Lab No 9

Utkarsh Bhangale
2020802124
Data Science

Aim - Program to fix AI Biasness in word embedding.

Software Required: Jupyter Notebook, IDLE, Any Python Editor

Prerequisite: Familiarity with bias & fairness concepts, pandas,Python syntax, ML datasets/workflow, basic debiasing

Theory/Concept:

Fairness in Machine Learning: The goal of the code is to build a fair machine learning model for hiring recommendations. It applies concepts of algorithmic fairness like disparate impact, equalized odds, calibration etc.

Bias Measurement: It measures bias in the initial dataset/model using fairness metrics like disparate impact. This helps quantify biases based on gender and race.

Debiasing Techniques: After identifying biases, it applies debiasing techniques like reweighting to mitigate them. Reweighting modifies the training data distribution to achieve demographic parity.

Synthetic Dataset Generation: A synthetic dataset is created programmatically to contain resume and job attributes. This allows experimenting with debiasing algorithms without real private data.

Pandas for Data Wrangling: Pandas DataFrame is used to efficiently manipulate, aggregate and filter the dataset for bias calculations.

Conditional Logic in Python: If/else statements filter rows based on conditions to calculate metrics separately for different demographic groups.

Machine Learning Workflow: It follows a standard ML process - data collection, preprocessing, bias evaluation,model development with fairness enhancements.

Guideline for Fair ML: The code provides a simple example implementation of fair machine learning guidelines around bias identification, quantification and mitigation.

Algorithm/Pseudo code:

Data Generation:
Define features/attributes like gender, race, skills etc
Initialize empty dictionary/list to store values
Generate random values for each attribute
Populate dictionary with feature-value pairs
Convert dictionary to DataFrame

Bias Measurement:
Filter rows by gender and check if hired = Yes
Count males and females matching this criteria
Calculate ratio of hired males to total males Similarly for females

Calculate disparate impact as ratio of ratios
Repeat above steps for race categories

Reweighting: Calculate total counts for each gender/race
Calculate weights as ratio of underrepresented to overrepresented group counts
Initialize weight column in DataFrame
Filter rows by gender/race and assign calculated weight

Repeat steps 2-3 on reweighted data to validate mitigation of bias

Pseudo-code:

GenerateData()
FilterRows(column1, condition1)
CountMatches(column1, condition1, condition2)
CalculateRatio(num, den)
CalculateDisparateImpact(ratio1, ratio2)
CalculateWeights(count1, count2)
AssignWeights(column1, weight)
ValidateReducedBias()

Relative Applications:

Hiring/Recruitment - Building a fair and unbiased resume screening/candidate recommendation system.

College Admissions - Creating equitable and inclusive admissions processes using ML while avoiding discrimination.

Credit Scoring - Developing explainable and bias-free credit risk models that don't negatively impact protected groups.

Healthcare - Applying fair algorithms for diagnoses, treatment recommendations, insurance approvals etc.

Policing - Implementing accountability and transparency into predictive policing systems through fairness techniques.

Transportation - Ensuring transportation infrastructure and mobility planning doesn't burden disadvantaged communities.

Education - Building personalized learning and student support systems that work for all demographic groups.

```
import random
import pandas as pd
# Create a synthetic dataset
data = {
    "resume_id": range(1, 31),
    "gender": [random.choice(["Male", "Female"]) for _ in range(30)],
    "race": [random.choice(["White", "Black", "Asian"]) for _ in range(30)],
    "education": [random.choice(["Bachelor's", "Master's", "Ph.D."]) for _ in
    "experience": [random.randint(1, 10) for _ in range(30)],
    "skills": [random.choice(["Java", "Python", "C++"]) for _ in range(30)],
    "hired": [random.choice(["Yes", "No"]) for _ in range(30)]
}

# Create a DataFrame
df = pd.DataFrame(data)
```

```
# Save the DataFrame as a CSV file
df.to_csv("hiring_recommendation_dataset.csv", index=False)
display(df)
```

	resume_id	gender	race	education	experience	skills	hired
0	1	Female	White	Master's	1	Python	No
1	2	Female	White	Ph.D.	2	Java	Yes
2	3	Female	Asian	Bachelor's	8	Java	No
3	4	Female	White	Master's	10	Python	Yes
4	5	Male	Black	Bachelor's	3	Java	No
5	6	Male	White	Ph.D.	10	Python	Yes
6	7	Female	Black	Ph.D.	2	C++	Yes
7	8	Male	Black	Bachelor's	10	Java	No
8	9	Male	Asian	Master's	1	Java	No
9	10	Male	Asian	Ph.D.	1	Python	Yes
10	11	Female	White	Master's	3	Java	Yes
11	12	Female	Black	Bachelor's	8	C++	No
12	13	Female	White	Ph.D.	6	C++	No
13	14	Male	Black	Ph.D.	1	C++	No
14	15	Male	Asian	Bachelor's	3	C++	No
15	16	Female	Asian	Bachelor's	1	Java	No
16	17	Female	Asian	Master's	5	C++	Yes
17	18	Female	Black	Bachelor's	5	C++	Yes
18	19	Male	Black	Bachelor's	4	C++	No
19	20	Male	White	Bachelor's	10	Java	Yes
20	21	Female	White	Ph.D.	9	C++	Yes
21	22	Male	White	Master's	10	Python	No
22	23	Male	Asian	Ph.D.	5	Python	Yes
23	24	Male	Asian	Ph.D.	9	Python	Yes
24	25	Male	Asian	Master's	9	Python	Yes
25	26	Male	Black	Master's	6	C++	No
26	27	Female	Black	Master's	2	Python	No
27	28	Female	Black	Master's	8	Python	No
28	29	Male	Black	Bachelor's	5	Java	No
29	30	Female	White	Master's	10	Python	No

```
import pandas as pd
import random

# Load the dataset
df = pd.read_csv("hiring_recommendation_dataset.csv")
```

```
# Calculate the number of male and female candidates who were hired.
male_hired = df[(df["gender"] == "Male") & (df["hired"] == "Yes")].shape[0]
female_hired = df[(df["gender"] == "Female") & (df["hired"] == "Yes")].shape[0

# Calculate the number of White, Black, and Asian candidates who were hired.
white_hired = df[(df["race"] == "White") & (df["hired"] == "Yes")].shape[0]
black_hired = df[(df["race"] == "Black") & (df["hired"] == "Yes")].shape[0]
asian_hired = df[(df["race"] == "Asian") & (df["hired"] == "Yes")].shape[0]

# Calculate the total number of male, female, White, Black, and Asian candidat
total_male = df[df["gender"] == "Male"].shape[0]
total_female = df[df["gender"] == "Female"].shape[0]
total_white = df[df["race"] == "White"].shape[0]
total_black = df[df["race"] == "Black"].shape[0]
total_asian = df[df["race"] == "Asian"].shape[0]

# Calculate disparate impact for gender and race.
disparate_impact_gender = female_hired / total_female / (male_hired / total_ma
disparate_impact_race_white = white_hired / total_white / (black_hired / total
disparate_impact_race_black = black_hired / total_black / (white_hired / total
disparate_impact_race_asian = asian_hired / total_asian / (white_hired / total

print(f"Disparate Impact (Gender): {disparate_impact_gender}")
print(f"Disparate Impact (Race - White): {disparate_impact_race_white}")
print(f"Disparate Impact (Race - Black): {disparate_impact_race_black}")
print(f"Disparate Impact (Race - Asian): {disparate_impact_race_asian}")

# Calculate reweighting factors for gender and race to ensure equal representa
weight_gender = total_male / (2 * total_male)
weight_race_white = total_black / (2 * total_black)
weight_race_black = total_white / (2 * total_white)
weight_race_asian = total_white / (2 * total_white)

# Apply reweighting to the dataset
df["weight"] = 1.0  # Initialize weights
df.loc[df["gender"] == "Male", "weight"] = weight_gender
df.loc[df["race"] == "White", "weight"] = weight_race_white
df.loc[df["race"] == "Black", "weight"] = weight_race_black
df.loc[df["race"] == "Asian", "weight"] = weight_race_asian
```

```
Disparate Impact (Gender): 0.8888888888888888
Disparate Impact (Race - White): 1.875
Disparate Impact (Race - Black): 0.5333333333333333
Disparate Impact (Race - Asian): 0.9846153846153847
```