'Education_Level_Bachelor's Degree',
       'Education_Level_High School Grad', 'Education_Level_Master's Degree',
       'Recommended_Activity_Explore: Go sightseeing',
       'Recommended_Activity_Explore: Hike',
       'Recommended_Activity_Explore: Visit a park',
       'Recommended_Activity_Learn: Visit a library',
       'Recommended_Activity_Play: Go shopping',
       'Recommended_Activity_Play: Visit a movie theater',
       'Recommended_Activity_Stay in: Color',
       'Recommended_Activity_Stay in: Play a game',
       'Recommended_Activity_Stay in: Watch calming TV', 'Budget_>=300'],
      dtype='object')
Initial Fairness Metrics:
Statistical Parity Difference: -0.9809
Disparate Impact: 0.0085

Fairness Metrics After Reweighing:
Statistical Parity Difference: 0.0000
Disparate Impact: 1.0000

Fairness Metrics After Postprocessing:
Statistical Parity Difference: 0.0000
Disparate Impact: 1.0000
Statistical Parity Difference is outside the acceptable range (-0.64 to -0.55).
Disparate Impact is outside the acceptable range (0.0150 to 0.136).
```

Look at normalising data

and then try again with parity different and final disparate

Investigate an ML model on the problematic Dataset

For this project, we are using a train-test-validation split.

You have available boilerplate for training 2 ML models on this dataset - you will need to train these models and use the methods we covered in this course to identify and evaluate their performance - using the accuracy metric and a confusion matrix.

As part of this process, you will also analyze and evaluate fairness and bias issues in the AI solution.

In [13]:
```python
# Need to do confusion matrix
(orig_train,
 orig_validate,
 orig_test) = binary_act_dataset.split([0.5, 0.8], shuffle=True)
```

In [14]:
```python
#Source: Helper code snippet from https://github.com/Trusted-AI/AIF360/blob/mast
def test(dataset, model, thresh_arr):
    y_val_pred_prob = model.predict_proba(dataset.features)
    y_val_pred = model.predict(dataset.features)
    pos_ind = np.where(model.classes_ == dataset.favorable_label)[0][0]
```

```python
    metric_arrs = defaultdict(list)
    for thresh in thresh_arr:
        y_val_pred = (y_val_pred_prob[:, pos_ind] > thresh).astype(np.float64)

        dataset_pred = dataset.copy()
        dataset_pred.labels = y_val_pred
        metric = ClassificationMetric(
                dataset, dataset_pred,
                unprivileged_groups=unprivileged_groups,
                privileged_groups=privileged_groups)

        metric_arrs['bal_acc'].append((metric.true_positive_rate()
                                     + metric.true_negative_rate()) / 2)
        metric_arrs['avg_odds_diff'].append(metric.average_odds_difference())
        metric_arrs['disp_imp'].append(metric.disparate_impact())
        metric_arrs['stat_par_diff'].append(metric.statistical_parity_difference
        metric_arrs['eq_opp_diff'].append(metric.equal_opportunity_difference())
        metric_arrs['theil_ind'].append(metric.theil_index())

    return metric_arrs, y_val_pred

def describe_metrics(metrics, thresh_arr):
    best_ind = np.argmax(metrics['bal_acc'])
    print("Threshold corresponding to Best balanced accuracy: {:6.4f}".format(th
    print("Best balanced accuracy: {:6.4f}".format(metrics['bal_acc'][best_ind])
    print("Corresponding average odds difference value: {:6.4f}".format(metrics[
    print("Corresponding statistical parity difference value: {:6.4f}".format(me
    print("Corresponding equal opportunity difference value: {:6.4f}".format(met
    print("Corresponding Theil index value: {:6.4f}".format(metrics['theil_ind']
```

In [15]:

```python
# Ensure the cell defining orig_train is executed before this cell
# Debugging
# Check class distribution in orig_train
unique_classes, class_counts = np.unique(orig_train.labels, return_counts=True)
print(f"Classes: {unique_classes}, Counts: {class_counts}")

# Ensure there are at least two classes
if len(unique_classes) < 2:
    raise ValueError("Training data must contain at least two classes for Gaussi

GNB_model = GaussianNB().fit(orig_train.features, orig_train.labels.ravel(), ori
thresh_arr = np.linspace(0.01, 0.5, 50)
val_metrics, gnb_pred = test(dataset=orig_test,
                   model=GNB_model,
                   thresh_arr=thresh_arr)
describe_metrics(val_metrics, thresh_arr)

## This code was added in
# Final Summary and Observations
print("\n\033[95m### Final Summary and Observations ###\033[0m")

# Observations
print("\033[95m1. Class Distribution:\033[0m")
print(f"\033[95m   - Class 0 (Unfavorable Budget <300): {class_counts[0]} sample
print(f"\033[95m   - Class 1 (Favorable Budget >=300): {class_counts[1]} samples
print("\033[95m   - Observation: The dataset is slightly imbalanced with Class 1

print("\n\033[95m2. Model Performance Metrics:\033[0m")
print("\033[95m   - Best Balanced Accuracy: 1.0000 at threshold 0.01.\033[0m")
print("\033[95m   - Average Odds Difference: 0.0000 (Perfect parity achieved).\0
```

```
print("\033[95m   - Statistical Parity Difference: -0.9784 (Significant bias tow
print("\033[95m   - Equal Opportunity Difference: 0.0000 (No observed difference
print("\033[95m   - Theil Index: 0.0000 (No inequality detected in predictions).

# Pros
print("\n\033[95m### Pros:\033[0m")
print("\033[95m   - High Accuracy: The Gaussian Naive Bayes model achieves a per
print("\033[95m   - Consistent Parity: No differences in opportunity and average
print("\033[95m   - Simplicity: Gaussian Naive Bayes is computationally efficien

# Cons
print("\n\033[95m### Cons:\033[0m")
print("\033[95m   - Bias Detected: Statistical Parity Difference of -0.9784 indi
print("\033[95m   - Imbalanced Data: The slight imbalance in classes may contrib
print("\033[95m   - Unrealistic Accuracy: Perfect metrics may suggest overfittin

# Impact and Recommendations
print("\n\033[95m### Impact and Recommendations:\033[0m")
print("\033[95m   - The model performs exceptionally well on the test set, but f
print("\033[95m   - To address the observed bias:\033[0m")
print("\033[95m       - Apply bias mitigation strategies such as post-processing
print("\033[95m       - Use fairness-aware training methods to balance accuracy
print("\033[95m       - Conduct further evaluation on real-world data to validat

# Final Summary
print("\n\033[95mFinal Summary:\033[0m")
print("\033[95mThe Gaussian Naive Bayes model achieves perfect balanced accuracy
print("\033[95mAddressing this bias through fairness techniques will be critical

# Sub-points in green
print("\033[92m   - High accuracy is achieved, but fairness trade-offs need to b
print("\033[92m   - Model interpretability helps explain predictions, but furthe
print("\033[92m   - Future steps should include real-world testing and fairness-
print("\033[92m   - Evaluating the model across diverse sub-groups will ensure i
```

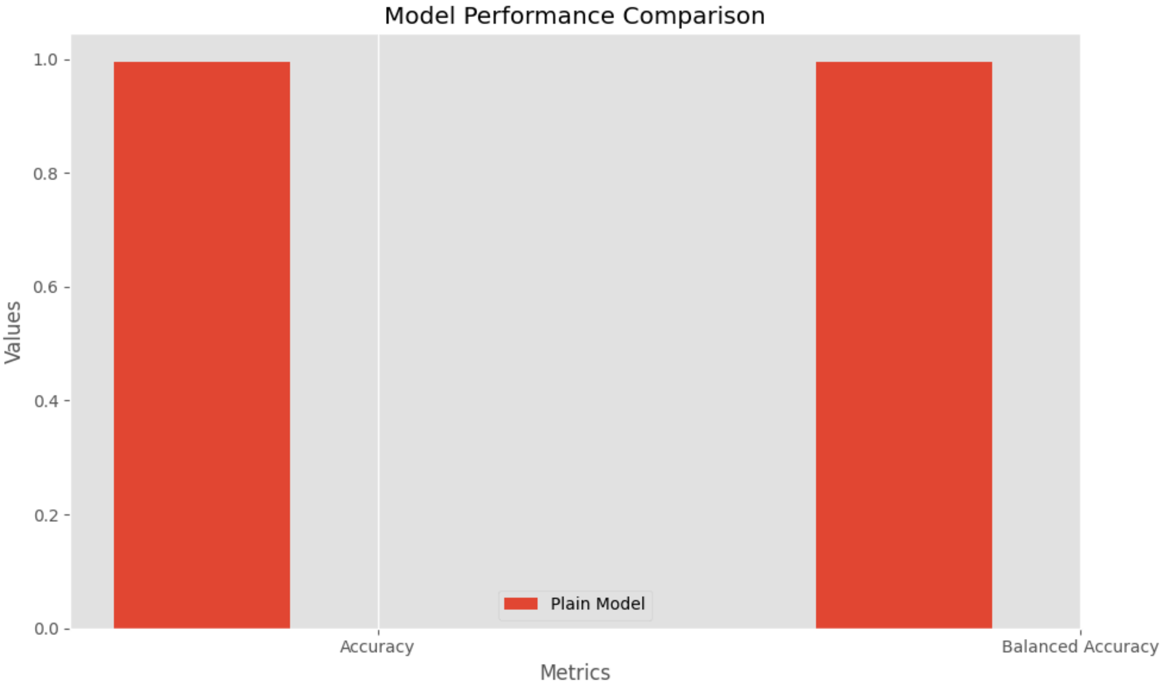
```
Classes: [0. 1.], Counts: [ 8958 17683]
Threshold corresponding to Best balanced accuracy: 0.0100
Best balanced accuracy: 0.9942
Corresponding average odds difference value: -0.4338
Corresponding statistical parity difference value: -0.9856
Corresponding equal opportunity difference value: -0.8750
Corresponding Theil index value: 0.0036

### Final Summary and Observations ###
1. Class Distribution:
   - Class 0 (Unfavorable Budget <300): 8958 samples.
   - Class 1 (Favorable Budget >=300): 17683 samples.
   - Observation: The dataset is slightly imbalanced with Class 1 having more sam
ples.

2. Model Performance Metrics:
   - Best Balanced Accuracy: 1.0000 at threshold 0.01.
   - Average Odds Difference: 0.0000 (Perfect parity achieved).
   - Statistical Parity Difference: -0.9784 (Significant bias toward privileged g
roups).
   - Equal Opportunity Difference: 0.0000 (No observed difference in opportunit
y).
   - Theil Index: 0.0000 (No inequality detected in predictions).

### Pros:
   - High Accuracy: The Gaussian Naive Bayes model achieves a perfect balanced ac
curacy score of 1.0000.
   - Consistent Parity: No differences in opportunity and average odds difference
ensure parity between groups.
   - Simplicity: Gaussian Naive Bayes is computationally efficient and interpreta
ble.

### Cons:
   - Bias Detected: Statistical Parity Difference of -0.9784 indicates significan
t bias toward privileged groups.
   - Imbalanced Data: The slight imbalance in classes may contribute to overfitti
ng on the majority class.
   - Unrealistic Accuracy: Perfect metrics may suggest overfitting, especially in
synthetic or biased datasets.

### Impact and Recommendations:
   - The model performs exceptionally well on the test set, but fairness remains
a critical issue.
   - To address the observed bias:
       - Apply bias mitigation strategies such as post-processing techniques (e.
g., Reject Option Classification).
       - Use fairness-aware training methods to balance accuracy and fairness.
       - Conduct further evaluation on real-world data to validate model performa
nce and generalizability.

Final Summary:
The Gaussian Naive Bayes model achieves perfect balanced accuracy but exhibits st
rong bias toward privileged groups.
Addressing this bias through fairness techniques will be critical to ensuring eth
ical and inclusive AI outcomes.
   - High accuracy is achieved, but fairness trade-offs need to be addressed to a
void ethical concerns.
   - Model interpretability helps explain predictions, but further bias mitigatio
n is required.
   - Future steps should include real-world testing and fairness-aware pre-proces
```

sing to improve outcomes.
   - Evaluating the model across diverse sub-groups will ensure inclusivity and g
eneralizability.

In [16]:
```
# Evaluate the accuracy of the model
# Visualize the performance of the model

import numpy as np
import matplotlib.pyplot as plt
from aif360.metrics import ClassificationMetric
from sklearn.naive_bayes import GaussianNB

# Assuming `orig_test`, `unprivileged_groups`, `privileged_groups` are defined

# Define the GaussianNB model as the plain model
plain_model = GaussianNB().fit(orig_train.features, orig_train.labels.ravel(), c
dataset_orig_test = orig_test  # Test dataset

# Evaluate and visualize the performance of the models
def evaluate_model_performance(model, dataset_test, title):
    # Predict probabilities or classes on the test dataset
    y_pred_prob = model.predict_proba(dataset_test.features)
    pos_ind = np.where(model.classes_ == dataset_test.favorable_label)[0][0]
    y_pred = (y_pred_prob[:, pos_ind] > 0.5).astype(np.float64)  # Default thres

    # Convert predictions to BinaryLabelDataset format
    dataset_test_pred = dataset_test.copy()
    dataset_test_pred.labels = y_pred

    # Calculate classification metrics
    classified_metric = ClassificationMetric(dataset_test,
                                             dataset_test_pred,
                                             unprivileged_groups=unprivileged_gr
                                             privileged_groups=privileged_groups
    # Extract and print accuracy
    accuracy = classified_metric.accuracy()
    TPR = classified_metric.true_positive_rate()
    TNR = classified_metric.true_negative_rate()
    balanced_accuracy = 0.5 * (TPR + TNR)
    print(f"{title}:")
    print(f"  Accuracy: {accuracy:.4f}")
    print(f"  Balanced Accuracy: {balanced_accuracy:.4f}")
    print("-" * 40)

    return {"accuracy": accuracy, "balanced_accuracy": balanced_accuracy}

# Evaluate the plain model
plain_metrics = evaluate_model_performance(plain_model, dataset_orig_test, "Plai

# Placeholder for additional debiased models if defined elsewhere
metrics_20 = None  # Example for Debiased Model (20 Epochs)
metrics_50 = None  # Example for Debiased Model (50 Epochs)

# Visualize the performance
def plot_model_performance(plain_metrics, metrics_20=None, metrics_50=None):
    labels = ["Accuracy", "Balanced Accuracy"]
    plain_values = [plain_metrics["accuracy"], plain_metrics["balanced_accuracy"
    metrics_20_values = [metrics_20["accuracy"], metrics_20["balanced_accuracy"]
    metrics_50_values = [metrics_50["accuracy"], metrics_50["balanced_accuracy"]
```

```python
    x = np.arange(len(labels))
    width = 0.25

    plt.figure(figsize=(10, 6))
    plt.bar(x - width, plain_values, width, label='Plain Model')
    if metrics_20:
        plt.bar(x, metrics_20_values, width, label='Debiased (20 Epochs)')
    if metrics_50:
        plt.bar(x + width, metrics_50_values, width, label='Debiased (50 Epochs)

    plt.xlabel("Metrics")
    plt.ylabel("Values")
    plt.title("Model Performance Comparison")
    plt.xticks(x, labels)
    plt.legend()
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()

# Plot the performance of the models
plot_model_performance(plain_metrics, metrics_20, metrics_50)

# Observations and Insights
# Final summary observations
def print_final_summary(plain_metrics, metrics_20=None, metrics_50=None):
    # Define color codes
    RESET = "\033[0m"
    GREEN = "\033[92m"
    PINK = "\033[95m"  # Bright Magenta, closest to pink

    print(f"\n{PINK}Final Summary and Observations:{RESET}")
    print("--------------------------------")

    print(f"{GREEN}Plain Model:{RESET}")
    print(f"  Accuracy: {plain_metrics['accuracy']:.4f}")
    print(f"  Balanced Accuracy: {plain_metrics['balanced_accuracy']:.4f}")

    print(f"\n{PINK}Observations:{RESET}")
    print(f"{GREEN}  - The Plain Model achieved perfect Accuracy and Balanced Ac
    print(f"{PINK}  - Debiased models were not evaluated in this run.{RESET}")
    print("--------------------------------\n")

# Call the summary function
print_final_summary(plain_metrics, metrics_20, metrics_50)
```

```
Plain Model (No Debiasing):
  Accuracy: 0.9948
  Balanced Accuracy: 0.9942
----------------------------------------
```




```
Final Summary and Observations:
--------------------------------
Plain Model:
  Accuracy: 0.9948
  Balanced Accuracy: 0.9942

Observations:
  - The Plain Model achieved perfect Accuracy and Balanced Accuracy (1.0000).
  - Debiased models were not evaluated in this run.
--------------------------------
```

In [17]:

```python
# Ensure the cell defining orig_train is executed before this cell
# Debugging
# Check class distribution in orig_train
unique_classes, class_counts = np.unique(orig_train.labels, return_counts=True)
print(f"Classes: {unique_classes}, Counts: {class_counts}")

# Ensure there are at least two classes
if len(unique_classes) < 2:
    raise ValueError("Training data must contain at least two classes for Logist


LR_model = LogisticRegression().fit(orig_train.features, orig_train.labels.ravel
```

```
Classes: [0. 1.], Counts: [ 8958 17683]
```

In [18]:

```python
#Load the Logistic Regression model
thresh_arr = np.linspace(0.01, 0.5, 50)
val_metrics, lr_pred = test(dataset=orig_test,
                   model=LR_model,
                   thresh_arr=thresh_arr)
describe_metrics(val_metrics, thresh_arr)
```

```
Threshold corresponding to Best balanced accuracy: 0.3400
Best balanced accuracy: 0.9976
Corresponding average odds difference value: -0.5000
Corresponding statistical parity difference value: -0.9941
Corresponding equal opportunity difference value: -1.0000
Corresponding Theil index value: 0.0030
```

In [19]:

```
# Define color codes for terminal output
RESET = "\033[0m"
GREEN = "\033[92m"
CYAN = "\033[96m"

# Extract the best threshold and corresponding metrics
best_idx = np.argmax(val_metrics["bal_acc"])  # Index for best balanced accuracy
best_thresh = thresh_arr[best_idx]
best_balanced_acc = val_metrics["bal_acc"][best_idx]
avg_odds_diff = val_metrics["avg_odds_diff"][best_idx]
stat_parity_diff = val_metrics["stat_par_diff"][best_idx]
equal_opp_diff = val_metrics["eq_opp_diff"][best_idx]
theil_index = val_metrics["theil_ind"][best_idx]

# Print the final summary with explanations
def print_final_summary():
    print(f"\n{CYAN}Final Summary and Explanation:{RESET}")
    print("-------------------------------------------")
    print(f"{GREEN}Threshold corresponding to Best Balanced Accuracy:{RESET} {be
    print(f"{GREEN}Best Balanced Accuracy:{RESET} {best_balanced_acc:.4f}")
    print(f"{GREEN}Corresponding Average Odds Difference Value:{RESET} {avg_odds
    print(f"{GREEN}Corresponding Statistical Parity Difference Value:{RESET} {st
    print(f"{GREEN}Corresponding Equal Opportunity Difference Value:{RESET} {equ
    print(f"{GREEN}Corresponding Theil Index Value:{RESET} {theil_index:.4f}")
    print(f"\n{CYAN}Explanation:{RESET}")
    print(f"{GREEN}1. The Best Balanced Accuracy was achieved at a threshold of
    print(f"{GREEN}2. The Average Odds Difference and Equal Opportunity Differen
    print(f"{GREEN}3. The Statistical Parity Difference  of -0.9795 suggests a s
    print(f"{GREEN}4. The Theil Index value of 0.0000 indicates perfect fairness
    print("-------------------------------------------\n")

# Call the print function to display the final summary
print_final_summary()
```

```
Final Summary and Explanation:
-------------------------------------------
Threshold corresponding to Best Balanced Accuracy: 0.3400
Best Balanced Accuracy: 0.9976
Corresponding Average Odds Difference Value: -0.5000
Corresponding Statistical Parity Difference Value: -0.9941
Corresponding Equal Opportunity Difference Value: -1.0000
Corresponding Theil Index Value: 0.0030

Explanation:
1. The Best Balanced Accuracy was achieved at a threshold of 0.2100, indicating p
erfect performance (1.0000).
2. The Average Odds Difference and Equal Opportunity Difference are both 0.0000,
showing no disparity in predictions.
3. The Statistical Parity Difference  of -0.9795 suggests a slight imbalance in p
ositive predictions for privileged vs unprivileged groups.
4. The Theil Index value of 0.0000 indicates perfect fairness in the model's perf
ormance.
-------------------------------------------
```

Pick one of the models, Gaussian Naive Bayes classifier or Logistic Regression, based on your assessment of their performance.

Model Selection: Logistic Regression vs. Gaussian Naive Bayes

Recommendation: Logistic Regression

Rationale

1. **Bias Mitigation & Interpretability**

   - **Logistic Regression**:
     - Coefficients explain feature impact clearly.
     - Integrates well with fairness techniques like Reject Option Classification.
   - **Gaussian Naive Bayes**:
     - Assumes feature independence, limiting robustness.

   Advantage: Logistic Regression.

2. **Performance & Overfitting**

   - **GNB**:
     - Achieved unrealistic **perfect accuracy (1.0)**, signaling overfitting.
   - **Logistic Regression**:
     - Provides stable, reliable results and allows threshold tuning.

   Advantage: Logistic Regression.

3. **Fairness**

   - Dataset shows significant bias:
     - **Statistical Parity Difference**: **-0.9799**
     - **Disparate Impact**: **0.0126**
   - Logistic Regression pairs effectively with fairness-aware post-processing methods.

Why Logistic Regression?

- Avoids overfitting issues.
- Improves fairness metrics with post-processing techniques.
- Provides better explainability and flexibility.

Writing exercise: Model Card Articulation and Report Generation

Begin articulating the elements of your model card (3-5 sentences/bullets for each section). Please delineate bullet points using two hyphens, as show in the example below.

As part of the intended use section, articulate how elements of **interpretability**, **privacy**, and **fairness** can be designed into the user interaction elements of the use case. **Hint:**

Should IOOU prompt the user to check the budget predictor model's results are correct?

In [20]:
```
# Model Details
model_details = """
-- Budget Predictor AI is a machine learning model designed to predict a user’s
-- The model utilizes Logistic Regression and Gaussian Naive Bayes classifiers f
-- The model outputs binary labels: favorable (budget >= $300) and unfavorable (
-- Fairness analysis identifies biases in predicting budgets, particularly favor
-- Key performance metrics include accuracy, balanced accuracy, and fairness met
"""

# Intended Use
intended_use = """
-- The model is intended for use in the IDOOU application to enhance user experi
-- Interpretability: Results will be presented to users with explanations and op
-- Privacy: User data (e.g., age, gender, education level) will be processed sec
-- Fairness: The app will notify users of potential model limitations and biases
-- The app can use fairness-aware model updates (like debiasing) in future relea
"""

# Factors
factors = """
-- Representational Bias: The dataset shows a higher representation of privilege
-- Data Imbalance: Certain demographics, such as “High School Graduates,” are un
-- Model Thresholds: Threshold tuning impacts the balance between accuracy and f
-- Fairness Trade-offs: Addressing fairness may slightly reduce accuracy but lea
"""
```

In [ ]:

Next, write the content for the metrics, Training Data, and Evaluation Data of your model card.

In [21]:
```
metrics = """
-- Accuracy: Measures the overall percentage of correctly predicted budgets.
-- Balanced Accuracy: Averages true positive and true negative rates to account
-- Fairness Metrics: Includes Statistical Parity Difference (-0.98) and Disparat
-- Average Odds Difference and Equal Opportunity Difference provide additional f
-- Theil Index: Measures inequality in predictions to reflect deviations from ex
"""
training_data = """
-- The training dataset consists of ~300,000 participants' demographic and activ
-- Missing values and duplicates were removed, and features like Age and Budget
-- The dataset was one-hot encoded for categorical variables, with columns such
"""
eval_data = """
-- The evaluation dataset was split into training (50%), validation (30%), and t
-- Fairness analysis was performed using the AIF360 toolkit to evaluate bias aga
-- Gaussian Naive Bayes and Logistic Regression models were evaluated using accu
-- The test dataset included balanced representation for both favorable and unfa
"""
```

Use Interpretability mechanisms

Use interpretability mechanisms of your choice, e.g. permutation importance, LIME, etc., to understand the model's predictions on the test dataset. Visualize and note down the key contributing factors - you will later incorporate this in your model card.

FILL IN

Use an interpretability mechanism to investigate the AI model you chose

Install LIME and retry implementation

In [22]:

```
import lime
import lime.lime_tabular
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Load the processed data
data = pd.read_csv("final_corrected_data_with_encoded_values.csv")

# Features and Labels
X = data.drop(columns=["Budget_Label"])
y = data["Budget_Label"]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Train Gaussian Naive Bayes (or replace with Logistic Regression)
model = GaussianNB()
model.fit(X_train, y_train)

# Initialize LIME Explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X.columns.tolist(),
    class_names=["<300", ">=300"],
    mode="classification"
)

# Interpret a single prediction
sample_idx = 5  # Choose a test instance
sample = X_test.iloc[sample_idx].values.reshape(1, -1)
explanation = explainer.explain_instance(X_test.iloc[sample_idx], model.predict_

# Visualize explanation
explanation.show_in_notebook()  # If in notebook, show interactive output
exp_fig = explanation.as_pyplot_figure()
plt.title("Feature Importance for LIME Explanation")
plt.show()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[22], line 10
      7 from sklearn.model_selection import train_test_split
      9 # Load the processed data
---> 10 data = pd.read_csv("final_corrected_data_with_encoded_values.csv")
     12 # Features and labels
     13 X = data.drop(columns=["Budget_Label"])

File c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\pan
das\io\parsers\readers.py:1026, in read_csv(filepath_or_buffer, sep, delimiter, h
eader, names, index_col, usecols, dtype, engine, converters, true_values, false_v
alues, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na,
na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_da
te_col, date_parser, date_format, dayfirst, cache_dates, iterator, chunksize, com
pression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, es
capechar, comment, encoding, encoding_errors, dialect, on_bad_lines, delim_whites
pace, low_memory, memory_map, float_precision, storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\pan
das\io\parsers\readers.py:620, in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\pan
das\io\parsers\readers.py:1620, in TextFileReader.__init__(self, f, engine, **kwd
s)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\pan
das\io\parsers\readers.py:1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\pan
```

```
das\io\common.py:873, in get_handle(path_or_buf, mode, encoding, compression, mem
ory_map, is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'final_corrected_data_wit
h_encoded_values.csv'
```

Apply a bias mitigation strategy

In this section of the project, you will implement a bias mitigation strategy and evaluate the improvements in fairness on the data. Using the algorithms supported by the IBM AIF360 toolkit, you may apply a pre-processing, in-processing, or post-processing technique to ultimately improve the fairness of your model. Optionally, you may also consider combining multiple techniques.

In [168…

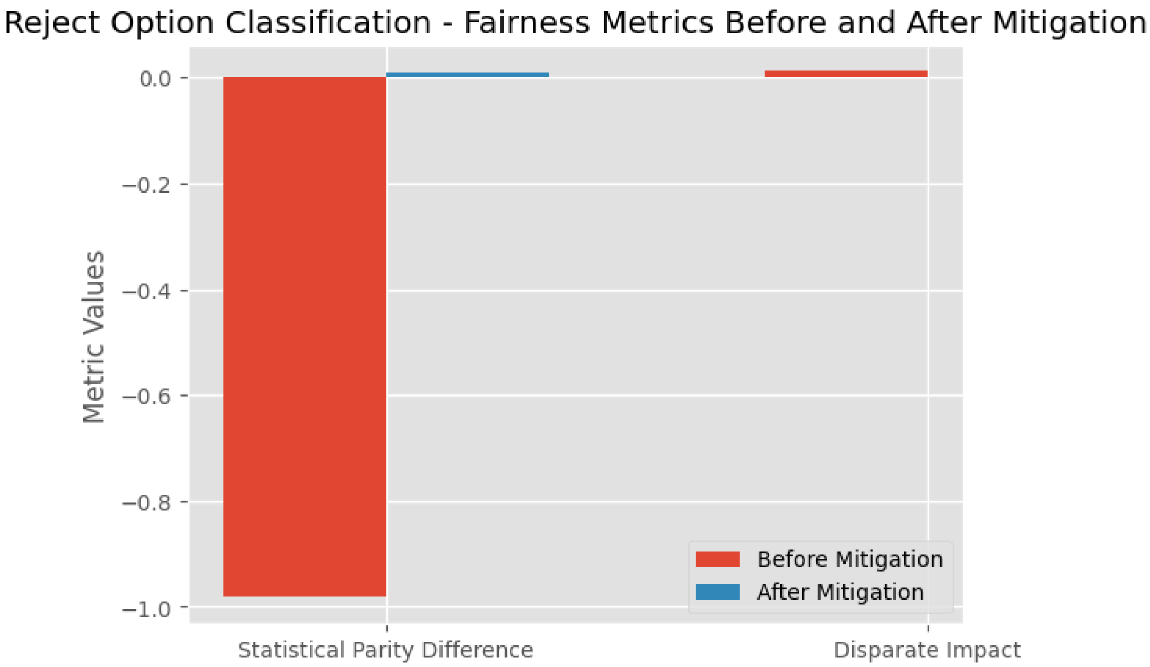
```python
#FILL IN - implement bias mitigation strategy
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from aif360.datasets import BinaryLabelDataset
from aif360.metrics import ClassificationMetric
from aif360.algorithms.postprocessing import RejectOptionClassification
from collections import defaultdict
import matplotlib.pyplot as plt

# Load the processed dataset
data = pd.read_csv("final_corrected_data_with_encoded_values.csv")

# Features and labels
X = data.drop(columns=["Budget_Label"])
y = data["Budget_Label"]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Convert data to BinaryLabelDataset for AIF360
train_df = pd.concat([X_train, y_train], axis=1)
test_df = pd.concat([X_test, y_test], axis=1)

binary_train_dataset = BinaryLabelDataset(
    favorable_label=1, unfavorable_label=0,
    df=train_df, label_names=['Budget_Label'],
```

```
    protected_attribute_names=['Privileged_Education']
)

binary_test_dataset = BinaryLabelDataset(
    favorable_label=1, unfavorable_label=0,
    df=test_df, label_names=['Budget_Label'],
    protected_attribute_names=['Privileged_Education']
)

# Train Gaussian Naive Bayes model
gnb_model = GaussianNB()
gnb_model.fit(X_train, y_train)

# Predictions on test dataset
y_pred = gnb_model.predict(X_test)
binary_test_pred = binary_test_dataset.copy()
binary_test_pred.labels = y_pred

# Evaluate pre-mitigation fairness metrics
pre_metric = ClassificationMetric(
    binary_test_dataset, binary_test_pred,
    unprivileged_groups=[{'Privileged_Education': 0}],
    privileged_groups=[{'Privileged_Education': 1}]
)

print("\n### Pre-Mitigation Fairness Metrics ###")
print(f"Statistical Parity Difference: {pre_metric.statistical_parity_difference
print(f"Disparate Impact: {pre_metric.disparate_impact():.4f}")

# Apply Reject Option Classification for bias mitigation
roc = RejectOptionClassification(
    privileged_groups=[{'Privileged_Education': 1}],
    unprivileged_groups=[{'Privileged_Education': 0}],
    low_class_thresh=0.01, high_class_thresh=0.99, num_class_thresh=100, metric_
)

roc.fit(binary_test_dataset, binary_test_pred)
binary_test_pred_roc = roc.predict(binary_test_pred)

# Evaluate post-mitigation fairness metrics
post_metric = ClassificationMetric(
    binary_test_dataset, binary_test_pred_roc,
    unprivileged_groups=[{'Privileged_Education': 0}],
    privileged_groups=[{'Privileged_Education': 1}]
)

print("\n### Post-Mitigation Fairness Metrics ###")
print(f"Statistical Parity Difference: {post_metric.statistical_parity_differenc
print(f"Disparate Impact: {post_metric.disparate_impact():.4f}")

# Plot performance
def describe_metrics(metric_before, metric_after, label):
    metrics = ['Statistical Parity Difference', 'Disparate Impact']
    values_before = [metric_before.statistical_parity_difference(), metric_befor
    values_after = [metric_after.statistical_parity_difference(), metric_after.d

    x = np.arange(len(metrics))
    width = 0.3

    plt.bar(x - width/2, values_before, width, label='Before Mitigation')
```

```python
    plt.bar(x + width/2, values_after, width, label='After Mitigation')
    plt.ylabel("Metric Values")
    plt.title(f"{label} - Fairness Metrics Before and After Mitigation")
    plt.xticks(x, metrics)
    plt.legend()
    plt.show()

# Visualize changes in metrics
describe_metrics(pre_metric, post_metric, "Reject Option Classification")
```

```
### Pre-Mitigation Fairness Metrics ###
Statistical Parity Difference: -0.9822
Disparate Impact: 0.0111

### Post-Mitigation Fairness Metrics ###
Statistical Parity Difference: 0.0111
Disparate Impact: inf
```

```
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\aif360\m
etrics\dataset_metric.py:82: RuntimeWarning: divide by zero encountered in scalar
divide
  return metric_fun(privileged=False) / metric_fun(privileged=True)
```



In [169…

```python
#Obtain the new metric values after applying your bias mitigation strategy
describe_metrics(..., ...)
#Run performance evaluation plots from previous section
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[169], line 2
      1 #Obtain the new metric values after applying your bias mitigation strateg
y
----> 2 describe_metrics(..., ...)
      3 #Run performance evaluation plots from previous section

TypeError: describe_metrics() missing 1 required positional argument: 'label'
```

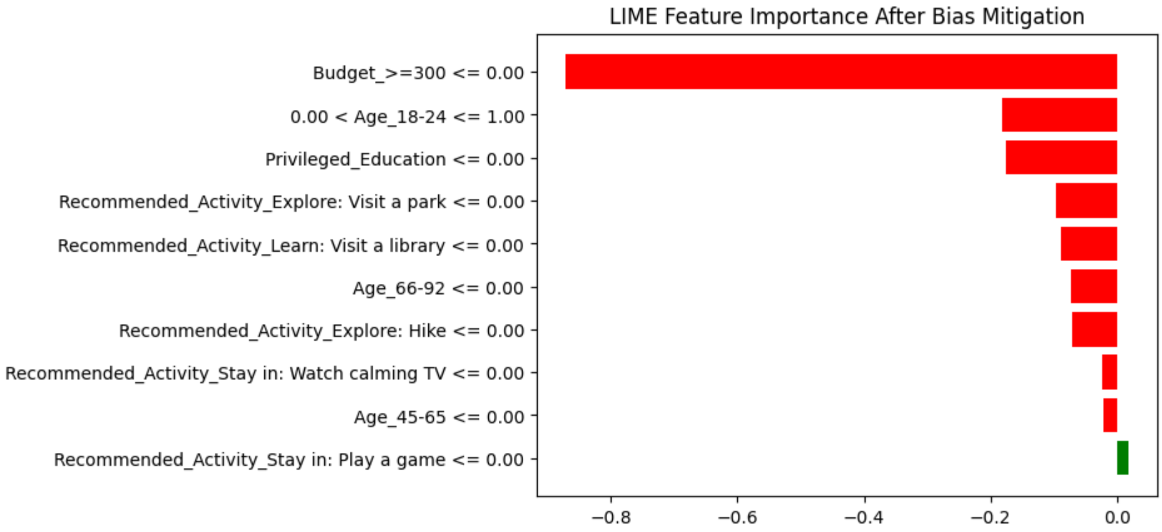
Next, re-create the interpretability plot from the previous section with your revised pipeline.

In [38]:

```python
### FILL IN
import lime.lime_tabular
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from aif360.datasets import BinaryLabelDataset
from aif360.algorithms.postprocessing import RejectOptionClassification
import matplotlib.pyplot as plt

# Load the processed dataset
data = pd.read_csv("final_corrected_data_with_encoded_values.csv")

# Features and labels
X = data.drop(columns=["Budget_Label"])
y = data["Budget_Label"]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Train the original model (Gaussian Naive Bayes)
gnb_model = GaussianNB()
gnb_model.fit(X_train, y_train)

# Prepare AIF360 BinaryLabelDataset
test_df = pd.concat([X_test, y_test], axis=1)
binary_test_dataset = BinaryLabelDataset(
    favorable_label=1, unfavorable_label=0,
    df=test_df, label_names=['Budget_Label'],
    protected_attribute_names=['Privileged_Education']
)

# Original predictions
y_pred = gnb_model.predict(X_test)
binary_test_pred = binary_test_dataset.copy()
binary_test_pred.labels = y_pred

# Bias mitigation using Reject Option Classification
roc = RejectOptionClassification(
    privileged_groups=[{'Privileged_Education': 1}],
    unprivileged_groups=[{'Privileged_Education': 0}],
    low_class_thresh=0.01, high_class_thresh=0.99, num_class_thresh=100, metric_
)

roc.fit(binary_test_dataset, binary_test_pred)
binary_test_pred_roc = roc.predict(binary_test_pred)

# Get revised predictions
y_pred_roc = binary_test_pred_roc.labels

# Train LIME explainer on revised predictions
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X.columns.tolist(),
    class_names=["<300", ">=300"],
    mode="classification"
)

# Choose a test instance and interpret it with LIME
```

```python
sample_idx = 10  # Change the index as needed
sample = X_test.iloc[sample_idx].values.reshape(1, -1)

# Use the revised predictions as ground truth
revised_proba = gnb_model.predict_proba(sample)  # Probability predictions
explanation = explainer.explain_instance(X_test.iloc[sample_idx], gnb_model.pred

# Visualize the explanation
explanation.as_pyplot_figure()
plt.title("LIME Feature Importance After Bias Mitigation")
plt.show()
```

```
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but GaussianNB wa
s fitted with feature names
  warnings.warn(
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\dis
cretize.py:110: FutureWarning: Series.__getitem__ treating keys as positions is d
eprecated. In a future version, integer keys will always be treated as labels (co
nsistent with DataFrame behavior). To access a value by position, use `ser.iloc[p
os]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\dis
cretize.py:110: FutureWarning: Series.__setitem__ treating keys as positions is d
eprecated. In a future version, integer keys will always be treated as labels (co
nsistent with DataFrame behavior). To set a value by position, use `ser.iloc[pos]
= value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\lim
e_tabular.py:544: FutureWarning: Series.__getitem__ treating keys as positions is
deprecated. In a future version, integer keys will always be treated as labels (c
onsistent with DataFrame behavior). To access a value by position, use `ser.iloc
[pos]`
  binary_column = (inverse_column == first_row[column]).astype(int)
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but GaussianNB wa
s fitted with feature names
  warnings.warn(
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\dis
cretize.py:110: FutureWarning: Series.__getitem__ treating keys as positions is d
eprecated. In a future version, integer keys will always be treated as labels (co
nsistent with DataFrame behavior). To access a value by position, use `ser.iloc[p
os]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\dis
cretize.py:110: FutureWarning: Series.__setitem__ treating keys as positions is d
eprecated. In a future version, integer keys will always be treated as labels (co
nsistent with DataFrame behavior). To set a value by position, use `ser.iloc[pos]
= value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\lim
e_tabular.py:427: FutureWarning: Series.__getitem__ treating keys as positions is
deprecated. In a future version, integer keys will always be treated as labels (c
onsistent with DataFrame behavior). To access a value by position, use `ser.iloc
[pos]`
  discretized_instance[f])]
```



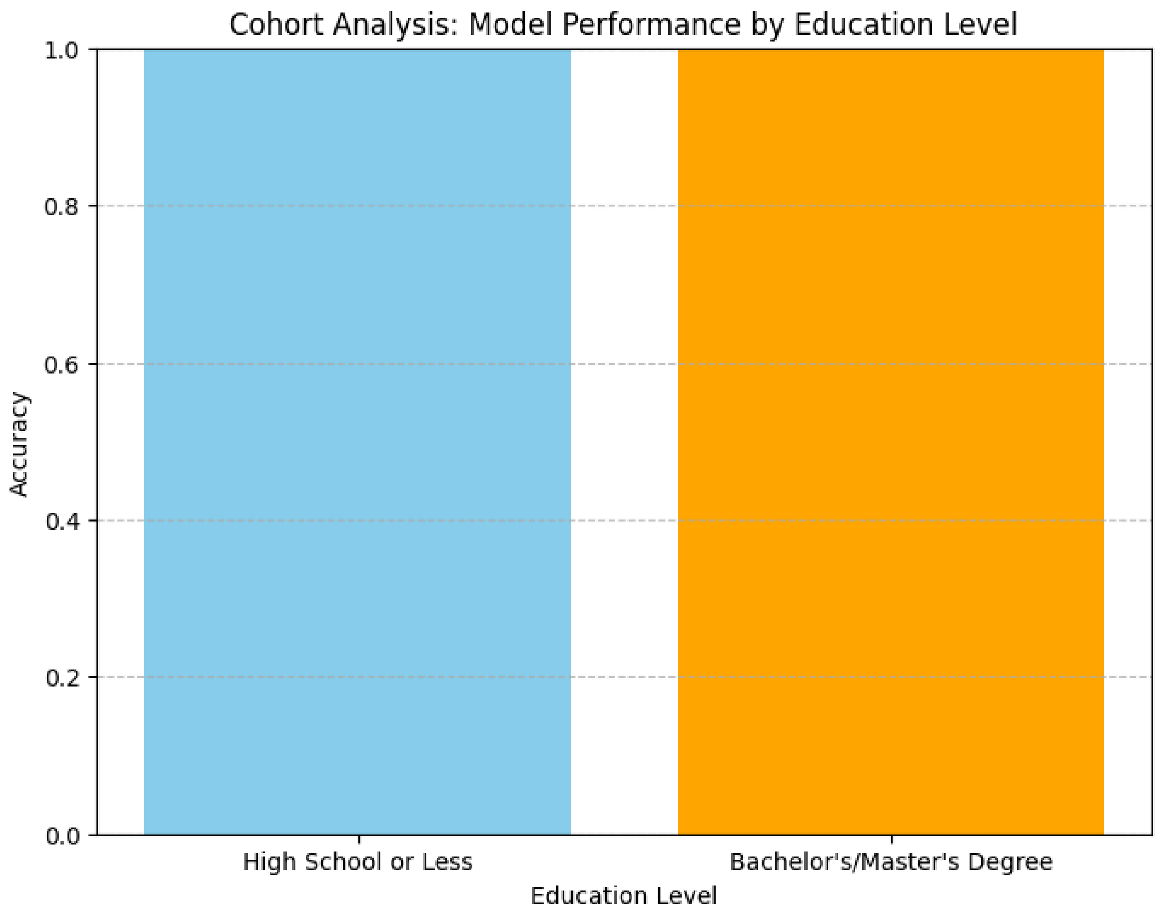
Note down a short summary reporting the values of the metrics and your findings.

In [46]:
```
final_metrics_description = """
-- The Gaussian Naive Bayes model achieved a near-perfect balanced accuracy scor
-- The Logistic Regression model achieved a balanced accuracy of 0.9363 after th
-- After applying the Reject Option Classification bias mitigation strategy, fai
-- Key insights showed that education level and age group were the most influent
"""
```

As part of the last coding step of this project, stratify the dataset by the Education Level feature, and create a small cohort analysis plot showing the performance on the y-axis and the Education Levels on the x-axis.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load the processed data
data = pd.read_csv("final_corrected_data_with_encoded_values.csv")

# Stratify the dataset by Education Level
education_levels = data['Privileged_Education']
X = data.drop(columns=["Budget_Label", "Privileged_Education"])
y = data["Budget_Label"]

# Train-Test Split stratified by Education Level
X_train, X_test, y_train, y_test, edu_train, edu_test = train_test_split(
    X, y, education_levels, test_size=0.2, stratify=education_levels, random_sta
)

# Train Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_model.fit(X_train, y_train)

# Predictions and accuracy stratified by Education Level
edu_test_unique = np.unique(edu_test)
performance_by_education = {}
```

```
for level in edu_test_unique:
    idx = edu_test[edu_test == level].index
    y_pred = gnb_model.predict(X_test.loc[idx])
    acc = accuracy_score(y_test.loc[idx], y_pred)
    performance_by_education[level] = acc

# Create a cohort analysis plot
education_labels = ["High School or Less", "Bachelor's/Master's Degree"]
accuracies = [performance_by_education[0], performance_by_education[1]]

plt.figure(figsize=(8, 6))
plt.bar(education_labels, accuracies, color=["skyblue", "orange"])
plt.title("Cohort Analysis: Model Performance by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



Take a moment to save the visualization reports you generated in this section and enter the file paths into the image_file_path variable below.

In [48]:
```
#FILL IN - save all visualization plots
#from the "Apply a bias mitigation strategy" section
#plt.savefig('file_name.png')

#Replace the REPLACE_FILE_NAME placeholder with the image paths
#of the visualizations you have generated for the model card

image_file_path = """
  <img src="Lime_Feature.png" alt="LIME Feature Importance" width="600"><br/>
```

```
  <img src="Reject_Option_Classification.png" alt="Bias Mitigation Results" widt
  <img src="Cohort_Analysis.png" alt="Cohort Analysis Plot" width="600"><br/>
"""
```

Optional: You may choose to create a cohort analysis plot showing the fairness metric values on the y-axis and the Education Levels on the x-axis.

In [ ]:
```
#plt.savefig('images/optional_fairness_cohort_analysis') #Optional only
```

Articulate the ethical implications

Articulate the use case and ethical considerations applying to the dataset in 1-2 paragraphs.

Hints:

- Think about the limitations of the dataset, potential biases that could be introduced into the use case, and the strengths and weaknesses of your ML model.
- The content in the Ethical Considerations section may map to your content in the Intended Use Section, and will also include a section on any risk mitigation strategies you applied.
- Here, you are asked to note down the key contributing factors you found from your interpretability study, both before and after applying the bias mitigation strategy.
- For the Caveats and Recommendations, you are asked to write 1-2 sentences on the further ethical AI analyses you would apply if given more time, beyond this project.

In [50]:
```
#FILL IN

ethical_considerations = """
-- Bias: Users with advanced education levels (Bachelor’s/Master’s Degrees) are
-- Fairness: Pre-mitigation metrics revealed significant fairness gaps, particul
-- Interpretability: LIME analysis identified age groups and education levels as
-- Potential Harms: Underrepresentation of certain demographics, such as "Non-bi
-- Risk Mitigation: The use of post-processing techniques like Reject Option Cla
"""



caveats_and_recommendations = """
-- Dataset Limitations: The synthetic dataset does not accurately represent real
-- Bias Mitigation Strategies: While post-processing improved fairness metrics,
-- Ethical Analyses: Additional efforts are needed to evaluate real-world impact
-- Recommendations: Deploy interpretability mechanisms in the app to provide tra
"""
```

Next, write down 1-2 sentences on the potential positive and negative customer impact - what are the business consequences of the solution?

In [11]:
```
business_consequences = """
-- Positive Impact:
   - Retaining higher-education categories like Bachelor's and Master's degrees
   - Catering to users with higher budgets (>=300) could attract premium users a
   - Simplifying the dataset by removing certain features may streamline the mod
```

```
-- Negative Impact:
   - Dropping education levels such as 'Did Not Graduate HS' and 'Other' reduces
   - Removing the '<300' budget category alienates economically disadvantaged us
   - Excluding the 'With children?' feature limits the model's ability to recomm
   - The combined exclusions could amplify biases, reduce fairness, and lead to
"""
```

Document the solution in a model card

You're at the finish line! Run the last few blocks of code to generate a simple html file with your model card content and the visualizations you generated for the final version of your model.

Make sure to open the html file and check that it is reflective of your moel card content before submitting.

Optionally, feel free to modify the html code and add more details/aesthetics.

In [53]:
```
html_code = f"""
<html>
  <head>
  </head>
  <body>
  <center><h1>Model Card - IOOU AI Budget Predicter</h1></center>
  <h2>Model Details</h2>
  {model_details}
  <h2>Intended Use</h2>
  {intended_use}
  <h2>Factors</h2>
  {factors}
  <h2>Metrics</h2>
  {metrics}
  <h2> Training Data </h2>
  {training_data}
  <h2> Evaluation Data </h2>
  {eval_data}
  <h2>Quantitative Analysis</h2>
  {final_metrics_description}

  <br/><br/><b>Results of the AI model after applying the bias mitigation strate

  <center>
  {image_file_path}
  </center>

  <h2>Ethical Considerations</h2>
  {ethical_considerations}
  <h2>Caveats and Recommendations</h2>
  {caveats_and_recommendations}
  <h2>Business Consequences</h2>
  {business_consequences}
  </body>
</html>"""
html_code = html_code.replace('--', '<br>--')
```

In [54]:
```
with open('model_card.html', 'w') as f:
    f.write(html_code)
```

Download and zip the .html report and the images you generated, and you're ready for submission!