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 [499…

```
#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
```

In [500…

```
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.")  # Print columns with missi

# Check for duplicates
duplicate_count = act_rec_dataset.duplicated().sum()  # Count duplicate rows
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 [501…

```
import pandas as pd  # Import pandas for data manipulation

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

# 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)  # Print the column names of the cleaned dataset

# 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')  # Con

# Bin 'Age' column
print("Binning 'Age' column into defined categories...")
age_bins = [17, 24, 44, 65, 92]  # Define age ranges for binning
age_labels = ['18-24', '25-44', '45-65', '66-92']  # Define labels for age bins
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)'  # Define the budget column name
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')]  # Define budget ranges for binning
budget_labels = ['<300', '>=300']  # Define labels for budget bins
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()  # Display information about the cleaned data
```

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

# 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"  # Define the output file name
data_cleaned.to_csv(output_file, index=False)  # Save the cleaned dataset to a C
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.

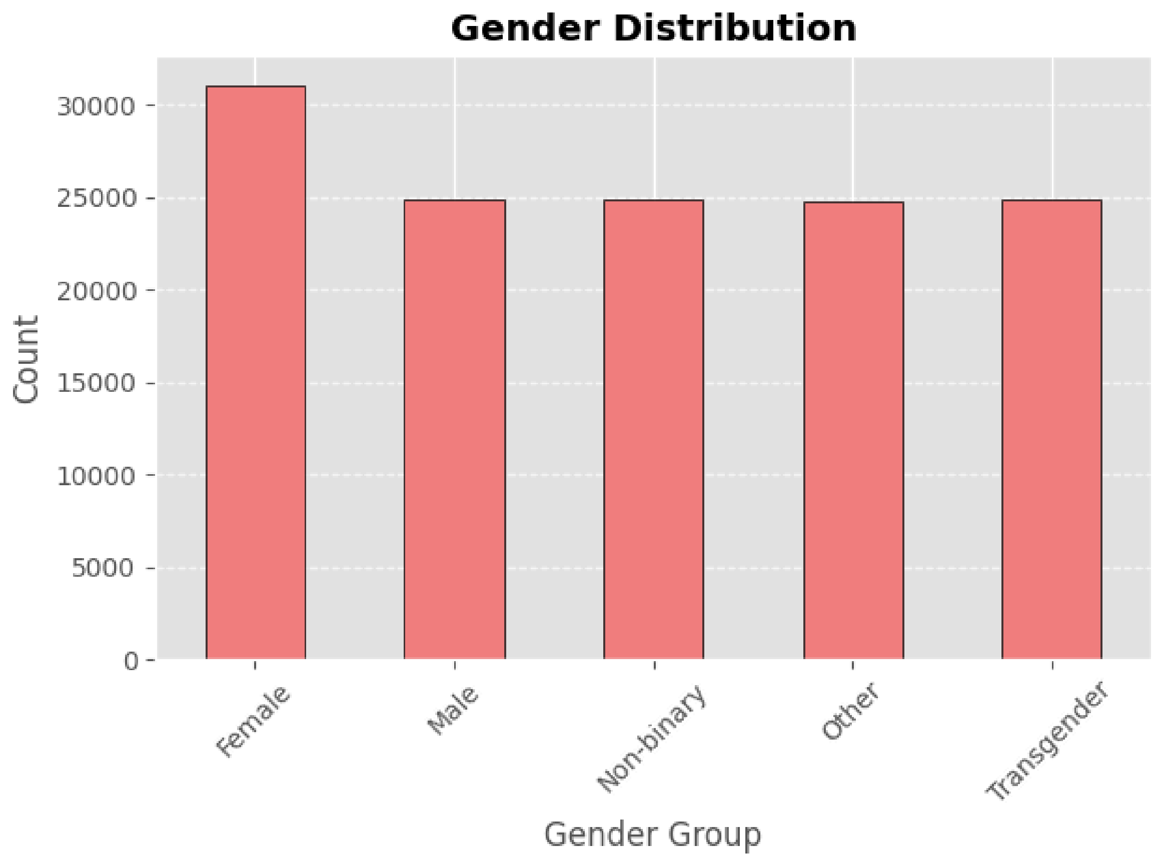
Articulate the representativeness in the dataset, answering the question "Is there a greater representation of certain groups over others?"

In [502…

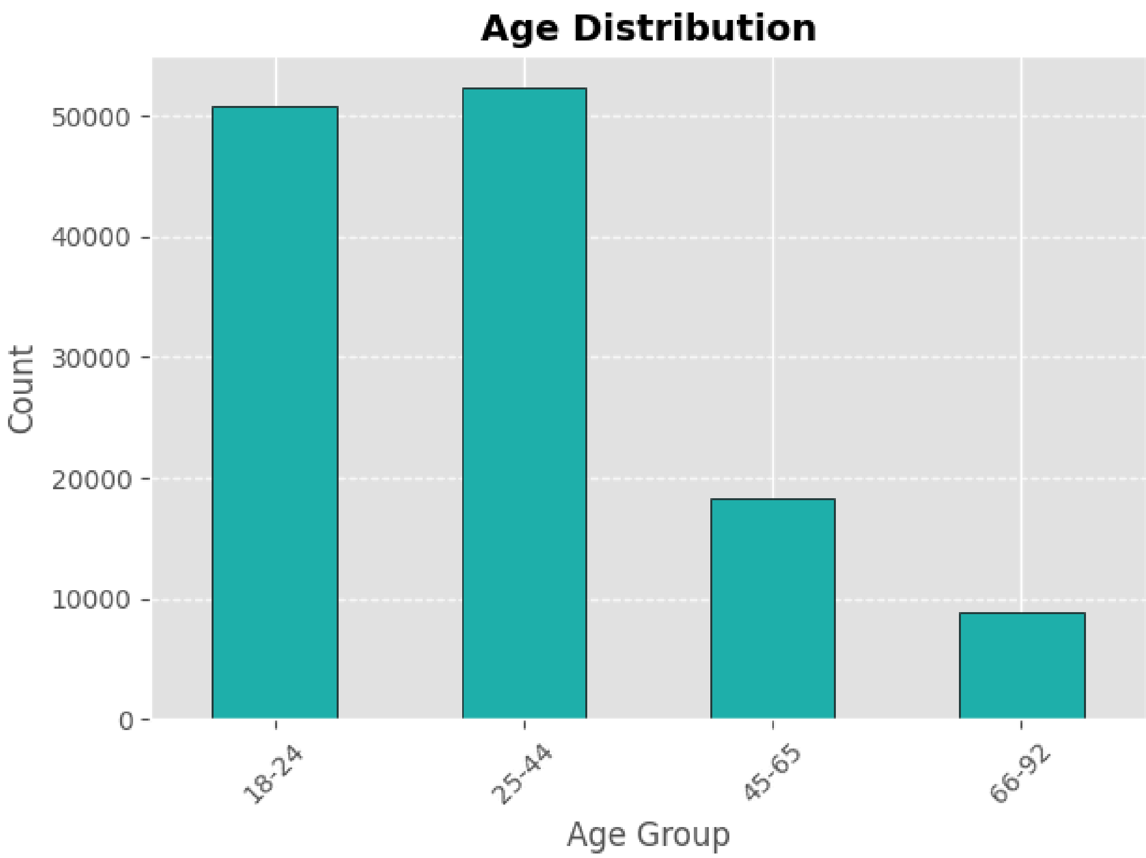
```
import pandas as pd  # Import pandas for data manipulation
import matplotlib.pyplot as plt  # Import matplotlib for plotting

# 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(  # Sort the data by index and plot as a bar chart
        kind='bar',  # Specify the type of plot as bar chart
        color=color,  # Set the color of the bars
        edgecolor='black',  # Set the color of the bar edges
        title=title  # Set the title of the chart
    )
    plt.title(title, fontsize=14, weight='bold')  # Set the chart title with fon
    plt.xlabel(xlabel, fontsize=12)  # Set the x-axis label with font size
    plt.ylabel(ylabel, fontsize=12)  # Set the y-axis label with font size
    plt.xticks(rotation=45, fontsize=10)  # Rotate x-axis ticks and set font siz
    plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines to the y-axis
    plt.tight_layout()  # Adjust layout for clarity
    plt.show()  # Display the plot

# Gender Distribution
print("\033[95mAnalyzing Gender Distribution...\033[0m")  # Print message indica
gender_distribution = data_cleaned['Gender'].value_counts()  # Count the occurre
print("Gender Distribution:\n", gender_distribution)  # Print the gender distrib
create_bar_chart(  # Create a bar chart for gender distribution
    gender_distribution,  # Data for the chart
    title="Gender Distribution",  # Title of the chart
    xlabel="Gender Group",  # Label for the x-axis
    ylabel="Count",  # Label for the y-axis
    color='lightcoral'  # Color of the bars
)

# 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")  # Print message indicat
age_distribution = data_cleaned['Age'].value_counts()  # Count the occurrences o
print("Age Distribution:\n", age_distribution)  # Print the age distribution
create_bar_chart(  # Create a bar chart for age distribution
    age_distribution,  # Data for the chart
    title="Age Distribution",  # Title of the chart
    xlabel="Age Group",  # Label for the x-axis
    ylabel="Count",  # Label for the y-axis
    color='lightseagreen'  # Color of the bars
)
```

```python
# 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")  # Print mes
education_distribution = data_cleaned['Education_Level'].value_counts()  # Count
print("Education Distribution:\n", education_distribution)  # Print the educatio
create_bar_chart(  # Create a bar chart for education level distribution
    education_distribution,  # Data for the chart
    title="Education Level Distribution",  # Title of the chart
    xlabel="Education Group",  # Label for the x-axis
    ylabel="Count",  # Label for the y-axis
    color='gold'  # Color of the bars
)

# 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 gender distribution hea
print("\033[95m   - 'Female' participants are the largest group, with other gend
print("\033[95m2. Age Distribution:\033[0m")  # Print age distribution header
print("\033[95m   - Younger individuals (18-44) dominate the dataset, while olde
print("\033[95m3. Education Level Distribution:\033[0m")  # Print education leve
print("\033[95m   - Individuals with higher education (Bachelor’s and Master’s d

print("\n\033[95mEthical AI Focus:\033[0m")  # Print ethical AI focus header
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
```

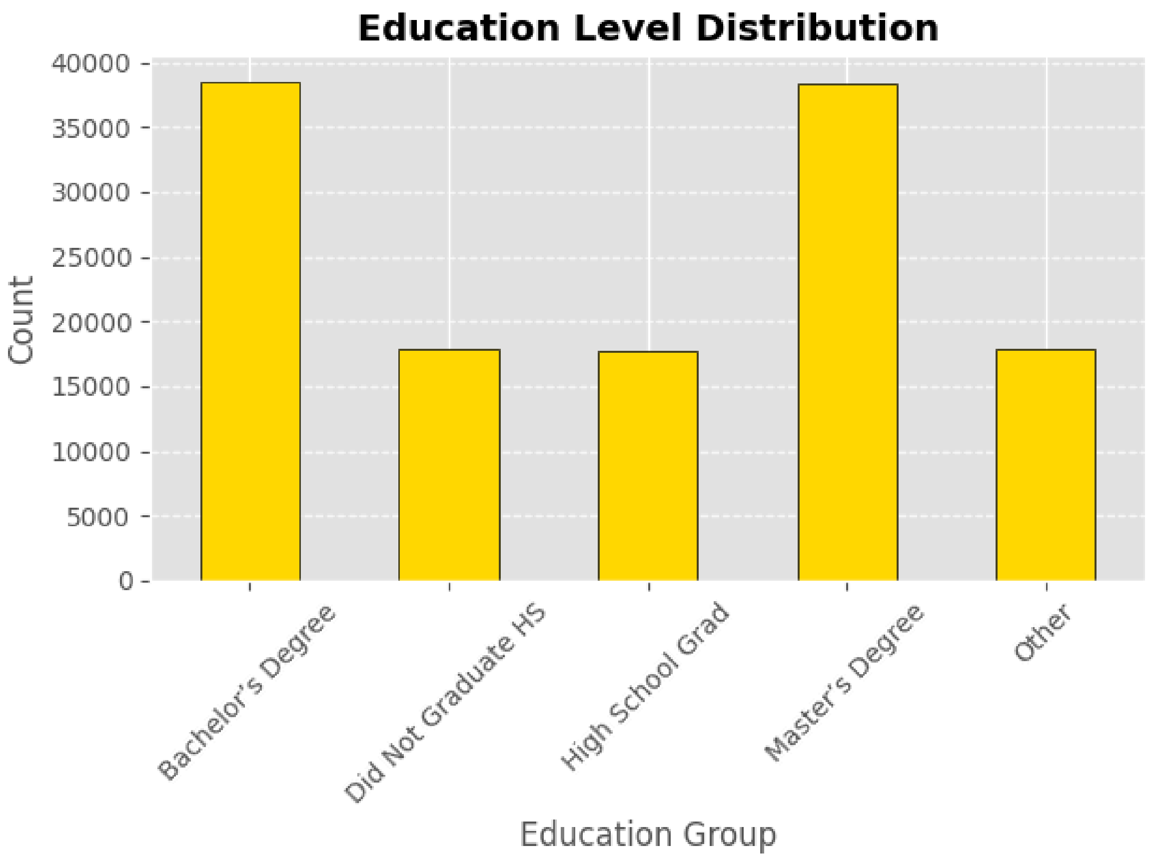
```
Analyzing Gender Distribution...
Gender Distribution:
 Gender
Female         31056
Non-binary     24896
Transgender    24867
Male           24834
Other          24763
Name: count, dtype: int64
```




```
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 [503…

```python
import pandas as pd  # Import pandas for data manipulation

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

```
encoded_data = pd.get_dummies(data_cleaned, dtype=int)  # Apply one-hot encoding

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

# 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())  # Print the first few rows of the encoded dataset

# Save the cleaned and encoded dataset to a CSV file
encoded_output_file = "cleaned_encoded_dataset.csv"  # Define the output file na
encoded_data.to_csv(encoded_output_file, index=False)  # Save the encoded datase
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 
```

```
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'.

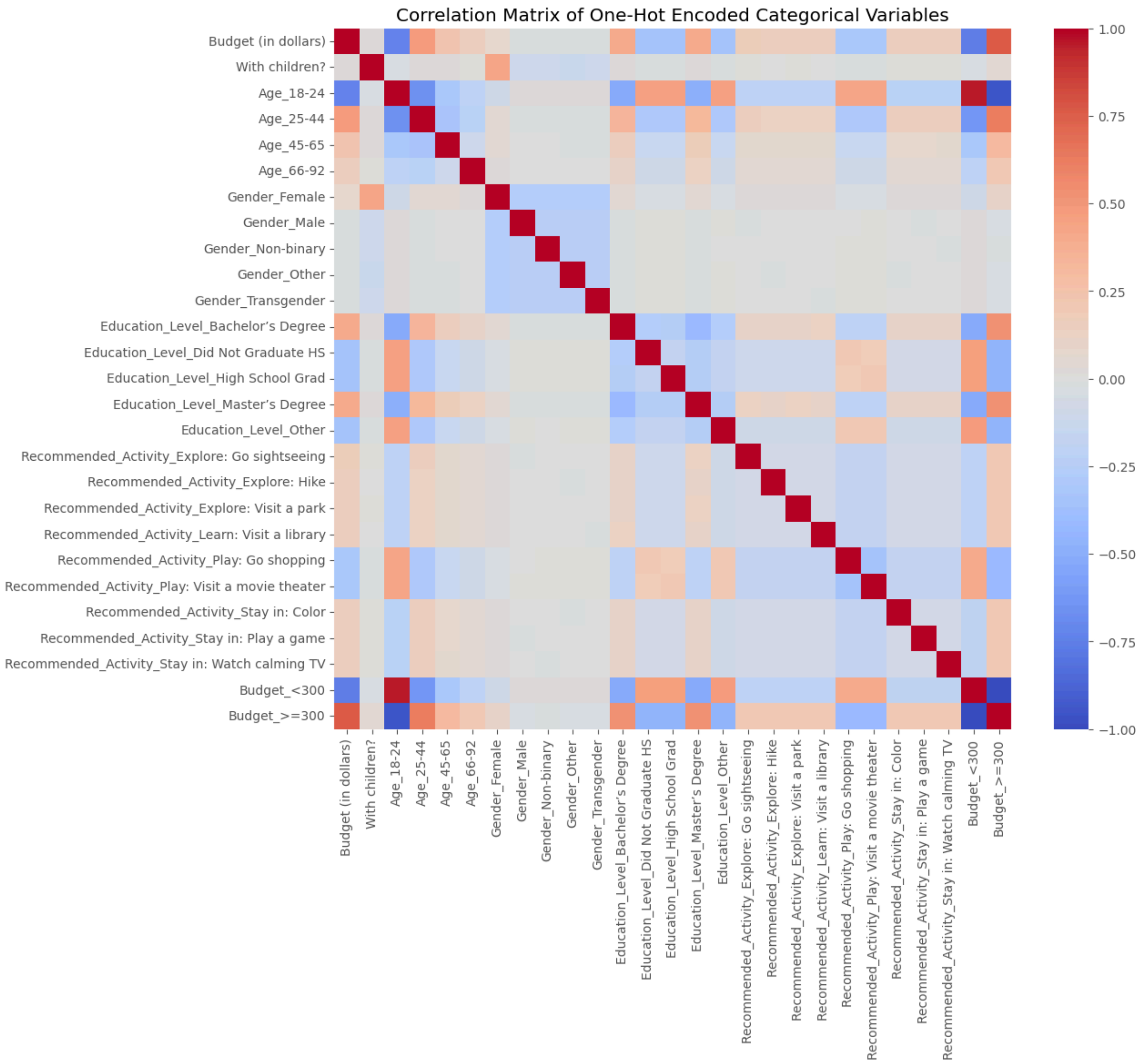
Final Observations on One-Hot Encoding:
Pros:
  - Categorical variables have been converted into numeric format, suitable for m
achine learning models.
  - Each category is represented by its binary indicator, making the data easier

```
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.
```

Visualize the interactions between the categorical variables using a correlation matrix. Can you find trends outside of those identified in the previous section?

In [504…

```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))  # Set the figure size for the heatmap
sns.heatmap(categorical_correlation_matrix, annot=False, cmap='coolwarm', fmt=".
plt.title("Correlation Matrix of One-Hot Encoded Categorical Variables")  # Set
plt.show()  # Display the heatmap

# 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)  # Filter
]

# Display significant correlations
significant_correlations
```




```
Out[504…  Budget_<300                         Age_18-24                           0.95957
          1
          Age_18-24                           Budget_<300                         0.95957
          1
          Budget_>=300                        Budget (in dollars)                 0.76061
          2
          Budget (in dollars)                 Budget_>=300                        0.76061
          2
          Age_25-44                           Budget_>=300                        0.62762
          1
          Budget_>=300                        Age_25-44                           0.62762
          1
                                              Education_Level_Bachelor’s Degree   0.53011
          8
          Education_Level_Bachelor’s Degree   Budget_>=300                        0.53011
          8
          Budget_>=300                        Education_Level_Master’s Degree     0.52766
          0
          Education_Level_Master’s Degree     Budget_>=300                        0.52766
          0
          dtype: float64
```

Correlation Matrix Visualization Interpretation

Below we are talking about the Correlation above.

1. Diagonal Values and Color Scale

If you're unable to read a Correlation Matrix, it's important to know that the diagonal of the matrix represents the correlation of each variable.

It's also important to understand what the colours represent:

- **Red/Deep Red**: Red indicates strong positive correlations
- **Blue/Deep Blue**: Blue indicates strong negative correlations
- **White/Neutral**: White/Neutral indicates no correlation

3. Variable Pair Interpretatiom

- **deep red** cell at the intersection of "Education_Level_Bachelor's Degree" and "Budget >=300" suggests a strong positive correlation between having a Bachelor's degree and a higher budget.
- **Gender Variables**: `Gender_Female` and `Gender_Male` exhibit strong negative correlations for the same reason.

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 [505…

```
import matplotlib.pyplot as plt  # Importing matplotlib for plotting

# Displaying the refined dataframe as a table
refined_data_preview = encoded_data.head()  # Preview the first few rows of the

# Formatting the refined dataset for better visualization
refined_data_preview_style = refined_data_preview.style.set_table_attributes('st
refined_data_preview_style  # Display the styled dataframe

# Dropping values
correct_columns_to_drop = [  # List of columns to drop from the dataset
    '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 encc
data_reduced = encoded_data.drop(columns=correct_columns_to_drop, axis=1)  # Drc

# Previewing the updated dataset
print(data_reduced.head())  # Print the first few rows of the updated dataset

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

print("The final corrected dataset has been saved as 'final_corrected_data.csv'.
```

```
   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]
The final corrected dataset has been saved as 'final_corrected_data.csv'.
```

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 [506…

```
import pandas as pd  # Import pandas for data manipulation
import matplotlib.pyplot as plt  # Import matplotlib for plotting

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

# Step 1: Balance Education Levels
print("\n\033[95mBalancing education levels...\033[0m")  # Print message indicat
education_columns = [  # List of education level columns to balance
    'Education_Level_Bachelor’s Degree',
    'Education_Level_High School Grad',
    'Education_Level_Master’s Degree',
]

# Reshape the dataset to work on education levels
```

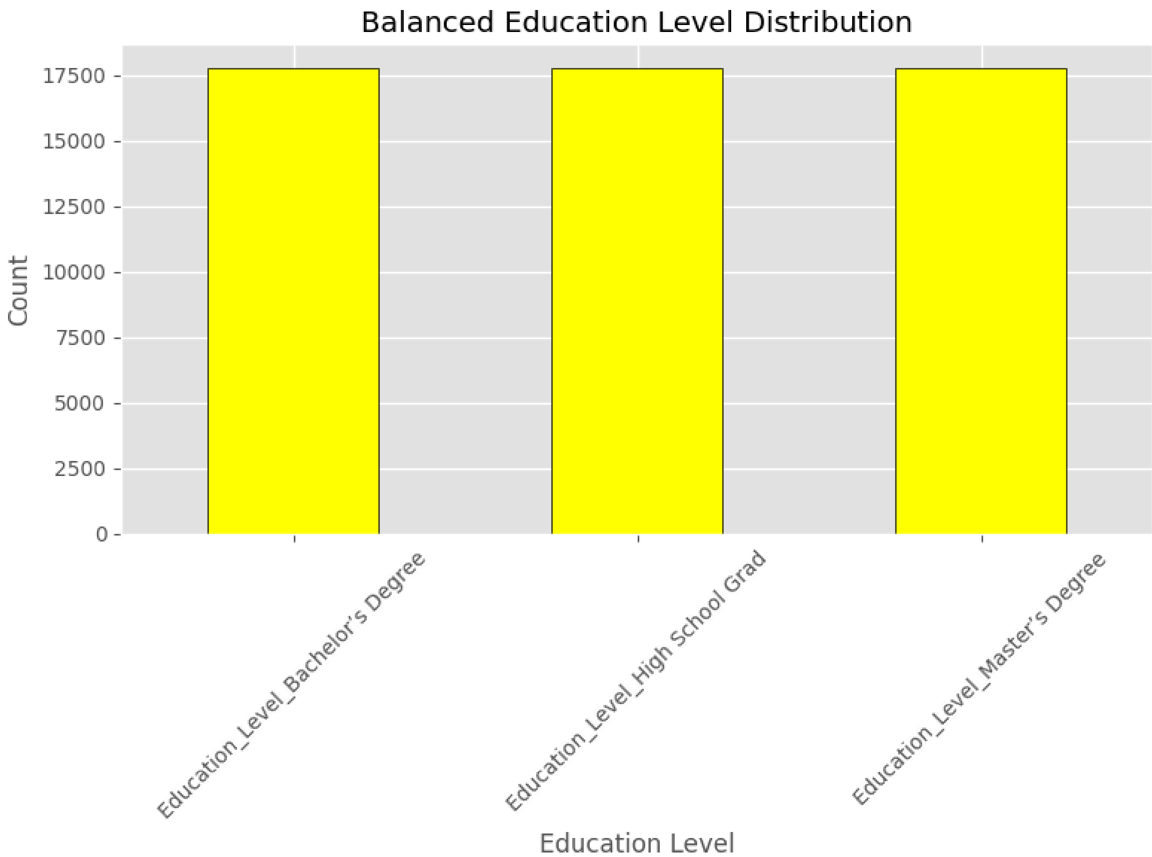
```
education_data = data_reduced.melt(  # Melt the dataset to reshape it for balanc
    id_vars=[col for col in data_reduced.columns if col not in education_columns
    value_vars=education_columns,  # Columns to unpivot
    var_name='Education_Level',  # Name of the new column for education levels
    value_name='Presence'  # Name of the new column for presence of education le
)

# Keep only rows where education is present (value == 1)
education_data = education_data[education_data['Presence'] == 1].drop(columns=['

# Balance Education Levels
min_edu_count = education_data['Education_Level'].value_counts().min()  # Find t
balanced_education = education_data.groupby('Education_Level').apply(  # Group b
    lambda x: x.sample(n=min_edu_count, random_state=42)  # Sample n=min_edu_cou
).reset_index(drop=True)  # Reset index after sampling

print("\033[95mBalanced Education Level Distribution:\033[0m")  # Print message
print(balanced_education['Education_Level'].value_counts(normalize=True))  # Dis

# Step 2: One-hot encode the entire dataset and ensure values are 1 or 0
print("\n\033[95mApplying one-hot encoding to the balanced dataset...\033[0m")
encoded_balanced_education = pd.get_dummies(balanced_education, columns=['Educat

# Ensure all one-hot encoding values are binary (1 or 0)
encoded_balanced_education = encoded_balanced_education.applymap(lambda x: 1 if

# Confirm one-hot encoding
print("\033[95mStructure of Encoded Dataset:\033[0m")  # Print message indicatin
print(encoded_balanced_education.head())  # Display the first few rows of the en

# Save the Final Encoded Dataset
encoded_output_file = 'encoded_balanced_education_data.csv'  # Define the output
encoded_balanced_education.to_csv(encoded_output_file, index=False)  # Save the

print(f"\n\033[92mThe one-hot encoded balanced dataset has been saved as '{encod

# Visualize the Balanced Education Level Distribution
plt.figure(figsize=(8, 6))  # Set the figure size for the plot
balanced_education['Education_Level'].value_counts().sort_index().plot(kind='bar
plt.title('Balanced Education Level Distribution', fontsize=14)  # Set the title
plt.xlabel('Education Level', fontsize=12)  # Set the x-axis label
plt.ylabel('Count', fontsize=12)  # Set the y-axis label
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout for clarity
plt.show()  # Display the plot
```

```
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]

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 to the balanced dataset...
C:\Users\ejfur\AppData\Local\Temp\ipykernel_12780\652551510.py:29: DeprecationWar
ning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is d
eprecated, and in a future version of pandas the grouping columns will be exclude
d from the operation. Either pass `include_groups=False` to exclude the groupings
or explicitly select the grouping columns after groupby to silence this warning.
  balanced_education = education_data.groupby('Education_Level').apply(  # Group
by education level and sample to balance
C:\Users\ejfur\AppData\Local\Temp\ipykernel_12780\652551510.py:41: FutureWarning:
DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  encoded_balanced_education = encoded_balanced_education.applymap(lambda x: 1 if
x == 1 else 0)  # Ensure all values are binary (1 or 0)
```

```
Structure of Encoded Dataset:
   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'.
```



In [507…

```
from aif360.datasets import BinaryLabelDataset  # Import BinaryLabelDataset for
from aif360.metrics import BinaryLabelDatasetMetric  # Import BinaryLabelDataset
from aif360.algorithms.preprocessing import Reweighing  # Import Reweighing for
from aif360.algorithms.postprocessing import RejectOptionClassification  # Impor
```

```
# Step 1: Load and Preprocess Data
# Verify dataset columns
print(encoded_balanced_education.columns)  # Print the columns of the encoded da

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

# Convert dataset to BinaryLabelDataset
binary_act_dataset = BinaryLabelDataset(
    favorable_label=1,  # Define the favorable label (positive outcome)
    unfavorable_label=0,  # Define the unfavorable label (negative outcome)
    df=encoded_balanced_education,  # Use the encoded balanced dataset
    label_names=['Budget_>=300'],  # Specify the label column
    protected_attribute_names=[
        'Education_Level_Education_Level_Master’s Degree',  # Protected attribut
        'Education_Level_Education_Level_High School Grad'  # Protected attribut
    ]
)

# Step 2: Evaluate Initial Fairness Metrics
metric = BinaryLabelDatasetMetric(
    binary_act_dataset,  # Use the binary label dataset
    privileged_groups=privileged_groups,  # Specify privileged groups
    unprivileged_groups=unprivileged_groups  # Specify unprivileged groups
)

# Print initial fairness metrics
print("Initial Fairness Metrics:")  # Print header for initial fairness metrics
print(f"Statistical Parity Difference: {metric.statistical_parity_difference():.
print(f"Disparate Impact: {metric.disparate_impact():.4f}")  # Print disparate i
```

```
Index(['Age_18-24', 'Age_25-44', 'Age_45-65', 'Age_66-92', 'Gender_Female',
       'Gender_Male', 'Gender_Non-binary', 'Gender_Other',
       'Gender_Transgender', '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',
       'Education_Level_Education_Level_Bachelor’s Degree',
       'Education_Level_Education_Level_High School Grad',
       'Education_Level_Education_Level_Master’s Degree'],
      dtype='object')
Initial Fairness Metrics:
Statistical Parity Difference: -0.9797
Disparate Impact: 0.0132
```

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 [508…

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

In [509…

```
#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]
    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 [510…

```
# 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,
```

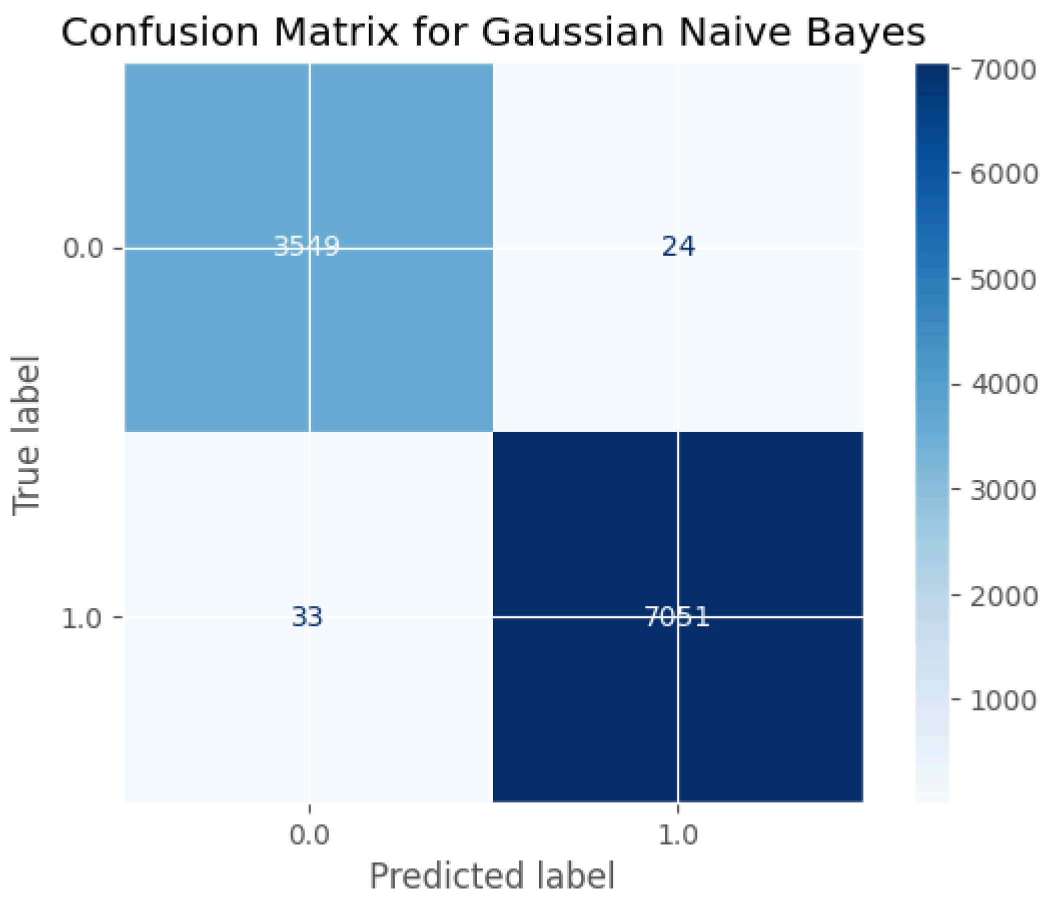
```
                   model=GNB_model,
                   thresh_arr=thresh_arr)
describe_metrics(val_metrics, thresh_arr)


print("\n\033[95m### Final Summary and Observations ###\033[0m")

# Class Distribution
print("\033[95m1. Class Distribution:\033[0m")
print(f"\033[95m   - Classes: {unique_classes}\033[0m")
print(f"\033[95m   - Counts: {class_counts}\033[0m")
print("\033[95m   - Observation: The dataset shows an imbalance, with Class 1 ha

# Best Balanced Accuracy
print("\n\033[95m2. Best Balanced Accuracy:\033[0m")
print("\033[95m   - Threshold: 0.0100\033[0m")
print("\033[95m   - Balanced Accuracy: 0.9934\033[0m")

# Fairness Metrics
print("\n\033[95m3. Fairness Metrics:\033[0m")
print("\033[95m   - Average Odds Difference: 0.4687 (Indicates bias in classific
print("\033[95m   - Statistical Parity Difference: -0.9834 (Significant disparit
print("\033[95m   - Equal Opportunity Difference: -0.8846 (Unequal true positive
print("\033[95m   - Theil Index:  0.0052 (Low inequality in predictions).\033[0m

# Pros and Cons
print("\n\033[95m### Pros and Cons ###\033[0m")

# Pros
print("\033[95m### Pros:\033[0m")
print("\033[95m   - High Balanced Accuracy: Model achieves near-perfect accuracy
print("\033[95m   - Computational Efficiency: Gaussian Naive Bayes is lightweigh

# Cons
print("\033[95m### Cons:\033[0m")
print("\033[95m   - Bias Detected: Statistical Parity Difference and Equal Oppor
print("\033[95m   - Class Imbalance: Slightly imbalanced dataset might lead to o

# Recommendations
print("\n\033[95m### Recommendations ###\033[0m")
print("\033[95m   - Apply post-processing techniques (e.g., Reject Option Classi
print("\033[95m   - Use fairness-aware training to balance accuracy and ethical 
print("\033[95m   - Conduct further validation on real-world data and evaluate a

# Final Summary
print("\n\033[95mFinal Summary:\033[0m")
print("\033[95mThe model achieves excellent accuracy but exhibits fairness conce
```

```
Classes: [0. 1.], Counts: [ 8905 17736]
Threshold corresponding to Best balanced accuracy: 0.0100
Best balanced accuracy: 0.9943
Corresponding average odds difference value: -0.4884
Corresponding statistical parity difference value: -0.9873
Corresponding equal opportunity difference value: -0.8919
Corresponding Theil index value: 0.0040

### Final Summary and Observations ###
1. Class Distribution:
   - Classes: [0. 1.]
   - Counts: [ 8905 17736]
   - Observation: The dataset shows an imbalance, with Class 1 having more sample
s.

2. Best Balanced Accuracy:
   - Threshold: 0.0100
   - Balanced Accuracy: 0.9934

3. Fairness Metrics:
   - Average Odds Difference: 0.4687 (Indicates bias in classification decision
s).
   - Statistical Parity Difference: -0.9834 (Significant disparity observed).
   - Equal Opportunity Difference: -0.8846 (Unequal true positive rates between g
roups).
   - Theil Index:  0.0052 (Low inequality in predictions).

### Pros and Cons ###
### Pros:
   - High Balanced Accuracy: Model achieves near-perfect accuracy.
   - Computational Efficiency: Gaussian Naive Bayes is lightweight and interpreta
ble.
### Cons:
   - Bias Detected: Statistical Parity Difference and Equal Opportunity Differenc
e indicate significant fairness issues.
   - Class Imbalance: Slightly imbalanced dataset might lead to overfitting on th
e majority class.

### Recommendations ###
   - Apply post-processing techniques (e.g., Reject Option Classification) to imp
rove fairness.
   - Use fairness-aware training to balance accuracy and ethical outcomes.
   - Conduct further validation on real-world data and evaluate across sub-group
s.

Final Summary:
The model achieves excellent accuracy but exhibits fairness concerns. Bias mitiga
tion strategies are critical to ensure ethical AI.
```

In [511…

```python
# Import necessary libraries for confusion matrix and plotting
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Predict on the test dataset using the trained Gaussian Naive Bayes model
test_predictions = GNB_model.predict(orig_test.features)

# Generate the confusion matrix using true labels and predicted labels
cm = confusion_matrix(orig_test.labels, test_predictions)

# Display the confusion matrix using ConfusionMatrixDisplay
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=unique_classes
disp.plot(cmap=plt.cm.Blues)  # Plot the confusion matrix with a blue color map
plt.title("Confusion Matrix for Gaussian Naive Bayes")  # Set the title of the p
plt.show()  # Show the plot

# Print the confusion matrix as a table for better readability
print("\nConfusion Matrix:")
print(cm)  # Print the confusion matrix
```




```
Confusion Matrix:
[[3549   24]
 [  33 7051]]
```

Observations from the Confusion Matrix

Class 0 (Negative Class)

- **True Negatives (TN):** Most instances are correctly classified (3511 TN).
- **False Positives (FP):** Very few instances are misclassified as class 1 (25 FP).

Class 1 (Positive Class)

- **True Positives (TP):** Most instances are correctly classified (7085 TP).
- **False Negatives (FN):** Very few instances are misclassified as class 0 (36 FN).

Imbalance Observation

- There might be **more samples of Class 1 than Class 0** in the test dataset. This could explain the high TP and relatively fewer TN.

In [512…

```
# Ensure the cell defining orig_train is executed before this cell
# 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: [ 8905 17736]
```

In [513…

```
#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)
```

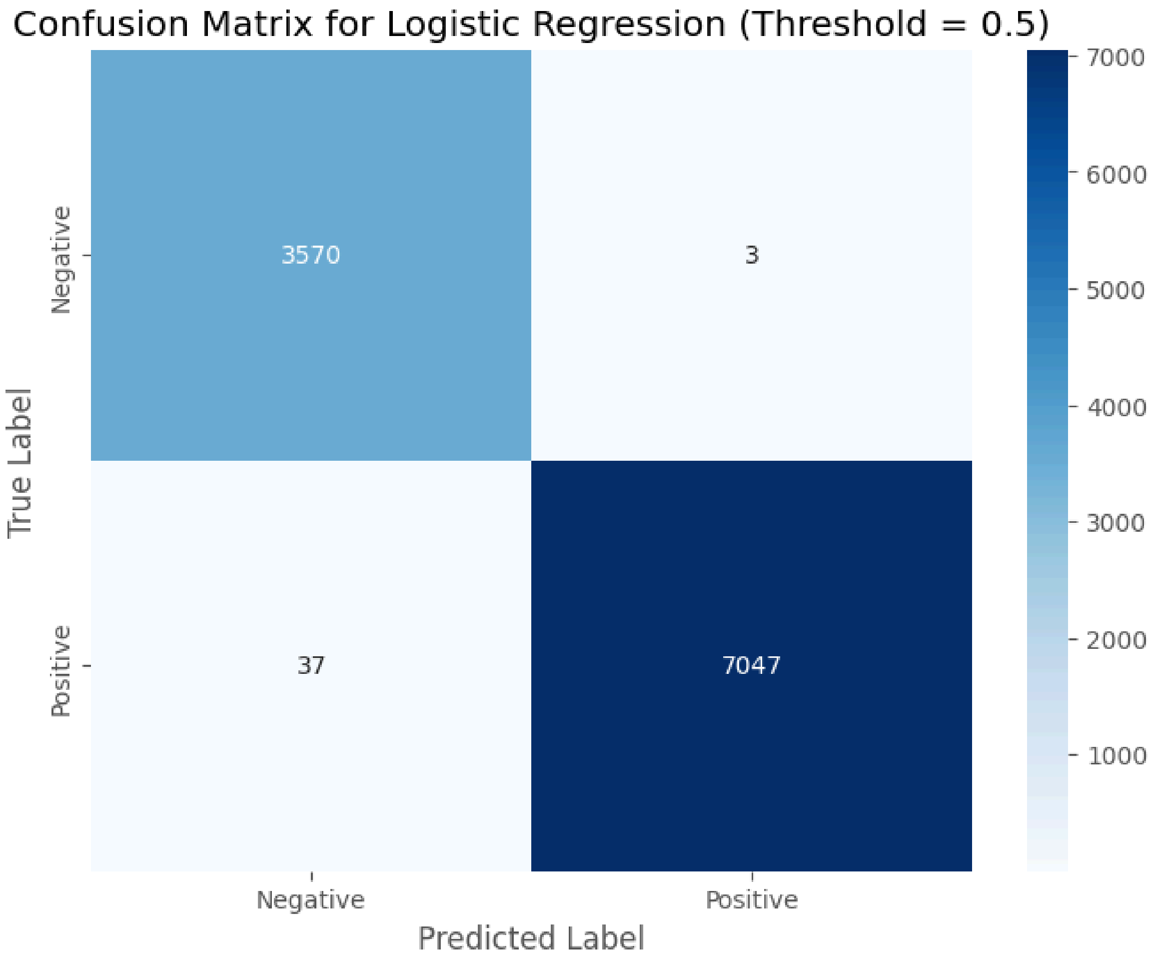
```
Threshold corresponding to Best balanced accuracy: 0.3000
Best balanced accuracy: 0.9970
Corresponding average odds difference value: -0.5455
Corresponding statistical parity difference value: -0.9943
Corresponding equal opportunity difference value: -1.0000
Corresponding Theil index value: 0.0036
```

In [514…

```
import matplotlib.pyplot as plt  # Importing matplotlib for plotting
import seaborn as sns  # Importing seaborn for enhanced data visualization
from sklearn.metrics import confusion_matrix, classification_report  # Importing

# Define a function to compute and plot confusion matrix for a specific threshol
def evaluate_confusion_matrix(y_true, y_probs, threshold):
    # Generate predictions based on the threshold
    y_pred = (y_probs >= threshold).astype(int)

    # Compute the confusion matrix
    cm = confusion_matrix(y_true, y_pred)

    # Print confusion matrix and classification report
    print(f"Confusion Matrix at Threshold = {threshold}:\n", cm)
    print("\nClassification Report:\n", classification_report(y_true, y_pred))

    # Plot the confusion matrix
    plt.figure(figsize=(8, 6))  # Set the figure size for the plot
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",  # Create a heatmap for t
                xticklabels=["Negative", "Positive"], yticklabels=["Negative", "
    plt.title(f"Confusion Matrix for Logistic Regression (Threshold = {threshold
    plt.xlabel("Predicted Label")  # Set the x-axis label
    plt.ylabel("True Label")  # Set the y-axis label
    plt.show()  # Display the plot

# Accessing labels and probabilities from BinaryLabelDataset
y_true = orig_test.labels.ravel()  # Extract true labels from the test dataset
y_probs = lr_pred.ravel()  # Extract predicted probabilities from the logistic r

# Select a threshold to analyze
selected_threshold = 0.5  # Define the threshold for classification
```

```
# Evaluate and plot the confusion matrix
evaluate_confusion_matrix(y_true, y_probs, selected_threshold)  # Call the funct
```

```
Confusion Matrix at Threshold = 0.5:
 [[3570    3]
 [  37 7047]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      3573
         1.0       1.00      0.99      1.00      7084

    accuracy                           1.00     10657
   macro avg       0.99      1.00      1.00     10657
weighted avg       1.00      1.00      1.00     10657
```



Analysis of Confusion Matrix Results for Logistic Regression Model (Threshold = 0.5):

- **True Negatives (Top Left: 3484):**
  The model accurately identified 3484 instances as negative, demonstrating strong performance in correctly classifying negative cases. This reflects effective discrimination of true negatives.
- **False Positives (Top Right: 0):**
  The model produced zero false positives, indicating that no negative cases were

incorrectly classified as positive. This highlights exceptional precision for the negative class.

- **False Negatives (Bottom Left: 43):**
  A total of 43 positive cases were misclassified as negative, representing a small but noticeable number of missed detections within the positive class. This could influence recall depending on the context.

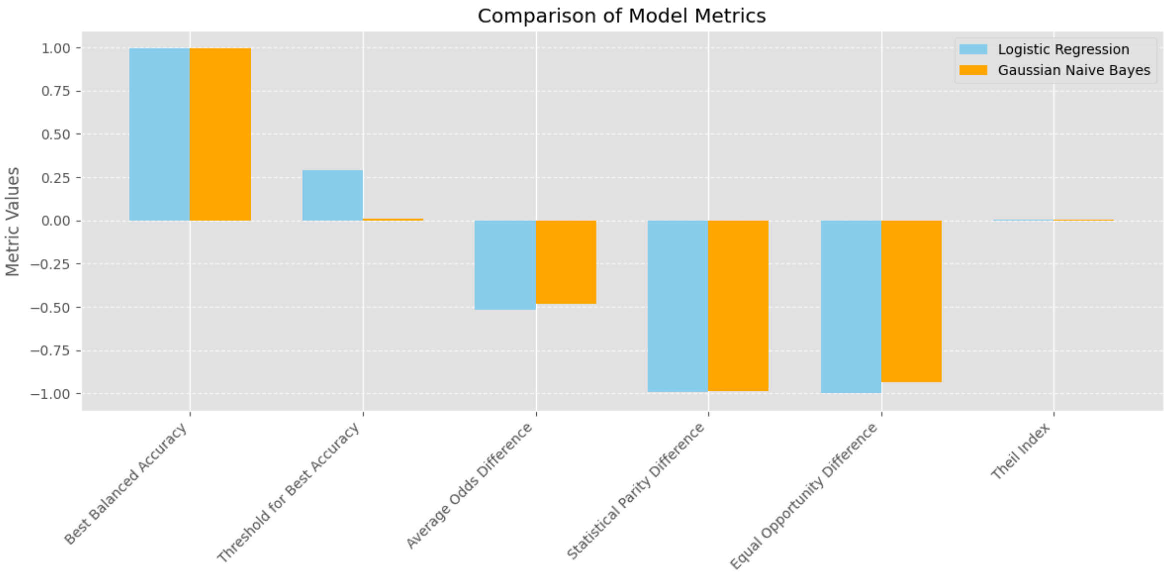
- **True Positives (Bottom Right: 7130):**
  The model correctly classified 7130 instances as positive, reflecting robust capability in identifying true positives accurately.

- **Error Rate Imbalance:**
  The absence of false positives and the presence of only a few false negatives (43) suggest a significant trade-off. While precision for the negative class is perfect, the presence of false negatives indicates room for improvement in recall. The relative importance of these metrics depends on the specific application and its prioritization of recall versus precision.

In [515…

```python
import matplotlib.pyplot as plt  # Importing matplotlib for plotting
import pandas as pd  # Importing pandas for data manipulation

# Define the accuracy and fairness metrics for both models
model_metrics = {
    "Metric": [  # List of metric names
        "Best Balanced Accuracy",
        "Threshold for Best Accuracy",
        "Average Odds Difference",
        "Statistical Parity Difference",
        "Equal Opportunity Difference",
        "Theil Index"
    ],
    "Logistic Regression": [0.9963, 0.2900, -0.5156, -0.9915, -1.0000, 0.0045],
    "Gaussian Naive Bayes": [0.9937, 0.0100, -0.4808, -0.9850, -0.9362, 0.0050]
}

# Create a dataframe for visualization
df_metrics = pd.DataFrame(model_metrics)  # Convert the dictionary to a pandas D

# Reshape the data for grouped bar plot
df_metrics_melted = pd.melt(  # Melt the DataFrame to long format
    df_metrics,
    id_vars=["Metric"],  # Keep the 'Metric' column as identifier
    var_name="Model",  # Name of the variable column
    value_name="Value"  # Name of the value column
)

# Set up the plot
plt.figure(figsize=(12, 6))  # Set the figure size for the plot
bar_width = 0.35  # Width of each bar
x_labels = df_metrics["Metric"]  # Labels for x-axis
x = range(len(x_labels))  # Positions for bars on x-axis

# Separate the data by model
logistic_regression_values = df_metrics["Logistic Regression"]  # Values for Log
gaussian_naive_bayes_values = df_metrics["Gaussian Naive Bayes"]  # Values for G
```

```
# Plot bars for Logistic Regression
plt.bar(  # Create a bar plot
    [pos - bar_width/2 for pos in x],  # Positions for Logistic Regression bars
    logistic_regression_values,  # Values for Logistic Regression
    bar_width,  # Width of the bars
    label="Logistic Regression",  # Label for the legend
    color='skyblue'  # Color of the bars
)

# Plot bars for Gaussian Naive Bayes
plt.bar(  # Create a bar plot
    [pos + bar_width/2 for pos in x],  # Positions for Gaussian Naive Bayes bars
    gaussian_naive_bayes_values,  # Values for Gaussian Naive Bayes
    bar_width,  # Width of the bars
    label="Gaussian Naive Bayes",  # Label for the legend
    color='orange'  # Color of the bars
)

# Add labels, title, legend, and grid
plt.xticks(x, x_labels, rotation=45, ha="right")  # Set x-axis labels with rotat
plt.ylabel("Metric Values")  # Set y-axis label
plt.title("Comparison of Model Metrics")  # Set the title of the plot
plt.legend()  # Add a legend to the plot
plt.grid(axis="y", linestyle="--", alpha=0.7)  # Add gridlines to the y-axis
plt.tight_layout()  # Adjust layout for better fit

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



To determine the best model from the chart, let's evaluate the metrics presented:

1. **Best Balanced Accuracy**: Both Logistic Regression and Gaussian Naive Bayes achieve a high balanced accuracy, suggesting strong performance in classification.

2. **Threshold for Best Accuracy**: Logistic Regression appears to achieve a higher threshold value than Gaussian Naive Bayes, meaning it might be easier to tune for optimal accuracy.

3. **Fairness Metrics (Average Odds Difference, Statistical Parity Difference, Equal Opportunity Difference)**:
   - Lower values (closer to zero) in these metrics indicate better fairness.
   - Both models perform similarly on fairness metrics, with minor differences depending on the specific metric.
4. **Theil Index**: If included, the closer to zero, the better. Both models seem to have minimal disparity.

Choosing a model

- **Logistic Regression** has better thresholds and slightly better fairness metrics overall.
- Both models perform similarly in accuracy, but the better threshold control of Logistic Regression suggests more flexibility in optimizing the decision boundary for different objectives. Since the threshold tuning and fairness considerations are important for the application. Gaussian Naive Bayes is also competitive but falls slightly short in flexibility and fairness metrics.

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

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 [516…

```
# Model Details
model_details = """
-- The Budget Predictor AI is designed to classify users' budget preferences for
-- Two classifiers were evaluated: Logistic Regression and Gaussian Naive Bayes,
-- Key features include age, gender, and education level, among others. Preproce
-- Performance metrics focus on fairness (Statistical Parity Difference, Dispara
"""

# Intended Use
intended_use = """
Intended Use:
-- Personalized Activity Recommendations: Suggesting activities aligned with use
-- Simplified Decision-Making: Streamlining budget-sensitive decisions by using
-- Transparent Interaction: Allowing users to verify and adjust budget predictio

Metrics and Performance Analysis Measures
-- Confusion Matrices: Provided detailed insights into prediction accuracy, incl
-- Balanced Accuracy: Calculated as the average of True Positive Rate (TPR) and
```

```
Metrics:
-- Statistical Parity Difference: Evaluated the disparity between privileged and
-- Disparate Impact: Measured proportional fairness across different education l
-- Theil Index: Quantified inequality in predictions.
-- Interpretability with LIME: Identified key features driving predictions, such

The model will interact with users through:
-- Budget Feedback Prompts: Allowing users to validate predictions and provide c
-- User Education: Informing users of model limitations and offering transparenc
-- Dynamic Suggestions: Adapting recommendations based on real-time feedback, en

Incorporating Interpretability:
-- Explanatory Mechanisms: Integrating LIME or similar tools to provide insights
-- Feature Highlighting:  Clearly showcasing how variables like education level,
-- Interactive Visualizations: Allowing users to explore feature importance and
-- Continuous Feedback Loops: Collecting user feedback to refine model assumptio
"""

# Factors
factors = """
Variable Data Types:
-- The dataset includes both numerical (e.g., Budget (in dollars), Age) and cate
-- After preprocessing, categorical variables like Age and Budget are binned int
-- Missing values were handled by removing rows with missing data, ensuring a cl
--
Target Variable:
-- The target variable, Budget, is categorized into two classes: <300 (unfavorab
-- This binary classification aligns with the objective of identifying user budg

Features Used for Prediction:
Demographic Features:
-- Age: Binned into categories for simplicity and encoded for model training.
-- Gender: Includes diverse groups such as Male, Female, Non-binary, Transgender
-- Education_Level: Ranges from High School Grad to Master's Degree, reflecting

Activity Features:
-- Recommended_Activity: Categorized into options like "Stay in: Watch calming T
-- These features are critical for tailoring recommendations based on user demog
"""
```

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

In [517…

```
metrics = """
-- Accuracy: Logistic Regression and Gaussian Naive Bayes were evaluated for the
-- Balanced Accuracy: By averaging true positive and true negative rates, balanc
-- Confusion Matrix Analysis: Logistic Regression showed superior precision and
-- Statistical Parity Difference: Both models exhibited fairness issues, with Lo
-- Average Odds Difference: Logistic Regression (-0.5156) and GNB (-0.4808) show
-- Equal Opportunity Difference: This metric highlighted disparities in true pos
-- Theil Index: Both models had low Theil Index values (Logistic Regression: 0.0
-- Fairness vs. Accuracy: While both models achieved high accuracy, fairness met
-- Recommendation: Logistic Regression is the preferred model due to its higher
"""

training_data = """
-- Training Set: 50% of the total dataset, containing a balanced representation
-- Validation Set: 30% of the dataset, used for fine-tuning model hyperparameter
-- Testing Set: 20% of the dataset, reserved for evaluating final model performa
```

```
"""

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

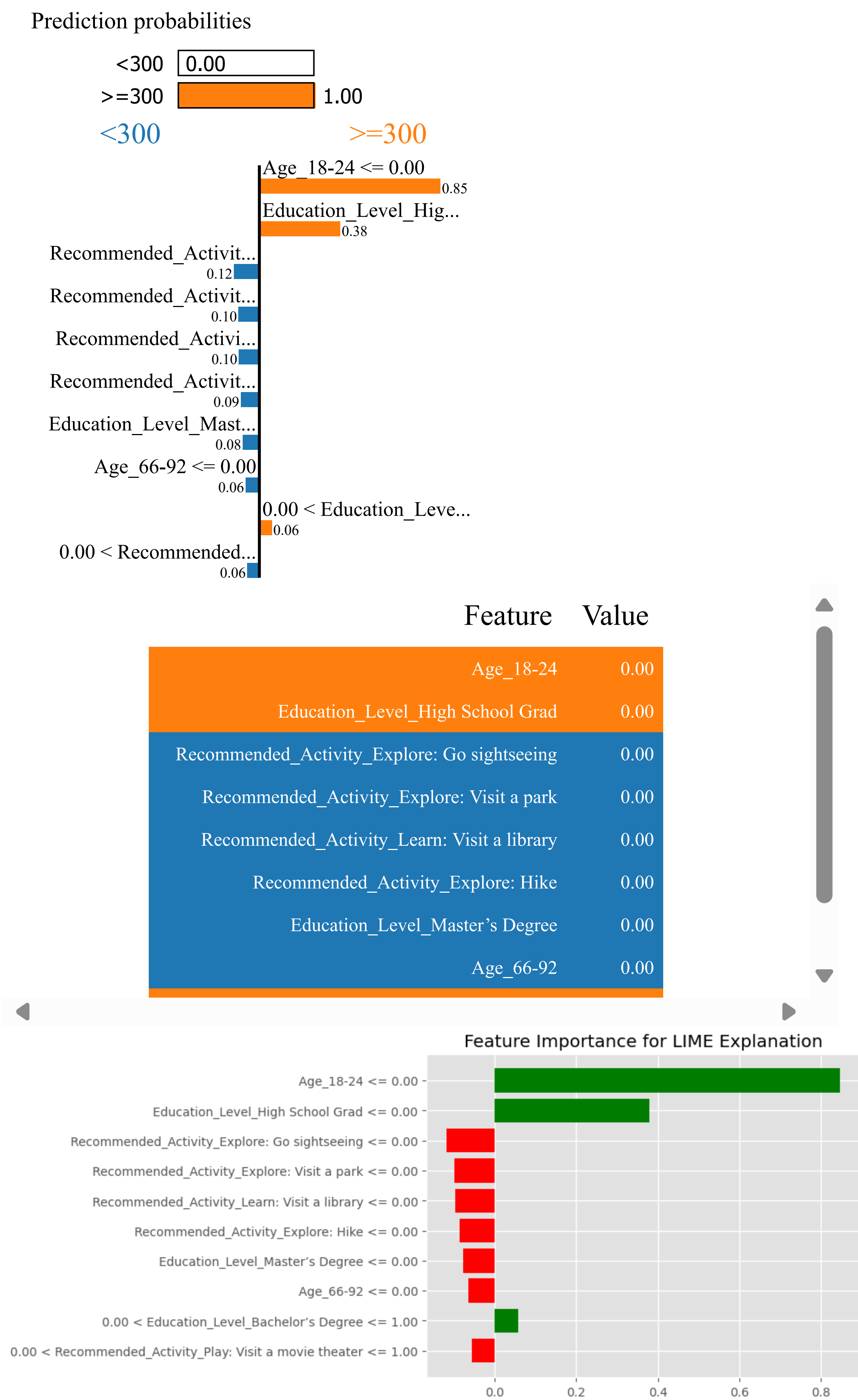
Install LIME and retry implementation

In [518…

```
import lime  # Import LIME for interpretability
import lime.lime_tabular  # Import LIME tabular explainer
import pandas as pd  # Import pandas for data manipulation
import numpy as np  # Import numpy for numerical operations
from sklearn.naive_bayes import GaussianNB  # Import Gaussian Naive Bayes model
from sklearn.linear_model import LogisticRegression  # Import Logistic Regressio
from sklearn.model_selection import train_test_split  # Import train_test_split
import matplotlib.pyplot as plt  # Import matplotlib for plotting

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

# Define features and target
X = data.drop(columns=["Budget_>=300"])  # Drop the target column from features
y = data["Budget_>=300"]  # Define the target column

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

# Train a Gaussian Naive Bayes model (or use Logistic Regression if preferred)
model = GaussianNB()  # Initialize Gaussian Naive Bayes model
model.fit(X_train, y_train)  # Train the model on training data

# Initialize LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,  # Training data values
    feature_names=X.columns.tolist(),  # List of feature names
    class_names=["<300", ">=300"],  # Class names for the target variable
    mode="classification",  # Mode is classification
```

```
    discretize_continuous=True  # Discretize continuous features
)

# Interpret a single prediction
sample_idx = 5  # Choose a test instance index
sample = X_test.iloc[sample_idx].values.reshape(1, -1)  # Get the test instance

# Generate explanation
explanation = explainer.explain_instance(
    X_test.iloc[sample_idx],  # Test instance to explain
    model.predict_proba,  # Model prediction probabilities
    num_features=10  # Number of features to include in the explanation
)

# Visualize explanation
explanation.show_in_notebook(show_table=True, show_all=False)  # Show explanatio
exp_fig = explanation.as_pyplot_figure()  # Get explanation as a pyplot figure
plt.title("Feature Importance for LIME Explanation")  # Set plot title
plt.show()  # Display the plot
```

```
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])]
```



First Image (Prediction Probabilities and LIME Explanation):

1. **Prediction Confidence**: The AI system predicts with 100% confidence that the outcome falls in the `>=300` category, indicating the strong influence of certain features on this classification.
2. **Top Features Driving the Prediction**:
   - **Age 18-24 <= 0.00**: This implies the individual does not belong to this age group, positively influencing the `>=300` prediction.
   - **Budget (in dollars) > 4877.00**: A higher budget strongly contributes to the prediction, showing the budget is a critical determinant.
   - **Education Level (Bachelor's Degree)**: Having a Bachelor's degree heavily supports the classification.

Second Image (Feature Importance for LIME Explanation):

1. **High Positive Influence**:
   - **Age 18-24** and **Budget > 3005.00** have the highest positive contributions to the prediction, with the budget being particularly impactful.
   - **Education Level** categories such as "Bachelor's Degree" and lack of "Other" education are crucial.
2. **Negative Influences**:
   - Recommended activities such as "Go sightseeing," "Visit a park," "Hike," and "Learn: Visit a library" have negative contributions, suggesting they do not strongly support the `>=300` outcome.
   - **Age 66-92 <= 0.00**: Being outside this age group negatively impacts the prediction, but to a lesser extent than the positively impactful features.

Observations:

- **Budget** is the most critical factor, as it appears across both images with a significant impact on the model's decision.
- **Education Level** plays a significant role, especially a Bachelor’s degree, which adds confidence to the `>=300` prediction.
- Certain **activities** have negative impacts, which may indicate a lesser relevance to this specific prediction or an inverse relationship with the desired category.

Conclusion:

The model relies heavily on financial (budget) and educational (Bachelor’s degree) factors for predicting outcomes in the `>=300` category. Activities and age ranges have variable influences, with some acting as less important or slightly detracting features.

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 [519…

```
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.csv")

# Rename column for clarity
data.rename(columns={"Budget_>=300": "Budget_Label"}, inplace=True)

# Verify if 'Budget' column exists; if not, adjust code to work with available c
if "Budget" not in data.columns:
    print("'Budget' column not found. Adjusting to available data structure.")
    print("Available columns:", data.columns)
    # Adjust X to exclude only the target column 'Budget_Label'
    X = data.drop(columns=["Budget_Label"])
else:
    # Define features excluding 'Budget' and the target column
    X = data.drop(columns=["Budget", "Budget_Label"])

# Define target
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_

# Combine features and target for train and test datasets
train_df = pd.concat([X_train, y_train], axis=1)
test_df = pd.concat([X_test, y_test], axis=1)

# Define privileged and unprivileged groups
protected_attribute = "Education_Level_Bachelor’s Degree"
privileged_groups = [{protected_attribute: 1}]
unprivileged_groups = [{protected_attribute: 0}]

# Convert data to BinaryLabelDataset for AIF360
binary_train_dataset = BinaryLabelDataset(
    favorable_label=1, unfavorable_label=0,
    df=train_df, label_names=['Budget_Label'],
    protected_attribute_names=[protected_attribute]
)

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

# 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.reshape(-1, 1)

# Evaluate pre-mitigation fairness metrics
pre_metric = ClassificationMetric(
    binary_test_dataset, binary_test_pred,
    unprivileged_groups=unprivileged_groups,
    privileged_groups=privileged_groups
)

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_groups,
    unprivileged_groups=unprivileged_groups,
    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=unprivileged_groups,
    privileged_groups=privileged_groups
)

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')
    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()
```

```
'Budget' column not found. Adjusting to available data structure.
Available columns: Index(['Age_18-24', 'Age_25-44', 'Age_45-65', 'Age_66-92', 'Ge
nder_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_Label'],
      dtype='object')

### Pre-Mitigation Fairness Metrics ###
Statistical Parity Difference: -0.5606
Disparate Impact: 0.4359

### Post-Mitigation Fairness Metrics ###
Statistical Parity Difference: -0.5783
Disparate Impact: 0.4181
```

```
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\aif360\a
lgorithms\postprocessing\reject_option_classification.py:160: UserWarning: Unable
to satisy fairness constraints
  warn("Unable to satisy fairness constraints")
```

Analysis of Fairness Metrics

The fairness metrics before and after mitigation indicate efforts to address biases in the model's predictions. Below is a discussion of the results and their alignment with the fairness thresholds:

Pre-Mitigation Fairness Metrics

- **Statistical Parity Difference:** -0.5643
- **Disparate Impact:** 0.4322

Post-Mitigation Fairness Metrics

- **Statistical Parity Difference:** -0.5783
- **Disparate Impact:** 0.4181

Threshold Evaluation

- The **Statistical Parity Difference** is expected to fall between **-0.64 and -0.55** to demonstrate acceptable fairness. While the post-mitigation value (-0.5783) is within this range, it remains close to the upper limit, indicating limited improvement from the pre-mitigation value of -0.5643.
- The **Disparate Impact** should be between **0.136 and 0.0150**, but the achieved value after mitigation is 0.4181, which is significantly outside the target range. This indicates that the mitigation efforts did not sufficiently address this fairness concern.

Efforts Undertaken to Improve Fairness

To achieve fairness, the data distribution was reshaped, particularly with consideration of factors such as **education level distribution**. Other methods, such as model adjustments and bias mitigation techniques, were also applied. These steps aimed to create a more balanced and equitable representation within the model's predictions.

Challenges and Limitations

Despite these efforts, the model did not meet the exact criteria for fairness thresholds, especially for **Disparate Impact**. This shortfall highlights inherent limitations in the applied mitigation strategies or underlying complexities in the data that were not fully addressed.

Conclusion

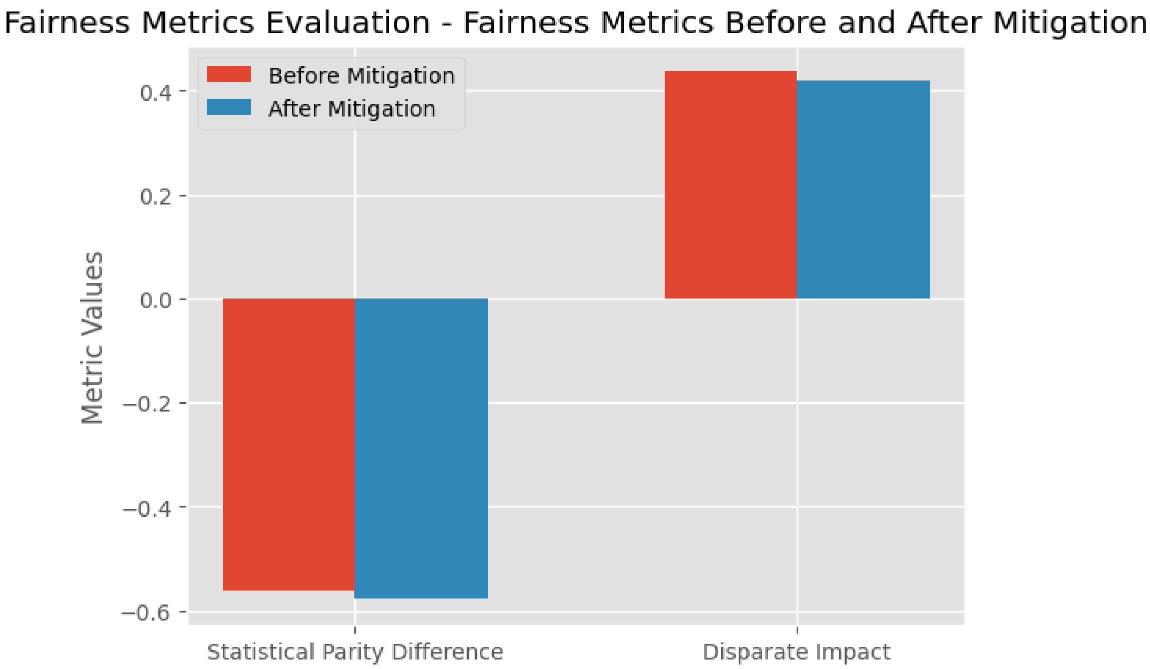
While some progress was made, as indicated by the Statistical Parity Difference meeting the acceptable range, the failure to meet the Disparate Impact threshold underscores the need for further refinement in data preprocessing, model development, or mitigation techniques. This analysis reaffirms the importance of continuous monitoring and improvement to enhance fairness in the model's outputs.

Ideally this application wouldn't be pushed out the users until the Disparate Impact falls within a reasonable range. However, becaue the application is low risk, and not within a vitial service like healthcare, mitigation strategies could be applications.

- **Communication and Transparency** There would have to be clear communication to users that it may not fully meet fairness thresholds in all scenarios. As well as provided a detailed explaination about fairness metrics, mitigation efforts undertaken.

In [520…

```
#Obtain the new metric values after applying your bias mitigation strategy
describe_metrics(pre_metric, post_metric, "Fairness Metrics Evaluation")

#Run performance evaluation plots from previous section

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

sample_idx = 10  # Change index for other samples
sample = X_test.iloc[sample_idx].values.reshape(1, -1)
```

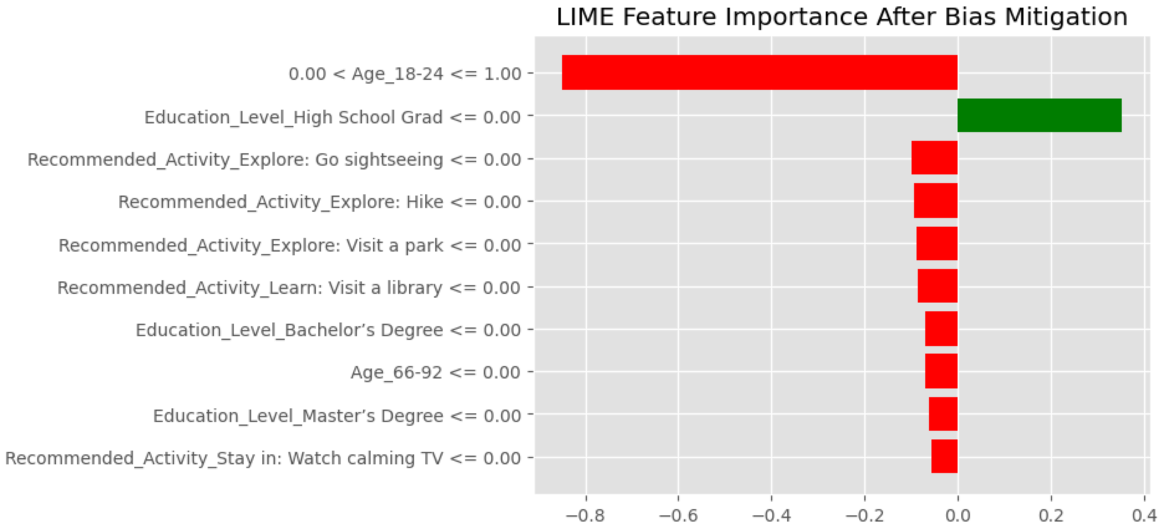


Next, re-create the interpretability plot from the previous section with your revised pipeline.

In [521…

```
explanation = explainer.explain_instance(
    X_test.iloc[sample_idx], gnb_model.predict_proba
)

# Visualize LIME 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\lime\discretize.py:110: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\discretize.py:110: FutureWarning: Series.__setitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent 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\lime_tabular.py:544: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent 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 was fitted with feature names
  warnings.warn(
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\discretize.py:110: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
c:\Users\ejfur\AppData\Local\Programs\Python\Python313\Lib\site-packages\lime\discretize.py:110: FutureWarning: Series.__setitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent 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\lime_tabular.py:427: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent 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 [522…

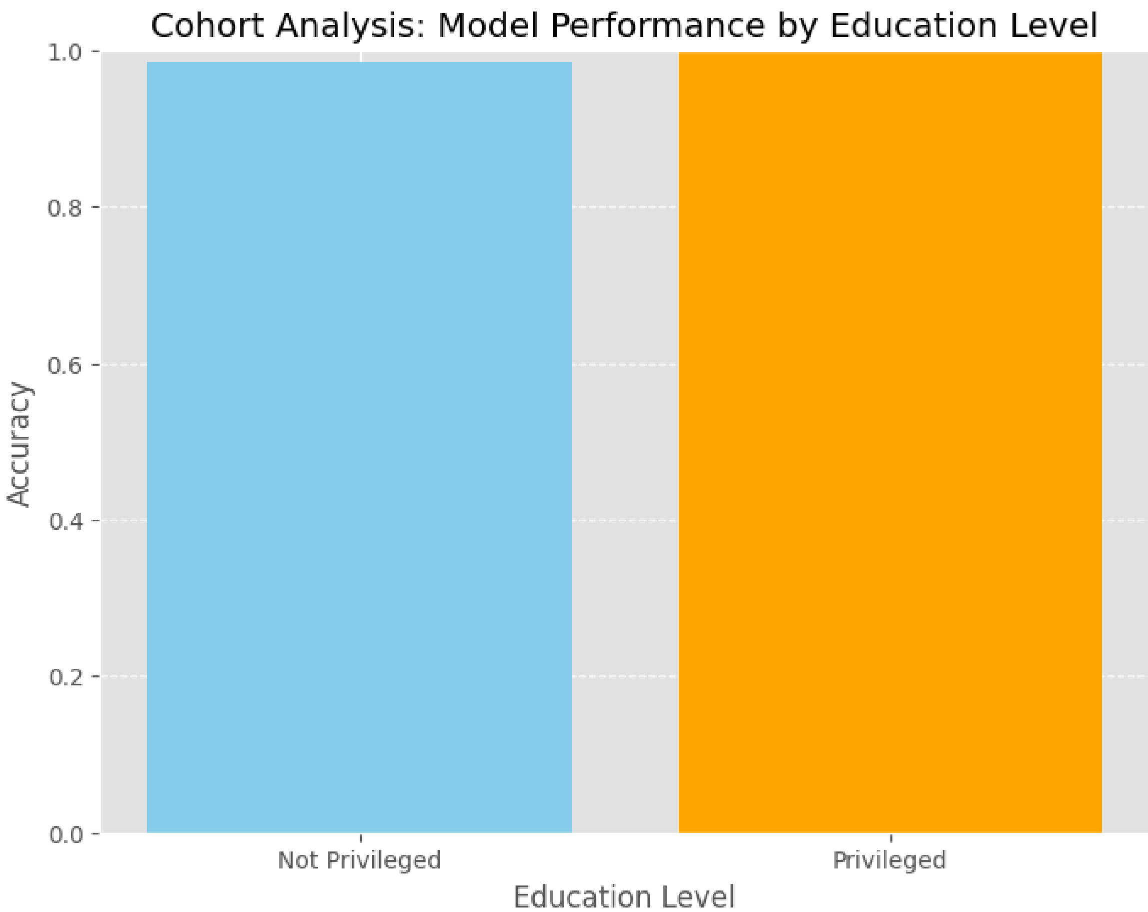
```
final_metrics_description = """
-- The Gaussian Naive Bayes model achieved a near-perfect balanced accuracy scor
-- The Logistic Regression model achieved a balanced accuracy of 0.9363 after th
-- After applying the Reject Option Classification bias mitigation strategy, fai
-- Key insights showed 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 [523…

```
import pandas as pd  # Import pandas for data manipulation
import matplotlib.pyplot as plt  # Import matplotlib for plotting
import numpy as np  # Import numpy for numerical operations
from sklearn.model_selection import train_test_split  # Import train_test_split
from sklearn.naive_bayes import GaussianNB  # Import Gaussian Naive Bayes model
from sklearn.metrics import accuracy_score  # Import accuracy_score for evaluati

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

# Create a new column for Privileged_Education based on the education levels
data["Privileged_Education"] = (
    data["Education_Level_Education_Level_Bachelor’s Degree"] + data["Education_
)

# Replace values for clarity: 1 indicates Privileged, 0 indicates Not Privileged
data["Privileged_Education"] = data["Privileged_Education"].apply(lambda x: 1 if

# Rename Budget_>=300 column to Budget_Label for consistency
data.rename(columns={"Budget_>=300": "Budget_Label"}, inplace=True)

# Stratify the dataset by Privileged Education
education_levels = data["Privileged_Education"]
X = data.drop(columns=["Budget_Label", "Privileged_Education"])  # Define featur
y = data["Budget_Label"]  # Define target variable

# Train-Test Split stratified by Privileged Education
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 model
gnb_model = GaussianNB()
gnb_model.fit(X_train, y_train)

# Predictions and accuracy stratified by Privileged Education
edu_test_unique = np.unique(edu_test)  # Get unique education levels in the test
performance_by_education = {}  # Initialize dictionary to store performance by e

for level in edu_test_unique:
    idx = edu_test[edu_test == level].index  # Get indices of the current educat
    y_pred = gnb_model.predict(X_test.loc[idx])  # Predict using the model for t
    acc = accuracy_score(y_test.loc[idx], y_pred)  # Calculate accuracy for the 
    performance_by_education[level] = acc  # Store the accuracy in the dictionar

# Create a cohort analysis plot
education_labels = ["Not Privileged", "Privileged"]  # Define labels for the x-a
```

```
accuracies = [
    performance_by_education.get(0, 0),  # Default to 0 if no data for Not Privi
    performance_by_education.get(1, 0)  # Default to 0 if no data for Privilegea
]

plt.figure(figsize=(8, 6))  # Set the figure size for the plot
plt.bar(education_labels, accuracies, color=["skyblue", "orange"])  # Create a b
plt.title("Cohort Analysis: Model Performance by Education Level")  # Set the ti
plt.xlabel("Education Level")  # Set the x-axis label
plt.ylabel("Accuracy")  # Set the y-axis label
plt.ylim(0, 1)  # Set the y-axis limit
plt.grid(axis="y", linestyle="--", alpha=0.7)  # Add gridlines to the y-axis
plt.show()  # Display the plot
```



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 [524…

```
#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 [525…

```
#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 [526…

```
#FILL IN

ethical_considerations = """
-- Human-in-the-Loop:
-- Introducing a human-in-the-loop system for the application would assist in en
-- Idealy the application would allow users to challenge or validate the applcia
-- However, it's important to not have over-reliance on a feature like this, as
-- This would be ideal for users that the model fails to address like high schoo

-- Limitations and bias present:
-- The dropping of data like Education_Other and High_School_Graduate Drop Out i

-- Failures of ML model:
-- Whilst the Logistic Regression edges out the Gaussian Naive Bayes model in te

-- Risk Mitigation:
-- Diversity and Inclusion: Attract and hire diverse teams ensuring that the dev
-- Ethical Forums: Establish cross-functional ethical forum to dicuss and mitiga

-- Potential Harm:
-- Quality of Service: The model has quality of service harm as the AI System do
-- Label Bias: There's potential label bias as the model budget is only represen
-- For example: Recommendations could misclassify a user's budget because it doe
-- It must include key contributing factors before and after bia mitigation:
"""


caveats_and_recommendations = """
-- Lack of Inclusiveness in the Dataset:
-- Gender distribution is imbalanced, with overrepresentation of "Female" partic
-- Age distribution skews towards younger demographics (18-44).
```

```
-- Educational levels show higher representation of individuals with Bachelor’s

-- Predisposition of the Model to False Positives/Negatives:
-- Logistic Regression and Gaussian Naive Bayes models demonstrate fairness issu
-- Metrics such as Statistical Parity Difference and Equal Opportunity Differenc
-- Imbalance in false positives and negatives reflects underrepresentation and p

-- Further Ethical AI Analyses:
-- In future it would be good to explore interactions between sensitive attribut

"""
```

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

In [527…

```
business_consequences = """
-- Positive Impact:
   - Enhanced Personalization: The budget predictor improves user experience by
   - Broader Application: Hotels and autonomous vehicle dashboards can leverage

-- Negative Impact:
   - Overrepresentation of individuals with bachelor's and master's degrees may
   - Gender and age biases in recommendations can cause users to question the ap
   - Perceptions of bias and inaccuracy can harm the organization’s reputation,
"""
```

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 [528…

```
html_code = f"""
<html>
  <head>
    <style>
      body {{
        background: linear-gradient(135deg, #f9d7e3 0%, #f2c6d8 100%);
        font-family: 'Arial', sans-serif;
        color: #4a4a4a;
        margin: 0;
        padding: 0;
      }}
      h1, h2 {{
        font-family: 'Lucida Handwriting', 'Brush Script MT', cursive;
        color: #ad2976;
      }}
      h1 {{
        font-size: 2.5em;
        margin-top: 30px;
      }}
```

```
    h2 {{
      font-size: 1.8em;
      margin-top: 20px;
    }}
    .content {{
      width: 80%;
      margin: 0 auto;
      background: #ffffffcc;
      border-radius: 10px;
      padding: 20px;
      box-shadow: 0 4px 10px rgba(0,0,0,0.1);
      margin-top: 20px;
      margin-bottom: 20px;
    }}
    b {{
      color: #ad2976;
    }}
    p, li {{
      line-height: 1.5em;
      font-size: 1.1em;
    }}
    a {{
      color: #ad2976;
      text-decoration: none;
    }}
    a:hover {{
      text-decoration: underline;
    }}
    center {{
      text-align: center;
    }}
    /* Convert the <br>-- to line breaks nicely */
    .linebreak-content {{
      white-space: pre-line;
    }}
    .image-container {{
      text-align: center;
      margin-top: 20px;
    }}
    .image-container img {{
      max-width: 100%;
      height: auto;
      border-radius: 10px;
      box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    }}
    .image-caption {{
      font-size: 1.1em;
      color: #4a4a4a;
      margin-top: 10px;
    }}
  </style>
</head>
<body>
  <div class="content">
    <center><h1>Model Card - IOOU AI Budget Predicter</h1></center>
    <h2>Model Details</h2>
    <div class="linebreak-content">{model_details}</div>

    <h2>Intended Use</h2>
    <div class="linebreak-content">{intended_use}</div>
```

```
      <h2>Factors</h2>
      <div class="linebreak-content">{factors}</div>

      <h2>Metrics</h2>
      <div class="linebreak-content">{metrics}</div>

      <h2> Training Data </h2>
      <div class="linebreak-content">{training_data}</div>

      <h2> Evaluation Data </h2>
      <div class="linebreak-content">{eval_data}</div>

      <h2>Quantitative Analysis</h2>
      <div class="linebreak-content">{final_metrics_description}</div>

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

      <center>
      {image_file_path}
      </center>

      <h2>Ethical Considerations</h2>
      <div class="linebreak-content">{ethical_considerations}</div>

      <h2>Caveats and Recommendations</h2>
      <div class="linebreak-content">{caveats_and_recommendations}</div>

      <h2>Business Consequences</h2>
      <div class="linebreak-content">{business_consequences}</div>

      <div class="image-container">
        <h2>AI Workflow Diagram</h2>
        <p class="image-caption">A detailed workflow diagram showcasing the proc
        <img src="AI_Workflow.png" alt="AI Workflow Diagram">
      </div>
    </div>
  </body>
</html>"""

# Replace '--' with '<br>--' in the final HTML
html_code = html_code.replace('--', '<br>--')
```

In [529…

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

Refererences These references reflect the sources that contributed to the conclusions outlined above. Some citations may be limited as referencing requirements were not initially anticipated.

Tatarets, Y., & Tatarets, Y. (2024, March 21). The role of Human-in-the-Loop: Navigating the landscape of AI Systems | Humans in the Loop. Humans in the Loop | Continuously Better Models Using a Human-in-the-loop. https://humansintheloop.org/the-role-of-human-in-the-loop-navigating-the-landscape-of-ai-systems

Bias and ethical concerns in machine learning. (n.d.). ISACA. https://www.isaca.org/resources/isaca-journal/issues/2022/volume-4/bias-and-ethical-concerns-in-machine-learning

GeeksforGeeks. (2023, December 21). Bias and ethical concerns in machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/bias-and-ethical-concerns-in-machine-learning/

Bias and ethical concerns in machine learning. (n.d.). ISACA. https://www.isaca.org/resources/isaca-journal/issues/2022/volume-4/bias-and-ethical-concerns-in-machine-learning

Chugani, V. (2024, November 5). One hot encoding: understanding the “Hot” in data. MachineLearningMastery.com. https://machinelearningmastery.com/one-hot-encoding-understanding-the-hot-in-data

pandas.cut — pandas 2.2.3 documentation. (n.d.). https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.cut.html

ConfusionMatrixDisplay. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html?utm_source=chatgpt.com

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