Handout 2: Intermediate Bias Audit Guide

Machine Learning for Smarter Innovation

1 Handout 2: Intermediate Bias Audit Guide

1.1 Step-by-Step Bias Detection

1.1.1 Phase 1: Preparation (1-2 hours)

1.1 Define Protected Attributes Identify characteristics that should not influence decisions: - **Demographics**: Age, gender, race, ethnicity - **Socioeconomic**: Income, education, zip code - **Physical**: Disability status, health conditions - **Intersectional**: Combinations of above

Action: List all protected attributes relevant to your domain.

1.2 Define Success Metrics Choose appropriate fairness metrics for your context: - **High-stakes**: Equalized odds (criminal justice, healthcare) - **Opportunity**: Equal opportunity (hiring, loans) - **Representation**: Demographic parity (admissions, recommendations)

Action: Document which metric(s) you'll use and why.

1.1.2 Phase 2: Data Audit (2-4 hours)

2.1 Representation Analysis

```
import pandas as pd

# Check group sizes
df.groupby('protected_attribute').size()

# Calculate proportions
df['protected_attribute'].value_counts(normalize=True)

# Identify underrepresented groups (< 5%)
```

Red Flag: Any group $< 5\%$ of dataset

2.2 Label Distribution

```
# Check positive rate by group
df.groupby('protected_attribute')['label'].mean()

# Statistical test for difference
from scipy.stats import chi2_contingency
chi2_contingency(pd.crosstab(df['protected_attribute'], df['label']))
```

Red Flag: p-value < 0.05 suggests systematic difference

2.3 Feature Correlation

```
# Check correlation with protected attribute
```

```
df.corr()['protected_attribute'].sort_values()

# Identify proxy variables (correlation > 0.3)
```

Red Flag: Features highly correlated with protected attribute

1.1.3 Phase 3: Model Audit (3-5 hours)

3.1 Disaggregated Performance Using Fairlearn:

```
from fairlearn.metrics import MetricFrame
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Calculate metrics by group
metric_frame = MetricFrame(
    metrics={
        'accuracy': accuracy_score,
        'precision': precision_score,
        'recall': recall_score
    },
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=sensitive_features
)

# Display results
print(metric_frame.by_group)
print(metric_frame.difference())  # Max difference across groups
```

Red Flag: Difference > 0.05 in key metrics

3.2 Fairness Metric Calculation Demographic Parity:

```
from fairlearn.metrics import demographic_parity_difference

dp_diff = demographic_parity_difference(
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=sensitive_features
)

print(f"Demographic Parity Difference: {dp_diff:.3f}")
# Ideal: 0, Acceptable: < 0.1
```

Equal Opportunity:

```
from fairlearn.metrics import equalized_odds_difference

eo_diff = equalized_odds_difference(
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=sensitive_features
)

print(f"Equalized Odds Difference: {eo_diff:.3f}")
# Ideal: 0, Acceptable: < 0.1
```

3.3 Error Analysis by Group

```
from sklearn.metrics import confusion_matrix

for group in sensitive_features.unique():
    mask = sensitive_features == group
    cm = confusion_matrix(y_test[mask], y_pred[mask])

    print(f"\nGroup: {group}")
    print(f"TPR: {cm[1,1]/(cm[1,0]+cm[1,1]):.3f}")
    print(f"FPR: {cm[0,1]/(cm[0,0]+cm[0,1]):.3f}")
```

Red Flag: TPR or FPR differs by > 0.1 across groups

1.1.4 Phase 4: Root Cause Analysis (2-3 hours)

4.1 Bias Source Identification Ask: 1. **Data Bias**: Is the training data unbalanced or unrepresentative? 2. **Label Bias**: Do labels reflect historical discrimination? 3. **Feature Bias**: Are proxy variables leaking protected information? 4. **Model Bias**: Is the algorithm overfitting to majority group?

4.2 Investigation Steps

```
# 1. Check feature importance
import shap
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test)

# 2. Identify problematic features
# Features correlated with protected attribute + high importance

# 3. Analyze prediction distribution
import matplotlib.pyplot as plt
for group in sensitive_features.unique():
    mask = sensitive_features == group
    plt.hist(y_pred_proba[mask], alpha=0.5, label=group)
plt.legend()
plt.show()
```

1.1.5 Phase 5: Mitigation (varies)

5.1 Choose Strategy **Pre-processing** (if data bias): - Reweighting - Resampling - Removing correlated features

In-processing (if model bias): - Fairness constraints - Adversarial debiasing

Post-processing (quick fix): - Threshold optimization - Calibration per group

5.2 Implementation Example

```
from fairlearn.reductions import ExponentiatedGradient
from fairlearn.reductions import EqualizedOdds

# Define constraint
constraint = EqualizedOdds()

# Train fair model
mitigator = ExponentiatedGradient(
    estimator=base_model,
    constraints=constraint
)

mitigator.fit(X_train, y_train, sensitive_features=A_train)
y_pred_fair = mitigator.predict(X_test)
```

```python
# Re-evaluate
metric_frame_fair = MetricFrame(
    metrics={'accuracy': accuracy_score},
    y_true=y_test,
    y_pred=y_pred_fair,
    sensitive_features=A_test
)

print("After mitigation:")
print(metric_frame_fair.by_group)
```

1.1.6 Phase 6: Documentation & Monitoring

6.1 Document Findings Create a bias audit report including: 1. Protected attributes analyzed 2. Fairness metrics measured 3. Disparities found 4. Root causes identified 5. Mitigation strategies applied 6. Residual risks

6.2 Set Up Monitoring

```python
# Monitor performance over time
def monitor_fairness(model, X, y, sensitive_features, timestamp):
    y_pred = model.predict(X)

    metrics = {
        'timestamp': timestamp,
        'accuracy': accuracy_score(y, y_pred),
        'demographic_parity': demographic_parity_difference(y, y_pred,
    sensitive_features),
        'equalized_odds': equalized_odds_difference(y, y_pred,
    sensitive_features)
    }

    # Log to monitoring system
    return metrics

# Run monthly
```

1.1.7 Tools Summary

Tool	Best For	Learning Curve
Fairlearn	Quick sklearn integration	Low
AIF360	Comprehensive analysis	Medium
What-If Tool	Visual exploration	Low
Aequitas	Detailed reporting	Medium

1.1.8 Common Pitfalls

1. **Testing only overall accuracy** - Always disaggregate
2. **Ignoring intersectionality** - Consider combinations of attributes
3. **One-time audit** - Bias can emerge post-deployment
4. **Over-correcting** - Balance fairness and utility
5. **Assuming technical fix is sufficient** - Consider systemic issues

1.1.9 Key Takeaway

Bias auditing is not a checkbox - it's an ongoing process requiring technical skills, domain knowledge, and ethical judgment.