Marcus J. Anderson – CS6603

Task 1

- **Public Artifact:**
  - **Title:** "AI Bias: Exploring Discriminatory Algorithmic Decision-Making Models and the Application of Possible Machine-Centric Solutions Adapted from the Pharmaceutical Industry"
  - **Released:** February 10, 2022
  - **Link:** https://link.springer.com/article/10.1007/s43681-022-00138-8#Sec3
- **Application/Scenario/Domain of Misuse:** Criminal Risk Assessment Algorithm
- **Regulated Domain/Protected Classes Impacted:**
  - **Race:** (Civil Rights Act of 1964, 1991)
  - **Sex**: (Equal Pay Act of 1963; Civil Rights Act of 1964, 1991)
    - I only focused on the **RACE** protected class in this report
- **Evidence:**
  - **Dataset:** https://github.com/propublica/compas-analysis

Task 2

The public artifact I chose to write about covers a variety of different discriminatory issues, mainly with race and gender, that AI tends to display in several scenarios. The section of the article that I focused on deals with an algorithm that assigns a risk assessment score from 1 to 10 (1 being the lowest amount of risk, and 10 being the highest) to convicted criminals. This would give criminals a certain profile that judicial systems can use to determine how likely the person will re-offend in a violent or non-violent crime. Those that are assigned higher risk scores are more likely to be to be imprisoned when waiting for their trial, or profiled more strictly, than lower risk individuals.

The issue with this is the data collected showed an unfair bias, or favoritism, within the algorithm rating black individuals a higher risk score than other races (Caucasian, Asian, Native Americans, etc.). This leads to a racial imbalance that could cause the algorithm to rate a black and non-black criminals differently even if they have the same criminal history. This showcase of bias is similar to the *Garbage In, Garbage Out* view example covered in our lectures. This view had an issue with discriminatory decisions, and the only way to fix it would be to "blind the model to protected features". This means that the protected class, **race**, would be taken out of the model, and the supplementary information (criminal history, violent offense rating, etc.) would be what the algorithm solely considers when assigning risk scores.

Task 3

- **Privileged/unprivileged groups:**
  - **Unprivileged:** African American
  - **Privileged:** Other races (Caucasian, Asian, Native Americans, etc.)
- **Any misleading graphs?:**
  - None; Both graphs that are included in the article have balanced scaling and include zero within their y-axes. Both axes in each graph are labelled properly and have an accurate title to show what data is being represented.

- **Sources of Data Bias:**
    - **Measurement Bias:** How we choose, analyze, and measure a specific feature. In this case, the algorithm is taking racial features and gender into account when assigning risk scores to criminals.
    - **Historical Bias:** A type of existing bias in society that is supported by the collection data. In this case, black criminals are more likely to re-offend, and commit more violent crimes, than other races according to the evidence dataset. This has been a stereotype in the black community for as long as I can remember.
- **Sources of Sampling Bias:**
    - **Selection Bias:** Bias introduced by the selection of data analysis in such a way that proper randomization is not achieved. In this case, to me, the data collected from criminals of different races were not randomized in the graphs shown within the public artifact.
- **Sampling Methods Used to Collect Data:**
    - **Clustered Sampling:** Splitting the population into similar parts or clusters. In this case, the risk assessment scores are separated by clusters of racial groups.
- **Correlations found in the data:**
    - **Decile Score – African American:** -0.6848588 → "strong" correlation
    - **Decile Score – Other Races:** -0.868270329→ "very strong" correlation
- **Outcome measures:**
    - **Averages:**
        - **African American:** 317.50
        - **Other Races:** 299.70
    - **Standard Deviations:**
        - **African American:** 37.96562773
        - **Other Races:** 252.1833768
    - **Quartiles (Data in Order):**
        - **African American:**
            - **Q1:** 301
            - **Q2:** 320.5
            - **Q3:** 343
        - **Other Races:** 252.1833768
            - **Q1:** 119
            - **Q2:** 235
            - **Q3:** 349
    - **Frequency Distributions:**

Race	Frequencies (Decile Scores)
African American	**1:** 365 **2:** 346 **3:** 298 **4:** 337 **5:** 323 **6:** 318 **7:** 343 **8:** 301 **9:** 317 **10:** 227
Other Races	**1:** 921 **2:** 476 **3:** 349 **4:** 329 **5:** 259 **6:** 211 **7:** 153 **8:** 119 **9:** 103 **10:** 77

- **Margins of Error (95% Confidence Level):**
  - **African American:** 156.3048763
  - **Other Races:** 23.53133986
- **Bias and Fairness Metrics (Favorable Outcome: decile_score <= 4):**
  - **Disparate Impact:** 0.614066
  - **Statistical Parity Difference:** -0.268024

Task 4

The issue that I wanted to mitigate in the dataset were the fairness metric results calculated in the previous task. To help mitigate the bias, I created an algorithm (shown below) that took five other variables into account to calculate a fairer risk score to each criminal. The inputs used were: **juv_fel_count, juv_misd_count, juv_other_count, priors_count, v_decile_score**, and the output is the newly calculated **decile_score (risk score)**.

My idea was to assign each method its own threshold value to return a numeric weight based on the input value, which would then be added together to calculate the new decile_score. The point of this algorithm is to provide a fairer risk score by including other factors, and excluding the race or sex of the individual. This would take previously higher risk scores (8-10), usually assigned to black criminals, and potentially rate them at a more medium (5-7) or low score (1-4), increasing the chance of more favorable outcomes (<=4) from the unprivileged group, African Americans.

For example, I tested a few inputs and it lowered their original decile_score by as much as 4 or 5 points based on the five input values in the evidence dataset.

Code:

Note: This code could be heavily refactored, but I wanted to provide a basic visual on how I intended the algorithm to work.

```
class Criminal:
    def __init__(self, juv_fel_count: int, juv_misd_count: int, juv_other_count:
int, priors_count: int, v_decile_score: int):
        self.juv_fel_count = juv_fel_count
        self.juv_misd_count = juv_misd_count
        self.juv_other_count = juv_other_count
        self.priors_count = priors_count
        self.v_decile_score = v_decile_score

# The weight rating from a criminal's juv_fel_count are as follows:
# If the input equals 0, the weight rating will be 0
# If the input is 1 > juv_fel_count <= 6, the weight rating will be 0.5
# Any input over 6 will return a rating of 1
def juv_fel_count_weight(jfc: int) -> int:
    if (jfc == 0):
        return 0
    elif (jfc > 1 and jfc <= 6):
        return 0.5
```

```
    else:
        return 1

# The weight rating from a criminal's juv_misd_count are as follows:
# If the input equals 0, the weight rating will be 0
# If the input is 1 > juv_misd_count <= 6, the weight rating will be 0.5
# Any input over 6 will return a rating of 1
def juv_misd_count_weight(jmc: int) -> int:
    if (jmc == 0):
        return 0
    elif (jmc > 1 and jmc <= 6):
        return 0.5
    else:
        return 1

# The weight rating from a criminal's juv_other_count are as follows:
# If the input equals 0, the weight rating will be 0
# If the input is 1 > juv_other_count <= 6, the weight rating will be 0.5
# Any input over 6 will return a rating of 1
def juv_other_count_weight(joc: int) -> int:
    if (joc == 0):
        return 0
    elif (joc > 1 and joc <= 6):
        return 0.5
    else:
        return 1

# The weight rating from a criminal's priors_count are as follows:
# If the input equals 0, the weight rating will be 0
# If the input is 1 > priors_count <= 12, the weight rating will be 0.25
# If the input is 12 > priors_count <= 24, the weight rating will be 0.5
# Any input over 24 will return a rating of 1
def priors_count_weight(pc: int) -> int:
    if (pc == 0):
        return 0
    elif (pc > 1 and pc <= 12):
        return 0.25
    elif(pc > 12 and pc <= 24):
        return 0.5
    else:
        return 1

# Because the violent score is a big part of the risk score, the weight values
are as follows:
# If the input is 1 or 2, the weight rating will be 1
```

```
# If the input is 3 or 4, the weight rating will be 2
# If the input is 5 or 6, the weight rating will be 3
# If the input is 7 or 8, the weight rating will be 4
# If the input is 9 or 10, the weight rating will be 5
def v_decile_score(vdc: int) -> int:
    if (vdc == 1 or vdc == 2):
        return 1
    elif(vdc == 3 or vdc == 4):
        return 2
    elif(vdc == 5 or vdc == 6):
        return 3
    elif(vdc == 7 or vdc == 8):
        return 4
    else:
        return 5

# This where the new decile_score value will be stored
# Based on the sum of the five methods that return a weight value, the new, and
more fairly assessed risk value will be assigned to the criminal
decile_score = 0

# This object holds the values assigned to the respective columns in the raw data
# These will be used as inputs for the five methods that return a respective
weight value
criminal = Criminal(0,12,2,28,5)

decile_score += juv_fel_count_weight(criminal.juv_fel_count)
decile_score += juv_misd_count_weight(criminal.juv_misd_count)
decile_score += juv_other_count_weight(criminal.juv_other_count)
decile_score += priors_count_weight(criminal.priors_count)
decile_score += v_decile_score(criminal.v_decile_score)

# Final decile_score based on the sum of all calculated weights
print(decile_score)
```