Lab No 10

Utkarsh Bhangale
2020802124
Data Science

Aim - Program to solve AI Fairness issue.

Software Required: Jupyter Notebook, IDLE, Any Python Editor

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

Theory/concept:
Algorithmic Fairness: The goal is to develop models that are fair and avoid discrimination or disparate treatment/impact based on sensitive attributes like gender, race etc.

Demographic Parity: Used as a fairness metric to measure unfair differences in predictive outcomes across demographic groups. Ensures equal true positive/negative rates.

Fairness Constraints: Algorithms like Exponentiated Gradient optimize models to directly satisfy fairness constraints like demographic parity during training.

Disparate Impact/Treatment: Fairness metrics quantify initial unfairness in the model to check for differences in predictions or decisions that disadvantage protected groups.

Mitigation Techniques: Debiasing algorithms aim to modify model training to reduce unfairness while maintaining high accuracy. This tries to resolve unfairness problems.

Potential Bias in Data: Synthetic data is used to control for biases in real data collection/labeling processes and isolate effects of modeling choices on fairness.

Fair Machine Learning workflow: Stages involve defining metrics, auditing initial unfairness, using specialized techniques during training, and re-checking fairness post debiasing.

Explainable/Transparent Systems: Goals like interpretability and accountability promote developing fair models through techniques users and auditors can understand.

Ongoing Monitoring: Fairness is not a single evaluation but requires regular re-assessment over time as data and environments change.

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 -

Credit scoring/lending - Developing unbiased risk assessment models for loan approvals.

Criminal justice - Implementing fair algorithms in areas like risk assessment, bail/sentencing.

Employment/hiring - Building equitable recruitment/hiring systems that avoid discrimination.

Education - Creating inclusive personalized learning and student support tools.

Healthcare - Applying fair ML to disease risk prediction, treatment recommendations, insurance approvals etc.

Smart cities - Ensuring algorithmic decision systems for infrastructure/services don't negatively impact disadvantaged groups.

Autonomous vehicles - Developing self-driving car AI that considers safety and fairness for all road users.

Customer service - Building chatbots, support systems etc that are respectful and responsive to all customers regardless of personal attributes.

Marketing/advertising - Developing fair targeted advertising models that don't enable digital redlining or hidden biases.

Government services - Implementing fairness best practices in computational processes that directly impact citizens like welfare programs, public transportation etc.

Policing - Helping develop explainable, transparent and equitable tools to support public safety operations.

```
!pip install aif360
```

```
Collecting aif360
  Downloading aif360-0.5.0-py3-none-any.whl (214 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 214.1/214.1 kB 4.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.10/dist-packages (from aif360) (1.23.5)
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from aif360) (1.11.3)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.10/dist-packages (from aif360) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0 in /usr/local/lib/python3.10/dist-packages (from aif360) (1.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from aif360) (3.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.0->aif360) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.0->aif360) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0->aif360) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0->aif360) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->aif360) (3.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.24.0->aif360) (1.16.0)
Installing collected packages: aif360
Successfully installed aif360-0.5.0
```

```
import pandas as pd
import numpy as np

# Set a random seed for reproducibility
np.random.seed(0)

# Generate synthetic data for the job application dataset
n_samples = 1000

# Generate features
experience = np.random.randint(0, 20, n_samples)  # Years of experience
education_level = np.random.randint(1, 5, n_samples)  # Education level (1-4)

# Define the gender attribute (0: Female, 1: Male)
gender = np.random.choice([0, 1], n_samples, p=[0.4, 0.6])

# Simulate the interview invitation status (0: Not invited, 1: Invited)
interview_invitation = np.random.choice([0, 1], n_samples, p=[0.7, 0.3])

# Create a Pandas DataFrame
data = pd.DataFrame({
    'Experience': experience,
    'Education_Level': education_level,
    'Gender': gender,
    'Interview_Invitation': interview_invitation
})

# Save the synthetic dataset to a CSV file
data.to_csv('job_applications_data.csv', index=False)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from aif360.datasets import BinaryLabelDataset
```

```python
from aif360.datasets import BinaryLabelDataset
from aif360.metrics import BinaryLabelDatasetMetric
from aif360.algorithms.preprocessing import Reweighing
import warnings
warnings.simplefilter('ignore', FutureWarning)

data = pd.DataFrame({
    'Experience': experience,
    'Education_Level': education_level,
    'Gender': gender,
    'Interview_Invitation': interview_invitation
})

# Identify the sensitive attribute (Gender) and the target variable.
sensitive_attr = 'Gender'
target_attr = 'Interview_Invitation'

# Split the data into features (X) and the target variable (y).
X = data.drop(target_attr, axis=1)
y = data[target_attr]

# Split the data into training and test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train a baseline model without considering fairness (e.g., logistic regression).
baseline_model = LogisticRegression()
baseline_model.fit(X_train, y_train)

# Make predictions using the baseline model.
y_pred_baseline = baseline_model.predict(X_test)

# Define the protected attribute (Gender) and privileged group (e.g., Male).
protected_attr = 'Gender'
privileged_group = [{'Gender': 1}]  # Replace with the privileged group definition in your data

# Create a BinaryLabelDataset to perform fairness assessments.
dataset = BinaryLabelDataset(
    favorable_label=1,
    unfavorable_label=0,
    df=data,
    label_names=['Interview_Invitation'],
    protected_attribute_names=[protected_attr],
)

# Implement reweighing to mitigate bias.
RW = Reweighing(unprivileged_groups=[{protected_attr: 0}], privileged_groups=[{protected_attr: 1}])
reweighted_dataset = RW.fit_transform(dataset)

# Train a new hiring model using the reweighed data.
fair_model = LogisticRegression()
fair_model.fit(reweighted_dataset.features, reweighted_dataset.labels.ravel())

# Make predictions using the fair model.
y_pred_fair = fair_model.predict(X_test)

# Evaluate fairness and performance metrics.
metric_fair = BinaryLabelDatasetMetric(reweighted_dataset, unprivileged_groups=[{protected_attr: 0}],
                                       privileged_groups=[{protected_attr: 1}])

disparate_impact_fair = metric_fair.disparate_impact()

statistical_parity_difference_fair = metric_fair.statistical_parity_difference()

accuracy = accuracy_score(y_test, y_pred_fair)
precision = precision_score(y_test, y_pred_fair)
recall = recall_score(y_test, y_pred_fair)
f1 = f1_score(y_test, y_pred_fair)

# Print fairness and performance metrics.
print("Disparate Impact (Fair Model):", disparate_impact_fair)

print("Statistical Parity Difference (Fair Model):", statistical_parity_difference_fair)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Disparate Impact (Fair Model): 1.0000000000000004
Statistical Parity Difference (Fair Model): 1.6653345369377348e-16
Accuracy: 0.725
Precision: 0.0
Recall: 0.0
F1 Score: 0.0
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LogisticRegression was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to
  _warn_prf(average, modifier, msg_start, len(result))
```