Automating Training & Fairness Checks

AI Masters Capstone Project - Presentation 3

Jonathan Agustin

November 2024

What We'll Cover Today

- Automating model training: pipelines, hyperparameter tuning, CI/CD integration
- Model versioning & experiment tracking: reproducibility and accountability
- Explainability & transparency: fostering user trust, meeting compliance
- Fairness in AI models: detecting and mitigating biases
- Evaluation: data splits, handling class imbalance, interpretability-speed trade-offs

Automating Model Training

- Define pipelines for consistent preprocessing, training, and evaluation
- Integrate with CI/CD for reproducible, scalable runs
- Focus on insight and strategy, not repetitive labor

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

pl = Pipeline([('scaler', StandardScaler()), ('clf', LogisticRegression())])
pl.fit(X_train, y_train)
```

Hyperparameter Optimization

- Systematically explore parameters via grid, random, or Bayesian search
- Bayesian optimization learns from past trials for efficient convergence
- Log trials for traceability and reproducibility

```python
from skopt import BayesSearchCV

param_space = {'clf__C': (1e-3, 1e3, 'log-uniform')}
opt = BayesSearchCV(pl, param_space, n_iter=20, scoring='f1')
opt.fit(X_train, y_train)
```

Model Versioning & Experiment Tracking

- Log metrics, parameters, environment details per run
- Quickly revert or reproduce past experiments
- Maintain a living repository of model evolution

```python
import mlflow
with mlflow.start_run():
    pl.fit(X_train, y_train)
    acc = pl.score(X_val, y_val)
    mlflow.log_metric("accuracy", acc)
    mlflow.sklearn.log_model(pl, "model")
```

Transparent & Explainable AI

- Use SHAP/LIME to understand feature impact on predictions
- Provide evidence for decisions, enhancing trust
- Identify spurious reasoning or biases through interpretability

```python
import shap
explainer = shap.LinearExplainer(pl['clf'], X_train)
sv = explainer.shap_values(X_val[:1])
shap.force_plot(explainer.expected_value, sv, X_val.iloc[0])
```

Fairness in AI Models

- Evaluate performance by subgroup to detect disparities
- Apply fairness metrics, re-weighting, or threshold adjustments
- Continuously monitor fairness as data and usage evolve

```
# Fairness check (pseudo-code)
priv_scores = pl.score(X_priv, y_priv)
unpriv_scores = pl.score(X_unpriv, y_unpriv)
disparity = unpriv_scores / priv_scores
if disparity < 0.8:
    print("Potential bias detected.")
```

Considerations for Model Evaluation

- Proper splits (train, val, test) to prevent overfitting
- Use metrics beyond accuracy: precision, recall, F1, fairness measures
- Address class imbalance to avoid overlooking critical minority classes
- Consider interpretability and latency constraints
- Document biases, limitations, and known failure modes

Properly Splitting Your Data

- Train/Val/Test splits for unbiased assessment
- Representative distributions for realistic performance estimates
- Keep test set stable for consistent comparisons

```
from sklearn.model_selection import train_test_split
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.2)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.25)
```

Handling Class Imbalance

- Accuracy can hide poor minority class performance
- Use precision/recall/F1 or AUC for minority-focused insight
- Adjust class weights or resample to improve fairness

```
from sklearn.utils import class_weight
cw = class_weight.compute_class_weight(
    'balanced', classes=np.unique(y_train), y=y_train)
pl['clf'].set_params(class_weight=dict(enumerate(cw)))
```

Offline vs. Online Evaluation

- Offline: controlled, stable benchmarks for initial model selection
- Online: real-world feedback, latency, data drift considerations
- Combine both for sustained relevance and reliability

Trade-offs in Model Evaluation

- Interpretability vs. accuracy: transparent decisions may matter more than marginal accuracy gains
- Latency vs. complexity: speed can be critical for real-time applications
- Resource costs vs. performance: consider environmental and financial footprints

Limitations and Bias in Evaluation

- Acknowledge no metric is perfect
- Document known biases and failure modes in model cards
- Periodically test on challenging or adversarial data to ensure resilience

A Holistic, Ethical Development Lifecycle

- Automate and track to ensure reliability and scalability
- Integrate explainability, fairness, and balanced evaluation
- Adapt and refine with changing data and needs

This integrated approach elevates ML from mere functionality to principled, ethical practice.

Next Steps

- Next: Deployment Automation & Ethical Deployment Practices
- Extend fairness, transparency, and compliance into production
- Continuously monitor and adapt to maintain trust and efficacy