IAS Audit Report for COMPAS Recidivism Prediction Model

Overview

This report presents the results of a fairness audit conducted on a baseline Logistic Regression model trained on the COMPAS dataset. The analysis focuses on identifying racial biases between the privileged group (Caucasians) and the unprivileged group (African-Americans) in predicting recidivism outcomes. The audit includes data preparation, model training, fairness evaluation, and visualization of bias metrics.

Code Implementation

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from aif360.datasets import CompasDataset
from aif360.metrics import BinaryLabelDatasetMetric, ClassificationMetric
from aif360.algorithms.preprocessing import Reweighing
from aif360.algorithms.inprocessing import AdversarialDebiasing
import tensorflow as tf

# Load COMPAS dataset
dataset = CompasDataset()

# Define privileged and unprivileged groups
privileged_groups = [{'race': 1}]  # Caucasian
unprivileged_groups = [{'race': 0}]  # African-American

# Split dataset into train and test sets
train, test = dataset.split([0.7], shuffle=True)

# Extract features and labels
X_train = train.features
y_train = train.labels.ravel()
X_test = test.features
y_test = test.labels.ravel()
```

```
# Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Create predicted dataset for fairness evaluation
test_pred = test.copy()
test_pred.labels = y_pred.reshape(-1, 1)

# Evaluate fairness metrics
metric_test = BinaryLabelDatasetMetric(test,
                                       unprivileged_groups=unprivileged_groups,
                                       privileged_groups=privileged_groups)
metric_pred = ClassificationMetric(test, test_pred,
                                   unprivileged_groups=unprivileged_groups,
                                   privileged_groups=privileged_groups)

# Compute fairness metrics
stat_parity_diff = metric_pred.statistical_parity_difference()
disp_impact = metric_pred.disparate_impact()
avg_odds_diff = metric_pred.average_odds_difference()
eq_opp_diff = metric_pred.equal_opportunity_difference()
tpr_diff = metric_pred.true_positive_rate_difference()
fpr_diff = metric_pred.false_positive_rate_difference()

# Print fairness metrics
print("Statistical Parity Difference:", stat_parity_diff)
print("Disparate Impact:", disp_impact)
print("Average Odds Difference:", avg_odds_diff)
print("Equal Opportunity Difference:", eq_opp_diff)
print("True Positive Rate Difference:", tpr_diff)
print("False Positive Rate Difference:", fpr_diff)

# Visualize fairness metrics
metrics = {
    'Statistical Parity Difference': stat_parity_diff,
    'Disparate Impact': disp_impact,
    'Average Odds Difference': avg_odds_diff,
    'Equal Opportunity Difference': eq_opp_diff,
    'True Positive Rate Difference': tpr_diff,
    'False Positive Rate Difference': fpr_diff
```

```
}

plt.figure(figsize=(10, 6))
sns.barplot(x=list(metrics.keys()), y=list(metrics.values()), palette='coolwarm')
plt.xticks(rotation=45, ha='right')
plt.title('Fairness Metrics for COMPAS Logistic Regression Model')
plt.ylabel('Metric Value')
plt.tight_layout()
plt.show()

# Optional: Apply Reweighing for bias mitigation
RW = Reweighing(unprivileged_groups=unprivileged_groups,
                privileged_groups=privileged_groups)
RW.fit(train)
train_transf = RW.transform(train)

# Retrain model on reweighed data
X_train_rw = train_transf.features
y_train_rw = train_transf.labels.ravel()
model_rw = LogisticRegression(max_iter=1000)
model_rw.fit(X_train_rw, y_train_rw)
y_pred_rw = model_rw.predict(X_test)

# Evaluate fairness after reweighing
test_pred_rw = test.copy()
test_pred_rw.labels = y_pred_rw.reshape(-1, 1)
metric_pred_rw = ClassificationMetric(test, test_pred_rw,
                                      unprivileged_groups=unprivileged_groups,
                                      privileged_groups=privileged_groups)

print("Post-Reweighing Statistical Parity Difference:", metric_pred_rw.statistical_parity_difference())
print("Post-Reweighing Disparate Impact:", metric_pred_rw.disparate_impact())
```

Data Analysis Summary

- **Dataset:** COMPAS Recidivism Dataset
- **Samples:** 6167
- **Features:** 401
- **Missing Data:** 5 rows removed
- **Privileged Group:** Caucasian ({'race': 1})
- **Unprivileged Group:** African-American ({'race': 0})
- **Train/Test Split:** 4316 / 1851 samples
- **Scaler:** StandardScaler
- **Model:** Logistic Regression

Bias Audit Results

Metric	Value	Interpretation
Statistical Parity Difference	-0.1646	Indicates bias against the unprivileged group; fewer favorable outcomes.
Disparate Impact	0.7730	Below the fairness threshold (0.8–1.25), showing disadvantage for African-Americans.
Average Odds Difference	-0.1390	Suggests higher false positives and/or lower true positives for the unprivileged group.
Equal Opportunity Difference	-0.0945	Lower true positive rate for African-Americans.
True Positive Rate Difference	-0.0945	Confirms reduced correct identification of non-recidivists in the unprivileged group.
False Positive Rate Difference	-0.1834	Significantly higher false positive rate for African-Americans.

Interpretation and Implications

The baseline Logistic Regression model demonstrates clear racial bias against African-Americans. The unprivileged group is more likely to be incorrectly classified as high-risk recidivists and less likely to receive favorable predictions. Such disparities can reinforce systemic inequalities in judicial decision-making, leading to unfair treatment in sentencing and parole outcomes.

Recommended Remediation Strategies

1. **Reweighing (Pre-processing):**
   Adjust sample weights to balance representation across groups and outcomes before training. This method aims to achieve statistical parity by modifying input data distributions.
2. **Adversarial Debiasing (In-processing):**
   Train a predictor alongside an adversary that attempts to infer the protected attribute (race). The predictor learns to minimize both prediction error and the adversary's success, reducing bias in learned representations.

Conclusion

The audit confirms that the COMPAS Logistic Regression model exhibits measurable racial bias against African-Americans. Implementing fairness-aware algorithms such as Reweighing or Adversarial Debiasing is recommended to mitigate these disparities and promote equitable model performance across demographic groups.