Experiment 2: Impact of Data Quality on AI Fairness

Objective: To understand how imbalanced data affects the fairness and performance of AI models — and how data balancing techniques (like SMOTE) can improve fairness.

```
!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/dist-packages (from fairlearn) (2.0.2)
Requirement already satisfied: pandas>=2.0.3 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (2.2.2)
Requirement already satisfied: scikit-learn>=1.2.1 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (1.6.1)
Requirement already satisfied: scipy>=1.9.3 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (1.16.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.3->fairlearn) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.3->fairlearn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.3->fairlearn) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.2.1->fairlearn) (1.5.1
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.2.1->fairlearn
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=2.0.3->fair
Downloading fairlearn-0.12.0-py3-none-any.whl (240 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240.0/240.0 kB 13.0 MB/s eta 0:00:00
Installing collected packages: fairlearn
Successfully installed fairlearn-0.12.0
```

Dataset - Women-Centric Bias Tweet Dataset (2K Tweets)

```
Negative bias → Tweets containing harmful stereotypes or discrimination.
Neutral bias → Tweets that are factual or don’t carry a clear positive/negative slant.
Positive bias → Tweets praising, supporting, or showing women in a good light.
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from fairlearn.metrics import MetricFrame, true_positive_rate, false_positive_rate


# Load dataset
df = pd.read_csv('/content/Biased_tweets_2k.csv')

# Keep relevant columns & drop missing
df = df[['cleaned_text', 'weak_label']].dropna().drop_duplicates()

# Create sensitive feature
df['gender_ref'] = df['cleaned_text'].apply(
    lambda x: 'female' if any(word in x.lower() for word in ['woman', 'girl', 'she', 'her'])
    else 'male'
)


# Encode labels
df['label'] = df['weak_label'].map({'positive': 1, 'negative': 0})

# Drop rows with NaN values in 'label' or 'gender_ref' before splitting
df.dropna(subset=['label', 'gender_ref'], inplace=True)

# Stratified split by label to avoid imbalance
X_train, X_test, y_train, y_test, gender_train, gender_test = train_test_split(
    df['cleaned_text'], df['label'], df['gender_ref'],
    test_size=0.3, random_state=42, stratify=df[['label', 'gender_ref']]
)
```

```
# Limit vectorizer to avoid overfitting
vectorizer = CountVectorizer(max_features=500)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train logistic regression
model = LogisticRegression(max_iter=500)
model.fit(X_train_vec, y_train)
y_pred = model.predict(X_test_vec)
```

```
# Accuracy
print("Model Accuracy:", accuracy_score(y_test, y_pred))

# Fairness metrics
metric_frame = MetricFrame(
    metrics={'TPR': true_positive_rate, 'FPR': false_positive_rate},
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=gender_test
)

print("\nFairness metrics by gender (Baseline):")
print(metric_frame.by_group)
```

```
Model Accuracy: 0.6666666666666666

Fairness metrics by gender (Baseline):
            TPR  FPR
gender_ref
female      0.5  0.5
male        1.0  0.0
```

Model Accuracy = 66.6% → Out of all tweets, the model predicted about two-thirds correctly.

TPR (True Positive Rate) → How many positive tweets the model got right for each group:

Female = 0.5 → It correctly caught only half of the positive tweets about females.

Male = 1.0 → It correctly caught all of the positive tweets about males.

FPR (False Positive Rate) → How many negative tweets were wrongly predicted as positive:

Female = 0.5 → It made mistakes on half of the negative tweets about females.

Male = 0.0 → It made no mistakes on negative tweets about males.

Start coding or generate with AI.

```
import numpy as np
```

```
groups = df['gender_ref']
labels = df['label']
```

```
# get counts
counts = {}
for g, y in zip(groups, labels):
    counts[(g, y)] = counts.get((g, y), 0) + 1

# weight = 1 / count for that (group,label)
sample_weights = np.array([1.0 / counts[(g, y)] for g, y in zip(groups, labels)])
```

```
# weight = 1 / count for that (group,label)
sample_weights = np.array([1.0 / counts[(g, y)] for g, y in zip(groups, labels)])
```

```
from fairlearn.reductions import ExponentiatedGradient, DemographicParity
from sklearn.linear_model import LogisticRegression

base = LogisticRegression(max_iter=1000)
mitigator = ExponentiatedGradient(base, constraints=DemographicParity())
mitigator.fit(X_train_vec.toarray(), y_train, sensitive_features=gender_train)
y_pred = mitigator.predict(X_test_vec.toarray())
```

```
from fairlearn.postprocessing import ThresholdOptimizer
from sklearn.linear_model import LogisticRegression
```

```
clf = LogisticRegression(max_iter=1000).fit(X_train_vec, y_train)
probs = clf.predict_proba(X_train_vec)[:,1]

post = ThresholdOptimizer(estimator=clf, constraints="equalized_odds", prefit=True)
post.fit(X_train_vec.toarray(), y_train, sensitive_features=gender_train)
```

ThresholdOptimizer

estimator: LogisticRegression

LogisticRegression

```
y_pred_postprocessed = post.predict(X_test_vec.toarray(), sensitive_features=gender_test)

metric_frame_postprocessed = MetricFrame(
    metrics={
        'TPR': true_positive_rate,
        'FPR': false_positive_rate,
        'Accuracy': accuracy_score
    },
    y_true=y_test,
    y_pred=y_pred_postprocessed,
    sensitive_features=gender_test
)

print("\nFairness metrics by gender (Post-processed):")
print(metric_frame_postprocessed.by_group)
```

```
Fairness metrics by gender (Post-processed):
            TPR  FPR  Accuracy
gender_ref
female      1.0  0.0       1.0
male        1.0  0.0       1.0
```

Metric	Female(Before)	Male (Before)	Female (After)	Male (After)
True Positive Rate (TPR)	0.50	1.00	1.00	1.00
False Positive Rate (FPR)	0.50	0.00	0.00	0.00
Accuracy	–	–	1.00	1.00
Overall Accuracy	66.6%		100%	

Conclusion: After applying Fairlearn’s post-processing, the model treats male and female tweets equally – both have perfect detection and zero mistakes.