Introduction

Bias mitigation is one of the crucial step in data analysis, especially when you are dealing with Protected class variable. A protected class is a grouping of people who are qualified for superior protection by law, or any other special rights. In Canada and USA this class is usually connected with employees, employment and housing. In these domain, unlawful discrimination has been done on the basis of these protected classes. It may be related to religion, ethnicity, national origin, gender, age etc. Also the discrimination against pregnant women's or disable persons.

While performing data analysis, especially when you are using Machine Learning algorithm a high risk of biasness has been attaining. Bias in various forms can infiltrate data, algorithms, and decision-making processes, leading to unfair and discriminatory outcomes. Recognizing and addressing bias is crucial for creating equitable systems. The model on the data set which contain biasness lead to a wrong perdition or analysis. It is important to identify and remove these biases before performing any analysis on your data set. Over the times, lots of techniques and methods has been used for bias identification and mitigation such as:

Types of biases:

1. Algorithmic Bias
2. Data Bias
3. Selection Bias
4. Confirmation Bias

Identification methods:

1. Diversity assessment
2. Autcome analysis
3. Faieness techniques

Bias Mitigation Techniques:

1. Divers and representative data collection
2. Pre- processing techniques
3. Algorithmic fairness
4. Explianaility and transparency
5. Continuous monitoring and evaluation

In this task we focus on Algorithmic bias and data bias. We used fairness techniques to identify the bias in my data set. For bias mitigation, we used two approaches in an iterative manner. First we apply the model and then calculate algorithmic fairness after that we apply bias mitigation algorithm on our data set, and again apply model on our transformed data set and compare the result.

Data set description:

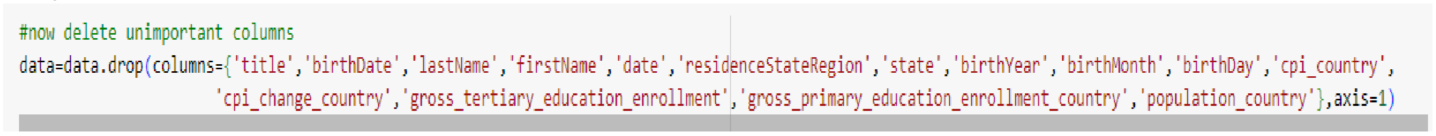
For this specific task we choose the data set from kaggle machine learning repository. [https://www.kaggle.com/datasets/nelgiriyewithana/billionaires-statistics-dataset]. This is Billionaire statistics data set 2023. This dataset originally consists of 2640 instance and 18 variables that are:

1. Rank
2. FinalWorth

3. Category,
4. Personname(ALEX) ,
5. Age
6. Gender
7. Country
8. City
9. Industries
10. Organization
11. Self made
12. Status
13. Birthdate
14. First name
15. Last name
16. countryOfCitizenship
17. cpi_change_country
18. gross_tertiary_education_enrollment
19. gross_primary_education_enrollment_country
20. total_tax_rate_country
21. population_country
22. **latitude_country**
23. **longitude_country**
24. **source**
25. gdp-country
26. etc

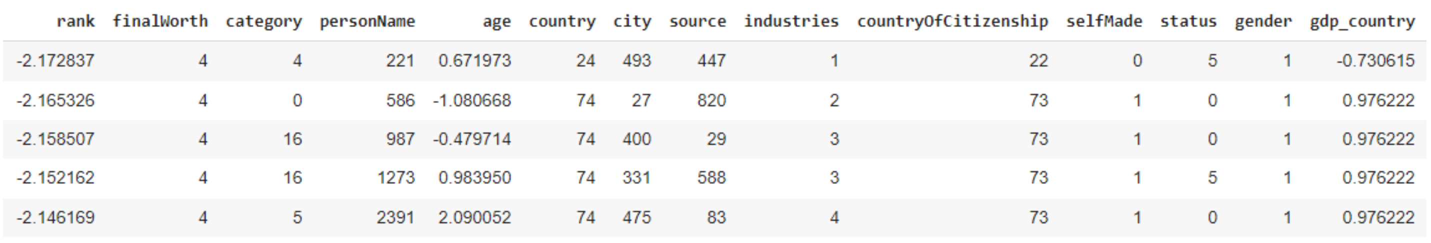
Preprocessing on dataset:

As data set contain missing values and categorical features. We need to transform and clean the data for smooth and fair training of model. For this purpose, we identify the missing value and remove them from our original data set. After handling missing values, we delete the column which are not important for our analysis.



```
#now delete unimportant columns
data=data.drop(columns={'title','birthDate','lastName','firstName','date','residenceStateRegion','state','birthYear','birthMonth','birthDay','cpi_country',
                        'cpi_change_country','gross_tertiary_education_enrollment','gross_primary_education_enrollment_country','population_country'},axis=1)
```

Now apply label encoding for categorical values:



rank	finalWorth	category	personName	age	country	city	source	industries	countryOfCitizenship	selfMade	status	gender	gdp_country
-2.172837	4	4	221	0.671973	24	493	447	1	22	0	5	1	-0.730615
-2.165326	4	0	586	-1.080668	74	27	820	2	73	1	0	1	0.976222
-2.158507	4	16	987	-0.479714	74	400	29	3	73	1	0	1	0.976222
-2.152162	4	16	1273	0.983950	74	331	588	3	73	1	5	1	0.976222
-2.146169	4	5	2391	2.090052	74	475	83	4	73	1	0	1	0.976222

In last step of preprocessing we apply power transformer for remaining columns and transform whole dataset. Now our dataset is ready for further analysis.

Step 1:

QNO 1: Which dataset did you select?

Answer: I choose "Billionaires Data Set 2022" from kaggle. As this data set belongs to finance domain. It shows latest demographics on world billionaires, their Age, total wealth, gender and which country they belong. We can do lots of EDA (Exploratory data Analysis on this data) and find answers of questions such as: World wealth distribution. Which country have youngest Billionaires. Distribution of wealth among gender etc.

QNO2: Which regulated domain does your dataset belong to?

Answer: This data set belongs to finance domain.

Q NO3: How many observations are in the dataset?

Answer: This dataset originally consists of 2640 instance and 35 variables.

Q NO4: Which variables did you select as your dependent variables?

Answer: according to requirement of this task I choose two dependent variables for analysis purpose:

1. FinalWorth
2. Country

QNO 5: How many and which variables in the dataset are associated with a legally recognized protected class?

Answer: There are two variables in dataset associated with legally recognized protected class in USA that are:

- Gender
- Age

QNO6: Which legal precedence/law (as discussed in the lectures) does each protected class fall under?

Answer: **Gender**: This protected class fall under **Civil Rights Acts of 1964.**

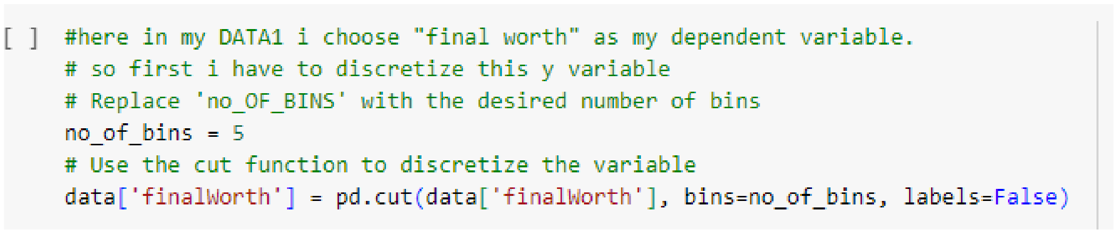
Age: This protected class fall under **Age Discrimination in Employment Act (ADEA)**

Step 2.1. Now we Identify the members associated with our protected class variables and group together into a subset of membership categories.

Table 1. Frequency of Each membership Category of protected class

Age	Self Made	Values	Total
0= “Less then 30”	False True	80 70	157
1= “Grater then 30”	False True	775 1742	2517
Gender	**Self Made**	**Values**	**Total**
0= Female	False True	241 96	345
1= Male	False True	587 1716	2303

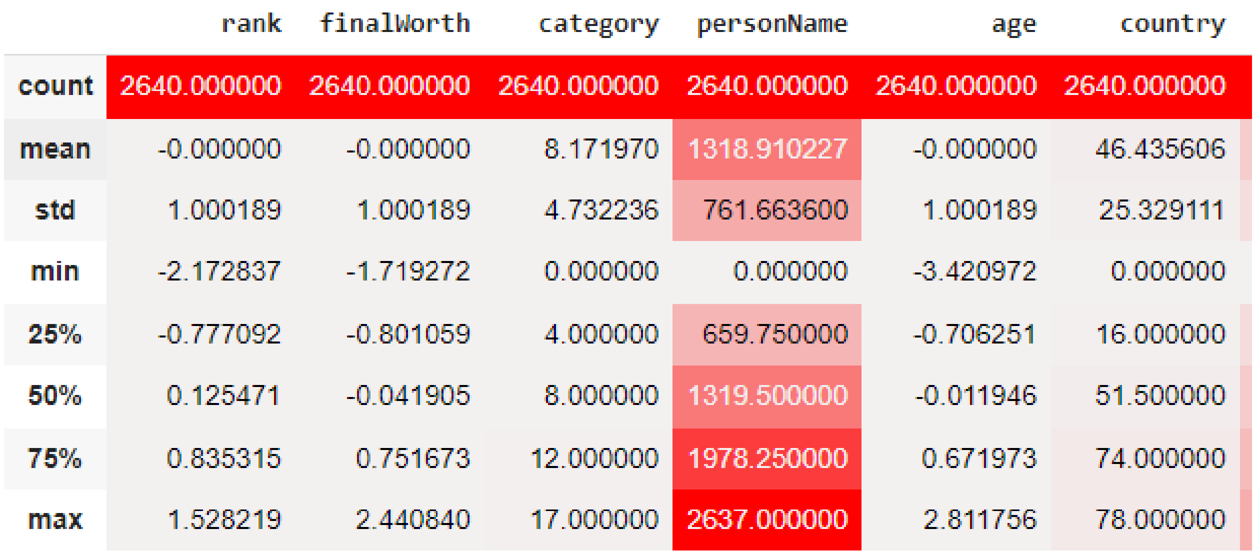
Step 2: In step 2 of this task first we perform discretization on my dependent variable that is “final worth”.



```
[ ]  #here in my DATA1 i choose "final worth" as my dependent variable.
     # so first i have to discretize this y variable
     # Replace 'no_OF_BINS' with the desired number of bins
     no_of_bins = 5
     # Use the cut function to discretize the variable
     data['finalWorth'] = pd.cut(data['finalWorth'], bins=no_of_bins, labels=False)
```

Some descriptive analysis of dataset:

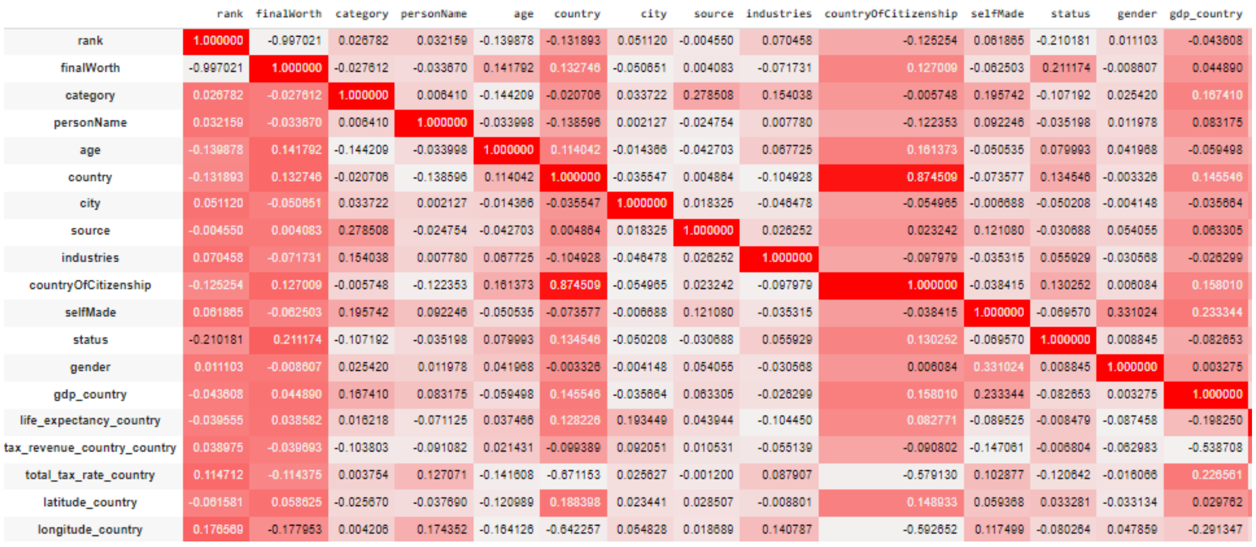
Fig:1. Mean, Standard Deviation, Min values, MAX values



	rank	finalWorth	category	personName	age	country
count	2640.000000	2640.000000	2640.000000	2640.000000	2640.000000	2640.000000
mean	-0.000000	-0.000000	8.171970	1318.910227	-0.000000	46.435606
std	1.000189	1.000189	4.732236	761.663600	1.000189	25.329111
min	-2.172837	-1.719272	0.000000	0.000000	-3.420972	0.000000
25%	-0.777092	-0.801059	4.000000	659.750000	-0.706251	16.000000
50%	0.125471	-0.041905	8.000000	1319.500000	-0.011946	51.500000
75%	0.835315	0.751673	12.000000	1978.250000	0.671973	74.000000
max	1.528219	2.440840	17.000000	2637.000000	2.811756	78.000000

Now we compute the correlation of different variables as shown in fig 2.

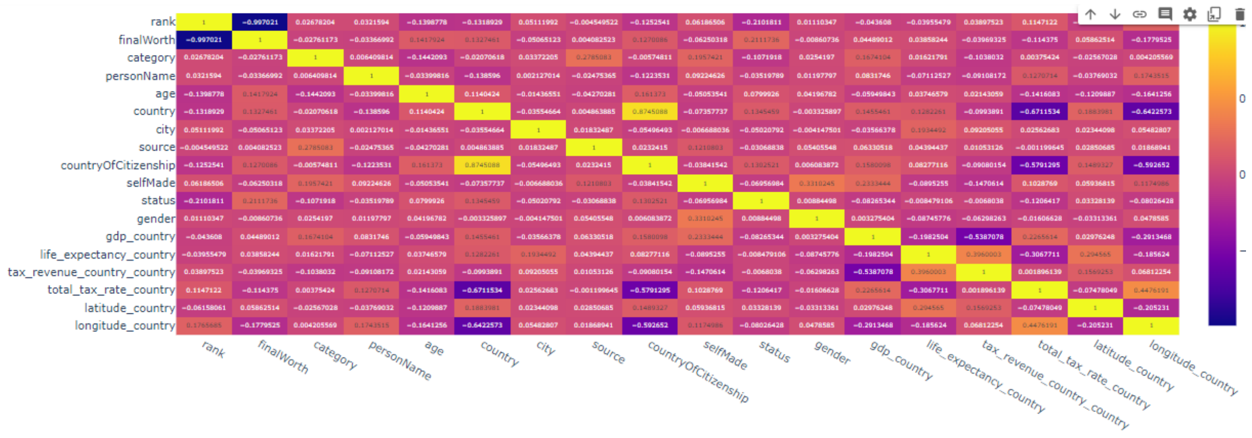
Fig 2. Correlation of Variable with each other



	rank	finalWorth	category	personName	age	country	city	source	industries	countryOfCitizenship	selfMade	status	gender	gdp_country
rank	1.000000	-0.997021	0.026782	0.032159	-0.139878	-0.131893	0.051120	-0.004550	0.070458	-0.125254	0.061865	-0.210181	0.011103	-0.043608
finalWorth	-0.997021	1.000000	-0.027612	-0.033670	0.141792	0.132746	-0.050651	0.004083	-0.071731	0.127009	-0.062503	0.211174	-0.008607	0.044890
category	0.026782	-0.027612	1.000000	0.006410	-0.144209	-0.020706	0.033722	0.278508	0.154038	-0.005748	0.195742	-0.107192	0.025420	0.167410
personName	0.032159	-0.033670	0.006410	1.000000	-0.033998	-0.138596	0.002127	-0.024754	0.007780	-0.122353	0.092246	-0.035198	0.011978	0.083175
age	-0.139878	0.141792	-0.144209	-0.033998	1.000000	0.114042	-0.014366	-0.042703	0.067725	0.161373	-0.050535	0.079993	0.041968	-0.059498
country	-0.131893	0.132746	-0.020706	-0.138596	0.114042	1.000000	-0.035547	0.004864	-0.104928	0.874509	-0.073577	0.134546	-0.003326	0.145546
city	0.051120	-0.050651	0.033722	0.002127	-0.014366	-0.035547	1.000000	0.018325	-0.046478	-0.054965	-0.006688	-0.050208	-0.004148	-0.035664
source	-0.004550	0.004083	0.278508	-0.024754	-0.042703	0.004864	0.018325	1.000000	0.026252	0.023242	0.121080	-0.030688	0.054055	0.063305
industries	0.070458	-0.071731	0.154038	0.007780	0.067725	-0.104928	-0.046478	0.026252	1.000000	-0.097979	-0.035315	0.055929	-0.030568	-0.026299
countryOfCitizenship	-0.125254	0.127009	-0.005748	-0.122353	0.161373	0.874509	-0.054965	0.023242	-0.097979	1.000000	-0.038415	0.130252	0.006084	0.158010
selfMade	0.061865	-0.062503	0.195742	0.092246	-0.050535	-0.073577	-0.006688	0.121080	-0.035315	-0.038415	1.000000	-0.069570	0.331024	0.233344
status	-0.210181	0.211174	-0.107192	-0.035198	0.079993	0.134546	-0.050208	-0.030688	0.055929	0.130252	-0.069570	1.000000	0.008845	-0.082653
gender	0.011103	-0.008607	0.025420	0.011978	0.041968	-0.003326	-0.004148	0.054055	-0.030568	0.006084	0.331024	0.008845	1.000000	0.003275
gdp_country	-0.043608	0.044890	0.167410	0.083175	-0.059498	0.145546	-0.035664	0.063305	-0.026299	0.158010	0.233344	-0.082653	0.003275	1.000000
life_expectancy_country	-0.039555	0.038582	0.016218	-0.071125	0.037466	0.128226	0.193449	0.043944	-0.104450	0.082771	-0.089525	-0.008479	-0.087458	-0.198250
tax_revenue_country_country	0.038975	-0.039693	-0.103803	-0.091082	0.021431	-0.099389	0.092051	0.010531	-0.055139	-0.090802	-0.147061	-0.006804	-0.062983	-0.538708
total_tax_rate_country	0.114712	-0.114375	0.003754	0.127071	-0.141608	-0.671153	0.025627	-0.001200	0.087907	-0.579130	0.102877	-0.120642	-0.016066	0.226561
latitude_country	-0.061581	0.058625	-0.025670	-0.037690	-0.120989	0.188398	0.023441	0.028507	-0.008801	0.148933	0.059368	0.033281	-0.033134	0.029762
longitude_country	0.176569	-0.177953	0.004206	0.174352	-0.164126	-0.642257	0.054828	0.018689	0.140787	-0.592652	0.117499	-0.080264	0.047859	-0.291347

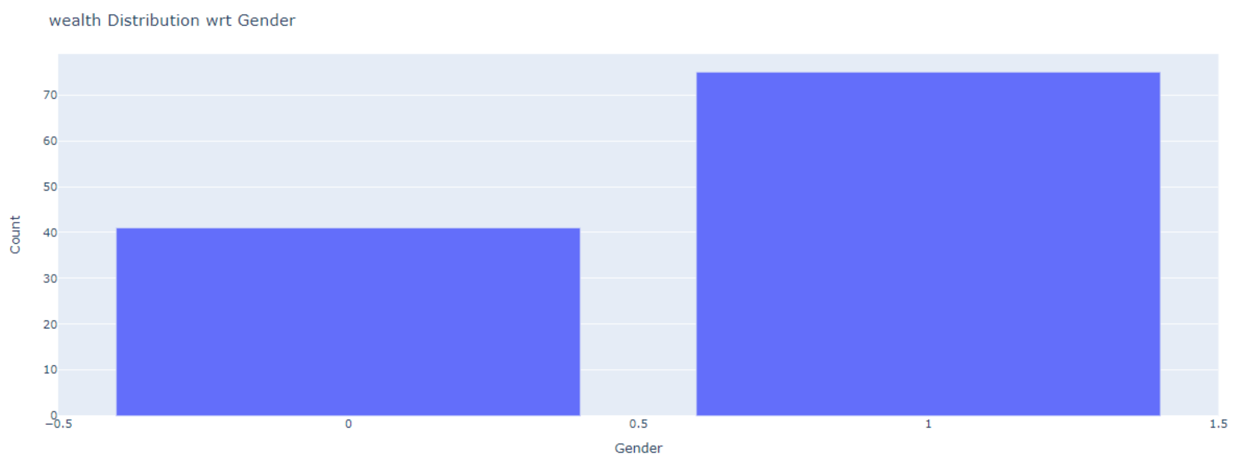
We also used "Heat Map" to see the Relationship of variables. Here are the results

Fig 3



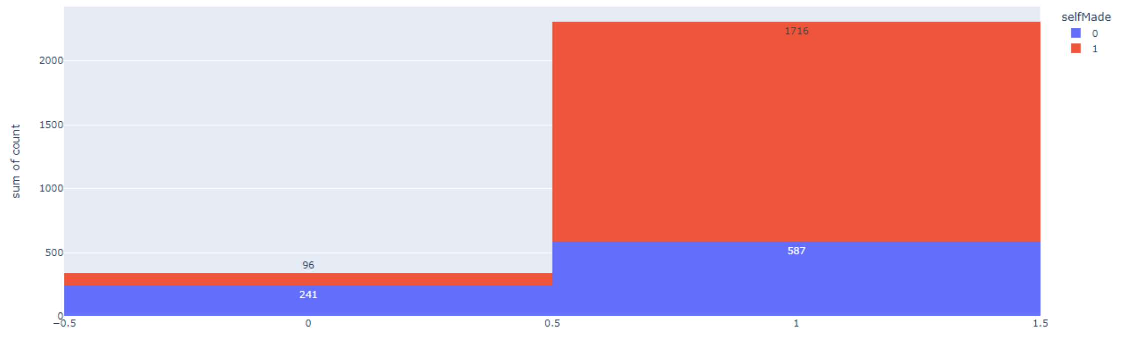
In next step we use Histogram to show the wealth distribution with respect to gender. Here we use our selected Protected variable "Age" to analyze that if there is any Biasness in our data set. As result show "male=1" has higher rates as compare to "female=0" in our data set which is shown in fig 4.

Fig4. Wealth Distribution With respect to Gender



Now we also calculate "Number of Counts" Of our "Protected Class Variable" = "Gender" with respect that are "selfMade". Results shows the numbers of male and female who are self-made persons.

Fig 5: No counts MALE & FEMALE



Similarly, we also try to analyzes the wealth distribution with respect to age. Here the demographics show that “older age people” have more wealth as compare to “youngers”. Here the results show high level of biasness in our dataset. It is shown in fig 5.

FIG5: Wealth Distribution wrt Age

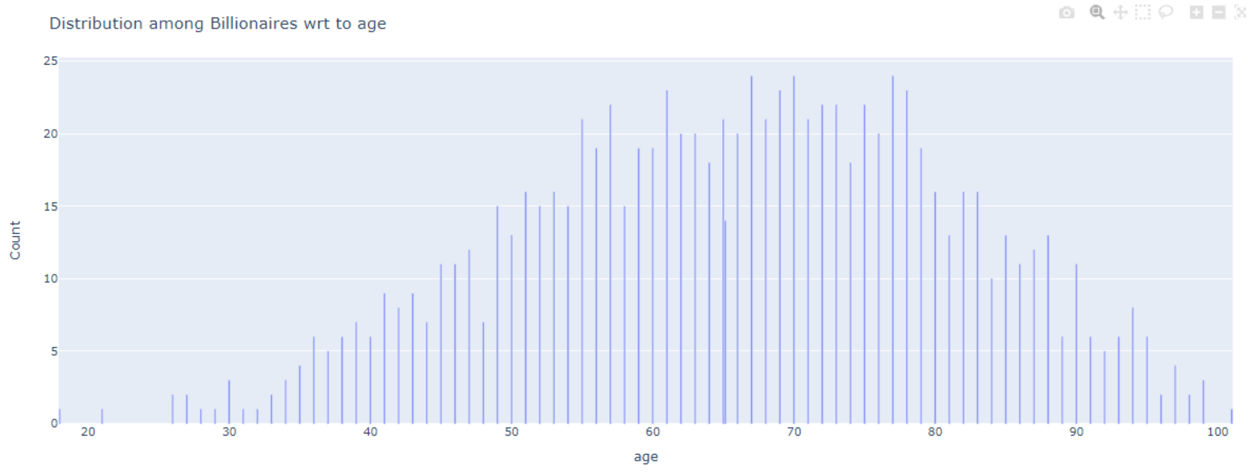
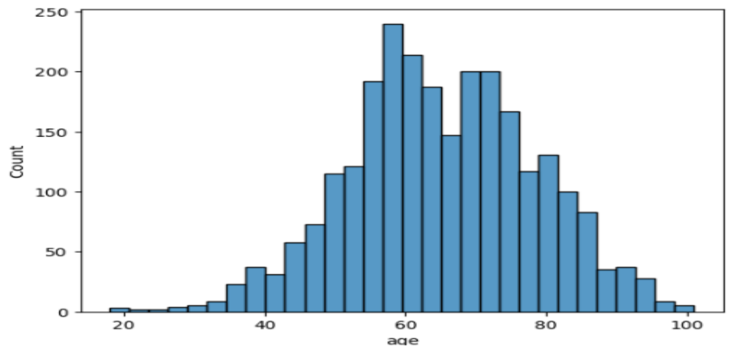
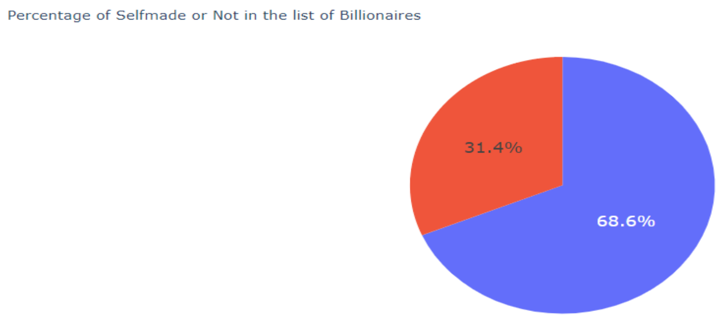


Fig 6: AGE DISTRIBUTION IN DATA SET

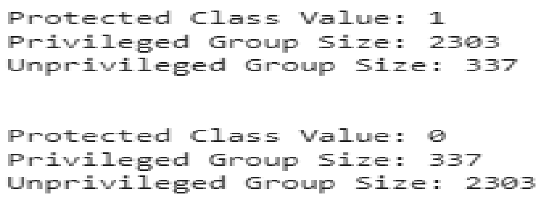


We also compute “percentage of self-made peoples” with respect to other. The results demonstrate that 68.6% peoples are self-made and 31.4 are by birth billionaires.



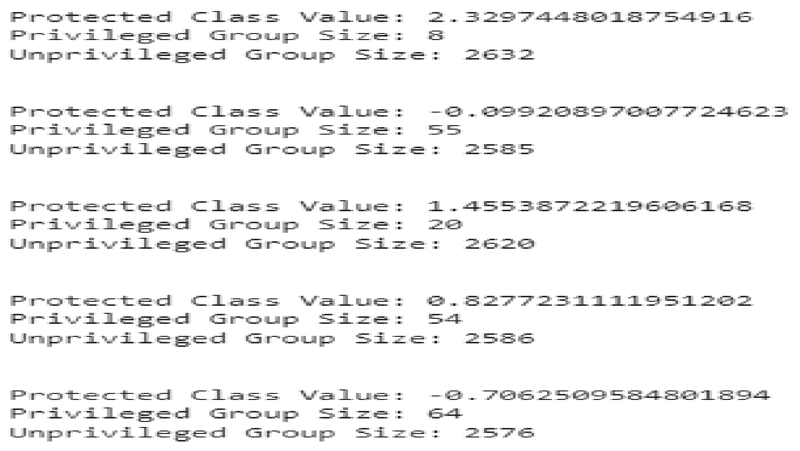
Step 3:

3.1 Based on our dataset, we identify the privileged/unprivileged groups associated with each of your protected class variables. Here “1” indicated “male” and “0” for “female”.



```
Protected Class Value: 1
Privileged Group Size: 2303
Unprivileged Group Size: 337

Protected Class Value: 0
Privileged Group Size: 337
Unprivileged Group Size: 2303
```

We also do this for “age” column:



```
Protected Class Value: 2.3297448018754916
Privileged Group Size: 8
Unprivileged Group Size: 2632

Protected Class Value: -0.0992089700772462З
Privileged Group Size: 55
Unprivileged Group Size: 2585

Protected Class Value: 1.4553872219606168
Privileged Group Size: 20
Unprivileged Group Size: 2620

Protected Class Value: 0.8277231111951202
Privileged Group Size: 54
Unprivileged Group Size: 2586

Protected Class Value: -0.7062509584801894
Privileged Group Size: 64
Unprivileged Group Size: 2576
```

3.2. Compute fairness metrics for both protected classes:

Here we choose two fairness metric for analysis.

1. **Statistical parity difference**
2. **Equal opportunity difference**

1. **Statistical parity difference**
   Statistical Parity Difference(spd) is used to measure fairness and bias assessment in machine learning algorithms. It is also known as Disparate Impact or also called Demographic Parity.

   Formula:

$$SPD = P(\hat{y} = 1 | D = \text{group}_1) - P(\hat{y} = 1 | D = \text{group}_2)$$

Here,

- $P(\hat{y} = 1 | D = \text{group}_1)$ is the probability of a positive prediction ($\hat{y} = 1$) given that the instance belongs to group_1.
- $P(\hat{y} = 1 | D = \text{group}_2)$ is the probability of a positive prediction given that the instance belongs to group_2.

2. **Equal opportunity difference**
   It is a fairness metric used in Machine Learning, when we are deal with classification task and made decisions like individuals belongs to different groups. It is calculated as
   - **Calculate True Positive Rate:**

$$\text{True Positive Rate (Recall)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

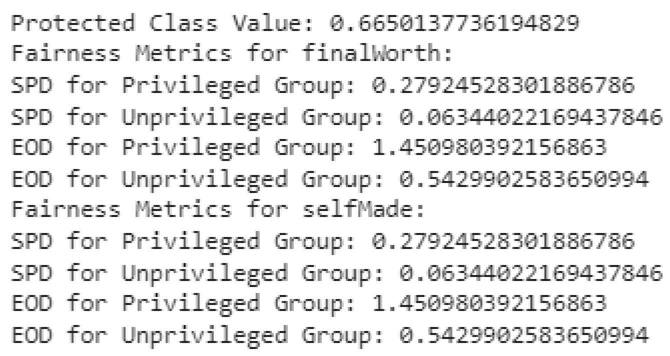
- **Calculate EOD:**
  After calculating TPR we calculate EOD by using following formula.

$$EOD = \text{True Positive Rate (Group 1)} - \text{True Positive Rate (Group 2)}$$

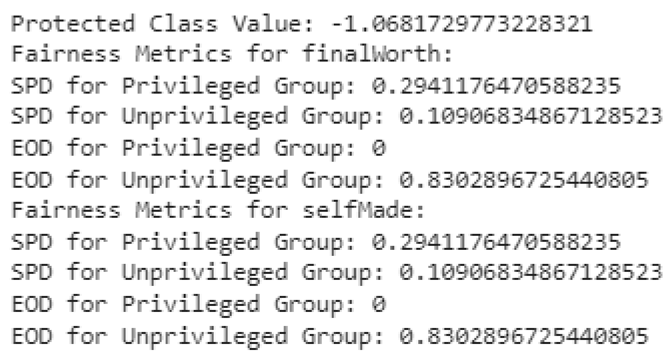
We calculate “Fairness” for our protected classes with respect to our two dependent variables that are “final worth” and “country”. We choose the threshold value 0.5 that we choose randomly.

Here are the results.

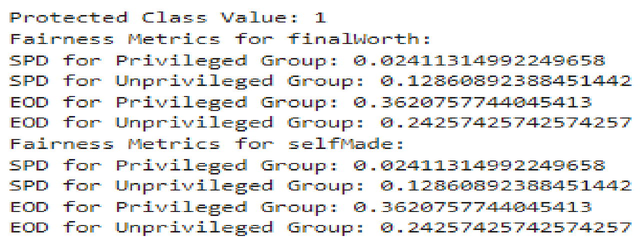
Fairness metrics values protected class “Age”.



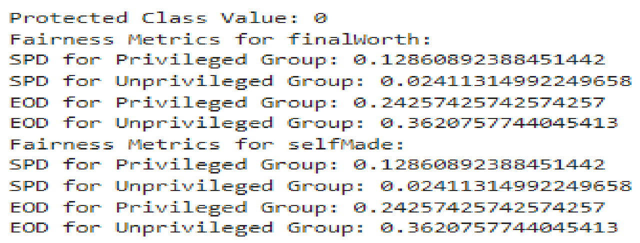
```
Protected Class Value: 0.6650137736194829
Fairness Metrics for finalWorth:
SPD for Privileged Group: 0.27924528301886786
SPD for Unprivileged Group: 0.06344022169437846
EOD for Privileged Group: 1.450980392156863
EOD for Unprivileged Group: 0.5429902583650994
Fairness Metrics for selfMade:
SPD for Privileged Group: 0.27924528301886786
SPD for Unprivileged Group: 0.06344022169437846
EOD for Privileged Group: 1.450980392156863
EOD for Unprivileged Group: 0.5429902583650994
```



```
Protected Class Value: -1.0681729773228321
Fairness Metrics for finalWorth:
SPD for Privileged Group: 0.2941176470588235
SPD for Unprivileged Group: 0.10906834867128523
EOD for Privileged Group: 0
EOD for Unprivileged Group: 0.8302896725440805
Fairness Metrics for selfMade:
SPD for Privileged Group: 0.2941176470588235
SPD for Unprivileged Group: 0.10906834867128523
EOD for Privileged Group: 0
EOD for Unprivileged Group: 0.8302896725440805
```

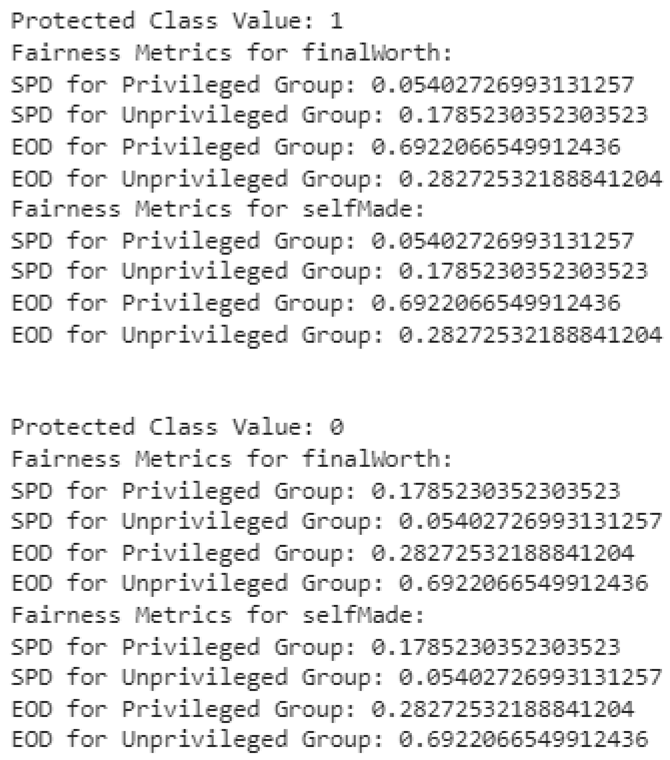


```
Protected Class Value: 1
Fairness Metrics for finalWorth:
SPD for Privileged Group: 0.02411314992249658
SPD for Unprivileged Group: 0.12860892388451442
EOD for Privileged Group: 0.3620757744045413
EOD for Unprivileged Group: 0.24257425742574257
Fairness Metrics for selfMade:
SPD for Privileged Group: 0.02411314992249658
SPD for Unprivileged Group: 0.12860892388451442
EOD for Privileged Group: 0.3620757744045413
EOD for Unprivileged Group: 0.24257425742574257
```



```
Protected Class Value: 0
Fairness Metrics for finalWorth:
SPD for Privileged Group: 0.12860892388451442
SPD for Unprivileged Group: 0.02411314992249658
EOD for Privileged Group: 0.24257425742574257
EOD for Unprivileged Group: 0.3620757744045413
Fairness Metrics for selfMade:
SPD for Privileged Group: 0.12860892388451442
SPD for Unprivileged Group: 0.02411314992249658
EOD for Privileged Group: 0.24257425742574257
EOD for Unprivileged Group: 0.3620757744045413
```

Fairness metrics values protected class “Gender”.



```
Protected Class Value: 1
Fairness Metrics for finalWorth:
SPD for Privileged Group: 0.05402726993131257
SPD for Unprivileged Group: 0.1785230352303523
EOD for Privileged Group: 0.6922066549912436
EOD for Unprivileged Group: 0.28272532188841204
Fairness Metrics for selfMade:
SPD for Privileged Group: 0.05402726993131257
SPD for Unprivileged Group: 0.1785230352303523
EOD for Privileged Group: 0.6922066549912436
EOD for Unprivileged Group: 0.28272532188841204


Protected Class Value: 0
Fairness Metrics for finalWorth:
SPD for Privileged Group: 0.1785230352303523
SPD for Unprivileged Group: 0.05402726993131257
EOD for Privileged Group: 0.28272532188841204
EOD for Unprivileged Group: 0.6922066549912436
Fairness Metrics for selfMade:
SPD for Privileged Group: 0.1785230352303523
SPD for Unprivileged Group: 0.05402726993131257
EOD for Privileged Group: 0.28272532188841204
EOD for Unprivileged Group: 0.6922066549912436
```

TABEL 1: STEP.3

Selected Protected class variables	**Privileged/unprivileged**	**Bias mitigation function**	**Before Fairness metrics**

GENDER	Privileged= Male
	Unprivileged= Female
		Resampling	SPD =0.05
			SPD =0.17
			EOD= 0.20
			EOD=0.69

AGE	Privileged= AGE>30
	Unprivileged= AGE<30
		Resampling	SPD=0.04
			SPD= 0.01
			EOD= 1.40

3.3. Applying Bias Mitigation Algorithm on dataset.

In this step first we randomly split our original data set in training and testing. After that we apply our machine learning algorithm. We used “Random Forest Classifier” as our selected model and we calculate the Mean Difference and Disparate Impact for both “Privileged” and “Unprivileged” groups.

In next step we apply our Bias Mitigation Algorithm on our data and Transform the data. We “AIF360” tool for this task.

3.4. Method used in Bias mitigation algorithm.

There are lots of methods in practice which can be used for bias mitigation. Such as:

1. Pre-processing Techniques
   - Resampling
   - Reweighing
   - Data augmentation
2. Post-processing methods
   - Calibration
   - Threshold adjustment
   - ROC Analysis
3. Fair-Representation methods
4. Bias-Aware Algorithms

For this specific task we use, a Pre-processing approach that is “Reweighing”. The goal is to assign different weights to instances in the training data based on the characteristics of the groups they belong to. This technique is often employed when certain demographic groups are underrepresented or overrepresented in the dataset, leading to biased model predictions.

The basic idea behind re-weighting is to assign higher importance or weight to instances from underrepresented groups, making them more influential during the training process. This way, the model pays more attention to the minority group, reducing the potential bias in its predictions.

The re-weighting process involves modifying the weights assigned to different instances in the training dataset, typically during the training of a machine learning model. The weights are adjusted based on the inverse of the class frequencies or some other metric that reflects the desired distribution.

Here's a simplified example in the context of binary classification:

1. Original Dataset:

- Class 0 (Negative instances): 900 instances

- Class 1 (Positive instances): 100 instances

2. Compute Weights:

- Weight for Class 0:(frac{1}{900}) (approximately)

- Weight for Class 1:(frac{1}{100}) (approximately)

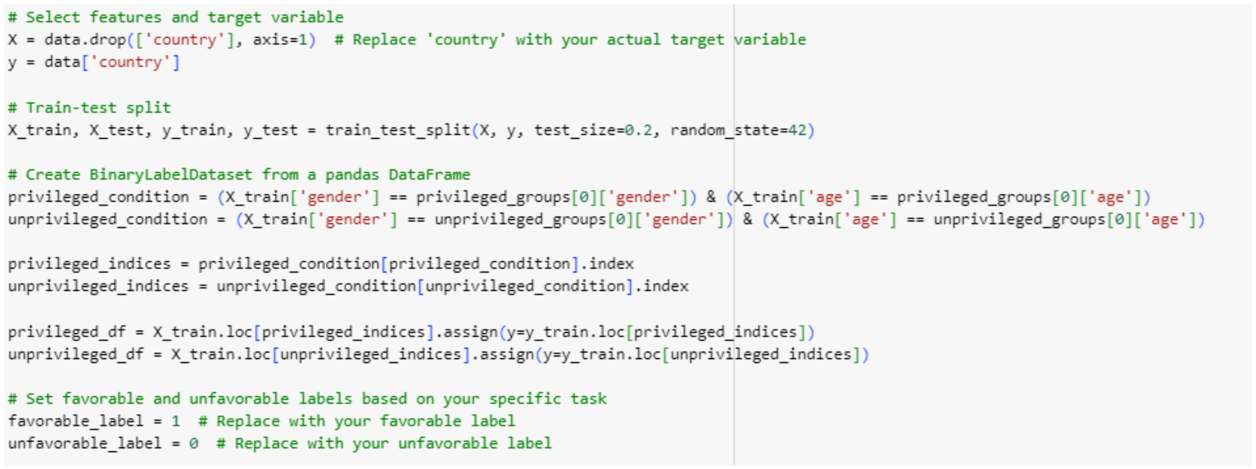
3. **Assign Weights during Training:**

- Instances from Class 0 are assigned a weight of (frac{1}{900}) during model training.
- Instances from Class 1 are assigned a weight of (frac{1}{100}) during model training.

By assigning higher weights to instances from the underrepresented group (Class 1 in this example), the model is encouraged to pay more attention to these instances, potentially mitigating bias and improving fairness.

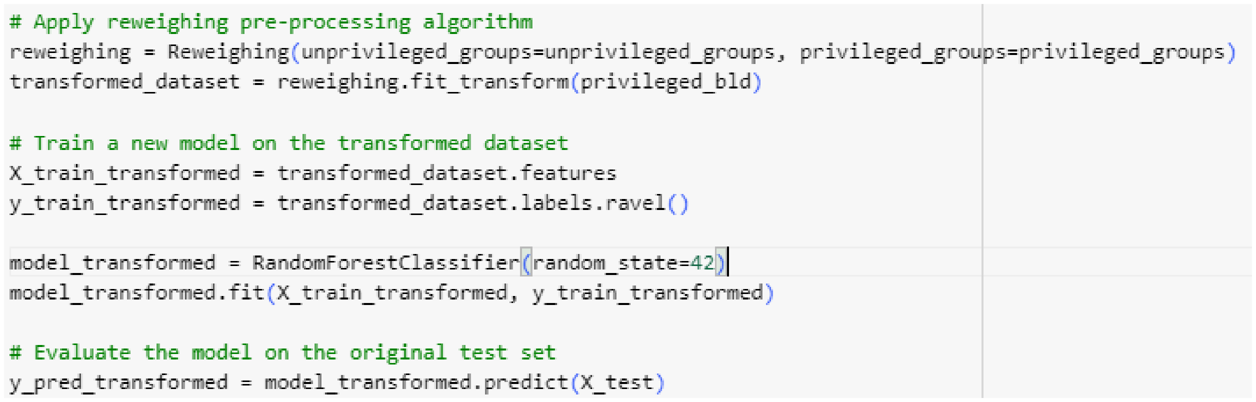
Here are some code snippets:

Fig.3.1



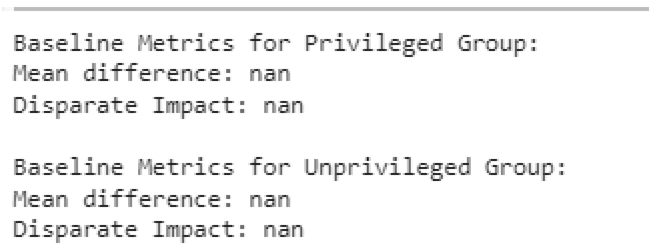
```
# Select features and target variable
X = data.drop(['country'], axis=1)  # Replace 'country' with your actual target variable
y = data['country']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create BinaryLabelDataset from a pandas DataFrame
privileged_condition = (X_train['gender'] == privileged_groups[0]['gender']) & (X_train['age'] == privileged_groups[0]['age'])
unprivileged_condition = (X_train['gender'] == unprivileged_groups[0]['gender']) & (X_train['age'] == unprivileged_groups[0]['age'])

privileged_indices = privileged_condition[privileged_condition].index
unprivileged_indices = unprivileged_condition[unprivileged_condition].index

privileged_df = X_train.loc[privileged_indices].assign(y=y_train.loc[privileged_indices])
unprivileged_df = X_train.loc[unprivileged_indices].assign(y=y_train.loc[unprivileged_indices])

# Set favorable and unfavorable labels based on your specific task
favorable_label = 1  # Replace with your favorable label
unfavorable_label = 0  # Replace with your unfavorable label
```

Fig.3.2.

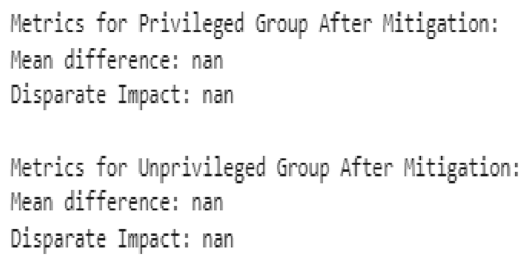


```
# Apply reweighing pre-processing algorithm
reweighing = Reweighing(unprivileged_groups=unprivileged_groups, privileged_groups=privileged_groups)
transformed_dataset = reweighing.fit_transform(privileged_bld)

# Train a new model on the transformed dataset
X_train_transformed = transformed_dataset.features
y_train_transformed = transformed_dataset.labels.ravel()

model_transformed = RandomForestClassifier(random_state=42)
model_transformed.fit(X_train_transformed, y_train_transformed)

# Evaluate the model on the original test set
y_pred_transformed = model_transformed.predict(X_test)
```

Here are the Results after Applying the Bias Mitigation Algorithm.



```
Baseline Metrics for Privileged Group:
Mean difference: nan
Disparate Impact: nan

Baseline Metrics for Unprivileged Group:
Mean difference: nan
Disparate Impact: nan
```



```
Metrics for Privileged Group After Mitigation:
Mean difference: nan
Disparate Impact: nan

Metrics for Unprivileged Group After Mitigation:
Mean difference: nan
Disparate Impact: nan
```

Dependent variabe	**Protected class variable**	**Privileged/unprivileged**	**Bias mitigation function**	**Before Fairness metrics**	**After Fairness metrics/resultin**
FinalWorth	GENDER	Privileged= Male Unprivileged= Female	Resampling	SPD =0.05 SPD =0.17 EOD= 0.20 EOD=O.69	DI = 0
	AGE	Privileged= AGE>30 Unprivileged= AGE<30	Resampling	SPD=0.04 SPD= 0.01 EOD= 1.40	DI = 0
Country	GENDER	Privileged= Male Unprivileged= Female	Resampling	SPD =0.23 SPD =0.22 EOD= 0.01 EOD=O.45	DI = 0
	AGE	Privileged= AGE>30 Unprivileged= AGE<30	Resampling	SPD =0.02 SPD =0.14 EOD= 0.01 EOD=O.07	DI = 0

Q1: Did any of these approaches seem to work to mitigate bias (or increase fairness)? Explain your reasoning.

Answer: Yes, as we apply Resampling on this data set and calculate the disparate impact it gives us “0” value which is consider an ideal value for biasness.

Q2: Did any group receive a positive advantage?

Answer: After applying Bias Mitigation Algorithm “Unprivileged group” receive a positive advantage.

Q3: Was any group disadvantaged by these approaches?

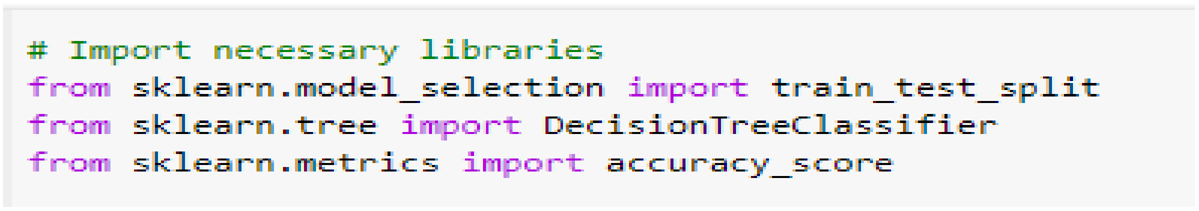
Answer: NO

Step4:

4.1: Model Training on original datset.

For model training I used “Decision Tree Regressor” for model training. We calculate evaluation metric “Root mean square error”. Here is some code snippet:

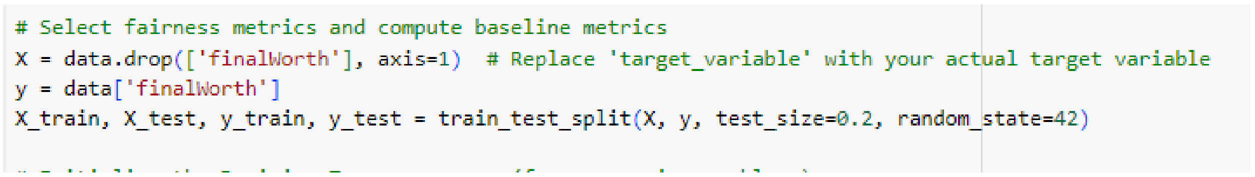
Import basic libraries:



```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

Split the data in training and testing:

Here we select “finalworth” as our Target class and train our model.



```
# Select fairness metrics and compute baseline metrics
X = data.drop(['finalWorth'], axis=1)  # Replace 'target_variable' with your actual target variable
y = data['finalWorth']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fit the model:

```
# Initialize the Decision Tree regressor (for regression problems)
model = DecisionTreeRegressor(random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)
```

Output:

```
Mean Squared Error: 0.00
```

Step 4.1: Model Training on original New unbiased dataset:

We split new “Unbiased datset” in training and testing, then we fit the model on this data set.