```
Name:- Shritej Kalekar
PRN NO.22SC114501002
Class:- B.Tech AI&ML

TITLE:- Impact of Data Quality on AI Fairness.
```

Using the Pima Indians Diabetes Dataset (a public health prediction dataset):-

Task

apply fairness mitigation, and evaluate the impact of mitigation on fairness.

Load a new dataset

Subtask:

Load a suitable dataset for binary classification that contains a potential sensitive attribute.

Reasoning: Load the Pima Indians Diabetes dataset from a URL and display the head and info to understand its structure.

```
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/diat
display(df.head())
display(df.info())
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-919053710.py in <cell line: 0>()
----> 1 df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/diabe
      2 display(df.head())
      3 display(df.info())
```

Next steps: Explain error

Reasoning: The previous command failed because pandas was not imported. I need to import pandas before using it to load the data.

```
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/diak
display(df.head())
display(df.info())
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete:
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

Identify sensitive attribute and target variable

Subtask:

Identify the column representing the sensitive attribute and the column representing the target variable in the loaded dataset.

Preprocess the data

Subtask:

Handle missing values, encode categorical features, and scale numerical features as needed for the new dataset.

Reasoning: Check for missing values in the DataFrame.

```
display(df.isnull().sum())
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

Reasoning: Since there are no missing values and no categorical features to encode, scale the numerical features and prepare the data for modeling.

```
from sklearn.preprocessing import StandardScaler

# Identify numerical features to scale (all except the target variable and pote
# In this case, we will scale all features except the target 'Outcome'. The 'Ag
numerical_features = df.drop(columns=['Outcome']).columns

scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(df[numerical_features]), columns=r
y = df['Outcome']
sensitive = df['Age'] # Keeping Age as the sensitive attribute as identified pr

display(X_scaled.head())
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Di
0	0.639947	0.848324	0.149641	0.907270	-0.692891	0.204013	
1	-0.844885	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	
2	1.233880	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	
3	-0.844885	-0.998208	-0.160546	0.154533	0.123302	-0.494043	
4	-1.141852	0.504055	-1.504687	0.907270	0.765836	1.409746	

Split data

Subtask:

Split the dataset into training and testing sets, ensuring stratification based on the sensitive attribute or target variable.

Reasoning: Split the scaled features, target variable, and sensitive attribute into training and testing sets, stratifying by the target variable.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test, s_train, s_test = train_test_split(
    X_scaled, y, sensitive, test_size=0.3, random_state=42, stratify=y
)

print("Training set sizes:")
print(f"X_train shape: {X_train.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"s_train shape: {s_train.shape}")
print("\nTesting set sizes:")
print(f"X_test shape: {X_test.shape}")
print(f"y_test shape: {y_test.shape}")
print(f"s_test shape: {s_test.shape}")

# Verify stratification by target variable
print("\nTarget distribution in training set:")
print(y_train.value_counts(normalize=True))
print("\nTarget distribution in testing set:")
print(y_test.value_counts(normalize=True))
```

```
Training set sizes:
X_train shape: (537, 8)
y_train shape: (537,)
s_train shape: (537,)

Testing set sizes:
X_test shape: (231, 8)
y_test shape: (231,)
s_test shape: (231,)

Target distribution in training set:
Outcome
0    0.651769
1    0.348231
Name: proportion, dtype: float64

Target distribution in testing set:
Outcome
0    0.649351
1    0.350649
Name: proportion, dtype: float64
```

Train baseline model

Subtask:

Train a standard classification model (e.g., Logistic Regression) on the training data.

Reasoning: Train a standard logistic regression model on the training data and make predictions on the test data as instructed.

```
from sklearn.linear_model import LogisticRegression

# Instantiate the model
clf = LogisticRegression(max_iter=1000)

# Fit the model to the training data
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)
```

Evaluate fairness before mitigation

Subtask:

Compute and display fairness metrics (e.g., TPR, FPR, Selection Rate) for different groups based on the sensitive attribute to identify disparities.

Reasoning: Compute fairness metrics for the baseline model's predictions on the test set, grouped by the sensitive attribute.

```
from fairlearn.metrics import MetricFrame, true_positive_rate, false_positive_r

# Create a MetricFrame to compute fairness metrics
metric_frame = MetricFrame(
    metrics={
        'TPR': true_positive_rate,
        'FPR': false_positive_rate,
        'Selection Rate': selection_rate
    },
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=s_test
)

# Print the fairness metrics for each group
print("Fairness Metrics by Sensitive Attribute Group:\n", metric_frame.by_grouf
```

```
---------------------------------------------------------------------------
-
ModuleNotFoundError                       Traceback (most recent call
last)
/tmp/ipython-input-1083434407.py in <cell line: 0>()
----> 1 from fairlearn.metrics import MetricFrame, true_positive_rate,
false_positive_rate, selection_rate
      2
      3 # Create a MetricFrame to compute fairness metrics
      4 metric_frame = MetricFrame(
      5     metrics={

ModuleNotFoundError: No module named 'fairlearn'

---------------------------------------------------------------------------
-
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

Next steps: Explain error

Reasoning: The previous command failed because the `fairlearn` library was not found. Although `fairlearn` was installed in a previous cell, it seems the kernel might have been restarted or the environment is not recognizing the installed package. I will re-install the `fairlearn` package to ensure it is available.

```
pip install fairlearn
```

```
Collecting fairlearn
  Downloading fairlearn-0.12.0-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: numpy>=1.24.4 in /usr/local/lib/python3.11/d
Requirement already satisfied: pandas>=2.0.3 in /usr/local/lib/python3.11/d
Requirement already satisfied: scikit-learn>=1.2.1 in /usr/local/lib/python
Requirement already satisfied: scipy>=1.9.3 in /usr/local/lib/python3.11/di
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Downloading fairlearn-0.12.0-py3-none-any.whl (240 kB)
   240.0/240.0 kB 5.6 MB/s eta 0:0
Installing collected packages: fairlearn
Successfully installed fairlearn-0.12.0
```

Reasoning: The `fairlearn` package has been successfully installed. Now I can proceed with the original task of computing and displaying the fairness metrics using `MetricFrame`.

```
from fairlearn.metrics import MetricFrame, true_positive_rate, false_positive_r

# Create a MetricFrame to compute fairness metrics
metric_frame = MetricFrame(
    metrics={
        'TPR': true_positive_rate,
        'FPR': false_positive_rate,
        'Selection Rate': selection_rate
    },
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=s_test
)

# Print the fairness metrics for each group
print("Fairness Metrics by Sensitive Attribute Group:\n", metric_frame.by_grour
```

```
Fairness Metrics by Sensitive Attribute Group:
          TPR       FPR  Selection Rate
Age
21   1.000000  0.000000        0.050000
22   0.750000  0.041667        0.142857
23   0.500000  0.230769        0.266667
24   0.000000  0.142857        0.125000
25   0.666667  0.000000        0.181818
26   0.500000  0.222222        0.307692
27   0.333333  0.125000        0.181818
28   0.000000  0.000000        0.000000
29   0.166667  0.000000        0.083333
30   0.500000  0.200000        0.285714
31   0.600000  0.000000        0.428571
32   0.250000  0.000000        0.166667
33   0.000000  0.000000        0.000000
34   0.000000  1.000000        0.333333
35   0.333333  0.000000        0.333333
36   1.000000  0.000000        0.400000
37   1.000000  0.000000        0.200000
38   0.500000  0.000000        0.500000
39   1.000000  0.000000        0.333333
40   0.000000  1.000000        1.000000
41   0.500000  0.500000        0.500000
42   0.000000  1.000000        0.333333
43   1.000000  0.000000        1.000000
45   0.500000  0.333333        0.400000
46   1.000000  0.000000        0.250000
47   0.000000  0.000000        0.000000
48   1.000000  0.000000        0.500000
49   1.000000  1.000000        1.000000
50   0.500000  0.000000        0.333333
51   1.000000  1.000000        1.000000
52   0.000000  0.000000        0.000000
53   0.000000  0.000000        0.000000
54   0.500000  1.000000        0.666667
55   0.000000  0.000000        0.000000
56   0.000000  0.000000        0.000000
58   0.000000  0.000000        0.000000
59   1.000000  0.000000        1.000000
60   0.000000  1.000000        0.500000
61   1.000000  0.000000        1.000000
62   1.000000  0.000000        1.000000
63   0.000000  0.000000        0.000000
66   1.000000  0.000000        1.000000
67   0.000000  0.500000        0.500000
68   0.000000  0.000000        0.000000
69   0.000000  0.000000        0.000000
```

Apply fairness mitigation

Subtask:

Use a fair learning algorithm (e.g., Exponentiated Gradient with Demographic Parity constraint) to train a mitigated model.

Reasoning: Instantiate and fit the ExponentiatedGradient model with the DemographicParity constraint and make predictions on the test set.

```
from fairlearn.reductions import ExponentiatedGradient, DemographicParity

# Instantiate a DemographicParity constraint object.
constraint = DemographicParity()

# Instantiate an ExponentiatedGradient object
expgrad_model = ExponentiatedGradient(
    estimator=LogisticRegression(max_iter=1000),
    constraints=constraint,
    eps=0.01
)

# Fit the ExponentiatedGradient model to the training data
expgrad_model.fit(X_train, y_train, sensitive_features=s_train)

# Make predictions on the test data using the fitted ExponentiatedGradient mode
y_pred_fair = expgrad_model.predict(X_test)
```

Evaluate fairness after mitigation

Subtask:

Compute and display fairness metrics for the mitigated model to assess the impact of the mitigation technique on fairness.

Reasoning: Compute and display fairness metrics for the mitigated model's predictions, including TPR, FPR, and Selection Rate, grouped by the sensitive attribute.

```
# Create a MetricFrame for the mitigated model
```

```
metric_frame_fair = MetricFrame(
    metrics={
        'TPR': true_positive_rate,
        'FPR': false_positive_rate,
        'Selection Rate': selection_rate
    },
    y_true=y_test,
    y_pred=y_pred_fair,
    sensitive_features=s_test
)

# Print the fairness metrics for each group after mitigation
print("\nFairness Metrics by Sensitive Attribute Group (After Mitigation):\n",
```

Fairness Metrics by Sensitive Attribute Group (After Mitigation):

Age	TPR	FPR	Selection Rate
21	1.000000	0.421053	0.450000
22	0.750000	0.541667	0.571429
23	1.000000	0.153846	0.266667
24	0.000000	0.285714	0.250000
25	0.666667	0.250000	0.363636
26	0.250000	0.444444	0.384615
27	0.333333	0.375000	0.363636
28	0.000000	0.500000	0.454545
29	0.500000	0.333333	0.416667
30	1.000000	0.400000	0.571429
31	0.200000	0.500000	0.285714
32	0.500000	0.500000	0.500000
33	0.000000	0.000000	0.000000
34	0.500000	1.000000	0.666667
35	0.666667	0.000000	0.666667
36	0.500000	0.666667	0.600000
37	1.000000	0.250000	0.400000
38	0.333333	0.000000	0.333333
39	0.000000	0.500000	0.333333
40	0.000000	1.000000	1.000000
41	0.833333	1.000000	0.875000
42	0.500000	1.000000	0.666667
43	0.500000	0.000000	0.500000
45	0.000000	0.666667	0.400000
46	1.000000	0.333333	0.500000
47	0.000000	0.000000	0.000000
48	0.000000	1.000000	0.500000
49	0.500000	0.000000	0.333333
50	0.500000	0.000000	0.333333
51	0.000000	0.500000	0.250000
52	0.000000	0.000000	0.000000
53	0.000000	1.000000	1.000000
54	0.500000	0.000000	0.333333
55	0.000000	0.000000	0.000000
56	0.000000	0.000000	0.000000
58	0.000000	0.000000	0.000000
59	1.000000	0.000000	1.000000
60	0.000000	1.000000	0.500000
61	0.000000	0.000000	0.000000
62	1.000000	0.000000	1.000000
63	0.000000	0.000000	0.000000
66	1.000000	0.000000	1.000000
67	0.000000	0.500000	0.500000
68	0.000000	1.000000	1.000000
69	0.000000	1.000000	1.000000

Compare results

Subtask:

Compare the fairness metrics before and after mitigation to demonstrate the effectiveness of the technique.

Reasoning: Access and display the fairness metrics before and after mitigation for comparison.

```
print("==== Fairness Metrics Before Mitigation ====")
print(metric_frame.by_group)

print("\n==== Fairness Metrics After Mitigation ====")
print(metric_frame_fair.by_group)
```

```
56   0.000000  0.000000        0.000000
58   0.000000  0.000000        0.000000
59   1.000000  0.000000        1.000000
60   0.000000  1.000000        0.500000
61   1.000000  0.000000        1.000000
62   1.000000  0.000000        1.000000
63   0.000000  0.000000        0.000000
66   1.000000  0.000000        1.000000
67   0.000000  0.500000        0.500000
68   0.000000  0.000000        0.000000
69   0.000000  0.000000        0.000000

==== Fairness Metrics After Mitigation ====
          TPR       FPR  Selection Rate
Age
21   1.000000  0.421053        0.450000
22   0.750000  0.541667        0.571429
23   1.000000  0.153846        0.266667
24   0.000000  0.285714        0.250000
25   0.666667  0.250000        0.363636
26   0.250000  0.444444        0.384615
27   0.333333  0.375000        0.363636
28   0.000000  0.500000        0.454545
29   0.500000  0.333333        0.416667
30   1.000000  0.400000        0.571429
31   0.200000  0.500000        0.285714
32   0.500000  0.500000        0.500000
33   0.000000  0.000000        0.000000
34   0.500000  1.000000        0.666667
35   0.666667  0.000000        0.666667
36   0.500000  0.666667        0.600000
```

```
37   1.000000  0.250000        0.400000
38   0.333333  0.000000        0.333333
39   0.000000  0.500000        0.333333
40   0.000000  1.000000        1.000000
41   0.833333  1.000000        0.875000
42   0.500000  1.000000        0.666667
43   0.500000  0.000000        0.500000
45   0.000000  0.666667        0.400000
46   1.000000  0.333333        0.500000
47   0.000000  0.000000        0.000000
48   0.000000  1.000000        0.500000
49   0.500000  0.000000        0.333333
50   0.500000  0.000000        0.333333
51   0.000000  0.500000        0.250000
52   0.000000  0.000000        0.000000
53   0.000000  1.000000        1.000000
54   0.500000  0.000000        0.333333
55   0.000000  0.000000        0.000000
56   0.000000  0.000000        0.000000
58   0.000000  0.000000        0.000000
59   1.000000  0.000000        1.000000
60   0.000000  1.000000        0.500000
61   0.000000  0.000000        0.000000
62   1.000000  0.000000        1.000000
63   0.000000  0.000000        0.000000
66   1.000000  0.000000        1.000000
67   0.000000  0.500000        0.500000
68   0.000000  1.000000        1.000000
```

Summarize findings

Subtask:

Provide a summary of the analysis and the impact of fairness mitigation on the new dataset.

Reasoning: Provide a summary of the analysis and the impact of fairness mitigation on the new dataset by comparing the fairness metrics before and after mitigation and discussing the observed changes.

```
print("Summary of Fairness Analysis and Mitigation Impact on the Pima Indians D

print("Fairness Metrics BEFORE Mitigation (by Age Group):")
print(metric_frame.by_group)

print("\nFairness Metrics AFTER Mitigation (by Age Group):")
print(metric_frame_fair.by_group)
```

```
print("\nImpact of Fairness Mitigation:")
print("The analysis of the Pima Indians Diabetes dataset revealed disparities i

print("\nObserving the 'Fairness Metrics AFTER Mitigation' table, after applyir

print("\nComparing the 'Selection Rate' column before and after mitigation, the

print("\nFor other metrics like TPR and FPR, the impact of mitigation is also v

print("\nOverall, the application of fairness mitigation techniques like Expone
```

```
Fairness Metrics AFTER Mitigation (by Age Group):
          TPR       FPR  Selection Rate
Age
21   1.000000  0.421053        0.450000
22   0.750000  0.541667        0.571429
23   1.000000  0.153846        0.266667
24   0.000000  0.285714        0.250000
25   0.666667  0.250000        0.363636
26   0.250000  0.444444        0.384615
27   0.333333  0.375000        0.363636
28   0.000000  0.500000        0.454545
29   0.500000  0.333333        0.416667
30   1.000000  0.400000        0.571429
31   0.200000  0.500000        0.285714
32   0.500000  0.500000        0.500000
33   0.000000  0.000000        0.000000
34   0.500000  1.000000        0.666667
35   0.666667  0.000000        0.666667
36   0.500000  0.666667        0.600000
37   1.000000  0.250000        0.400000
38   0.333333  0.000000        0.333333
39   0.000000  0.500000        0.333333
40   0.000000  1.000000        1.000000
41   0.833333  1.000000        0.875000
42   0.500000  1.000000        0.666667
43   0.500000  0.000000        0.500000
45   0.000000  0.666667        0.400000
46   1.000000  0.333333        0.500000
47   0.000000  0.000000        0.000000
48   0.000000  1.000000        0.500000
49   0.500000  0.000000        0.333333
50   0.500000  0.000000        0.333333
51   0.000000  0.500000        0.250000
52   0.000000  0.000000        0.000000
53   0.000000  1.000000        1.000000
54   0.500000  0.000000        0.333333
55   0.000000  0.000000        0.000000
56   0.000000  0.000000        0.000000
58   0.000000  0.000000        0.000000
59   1.000000  0.000000        1.000000
```

```
60   0.000000  1.000000        0.500000
61   0.000000  0.000000        0.000000
62   1.000000  0.000000        1.000000
63   0.000000  0.000000        0.000000
66   1.000000  0.000000        1.000000
67   0.000000  0.500000        0.500000
68   0.000000  1.000000        1.000000
69   0.000000  1.000000        1.000000

Impact of Fairness Mitigation:
The analysis of the Pima Indians Diabetes dataset revealed disparities in

Observing the 'Fairness Metrics AFTER Mitigation' table, after applying tl

Comparing the 'Selection Rate' column before and after mitigation, there a

For other metrics like TPR and FPR, the impact of mitigation is also visil
```

Summary:

Data Analysis Key Findings

- The dataset used for this analysis is the Pima Indians Diabetes dataset, containing information about female patients and whether they have diabetes ('Outcome').
- The 'Age' column was identified as the sensitive attribute, and the 'Outcome' column (0 or 1) is the binary target variable.
- The dataset has no missing values and contains only numerical features, which were scaled using `StandardScaler`.
- The data was split into training (70%) and testing (30%) sets, stratified by the target variable 'Outcome'.
- A baseline Logistic Regression model was trained and evaluated, revealing disparities in fairness metrics (True Positive Rate, False Positive Rate, and Selection Rate) across different age groups.
- The Exponentiated Gradient algorithm with a Demographic Parity constraint was applied as a fairness mitigation technique. Demographic Parity aims to equalize the Selection Rate across different groups.
- After applying mitigation, the fairness metrics for the mitigated model were evaluated and compared to the baseline model. The mitigation influenced the distribution of positive predictions across age groups, attempting to reduce disparities in the Selection Rate. Changes were also observed in TPR and FPR across groups, highlighting potential trade-offs.

Insights or Next Steps

- Quantify the reduction in disparity for each fairness metric (e.g., using disparity metrics like the difference between the maximum and minimum group values) before and after mitigation to provide a more concrete measure of effectiveness.
- Explore alternative fairness mitigation techniques (e.g., other reduction algorithms or pre-processing/in-processing methods) and compare their impact on fairness and overall model performance to find the most suitable approach for this dataset and task.