So now when we do tuning using average precision does it mean that calibration is done during that time or after all the parameters are selected we do the tuning?

Here's the crucial relationship between calibration and hyperparameter tuning using average precision (AP):

1. Order of Operations Matters

Standard Workflow:

```
graph LR
A[Hyperparameter Tuning] --> B[Select Best Model]
B --> C[Calibrate Best Model]
```

Key Insight:

- **Tuning uses uncalibrated probabilities** to calculate AP scores
- **Calibration is applied post-tuning** to adjust probabilities of the final selected model

2. Why This Sequence?

Step	Calibration Status	AP Calculation Basis	Impact
Hyperparameter Search	**Not calibrated**	Raw probabilities	Finds parameters that optimize ranking (AP cares about order, not absolute values)
Model Calibration	**Applied after**	Calibrated probabilities	Makes probabilities truthful for deployment/thresholding

Example:

- Model A: Raw probabilities [0.3, 0.5, 0.7] → AP=0.82
- Model B: Raw probabilities [0.1, 0.2, 0.9] → AP=0.85
- Tuning picks Model B **based on raw outputs**
- After calibration: Model B's probs → [0.4, 0.6, 0.95] (now truthful)

3. Why Not Calibrate During Tuning?

1. **Computational Cost**: Calibration adds 2-3× training time per configuration
2. **Data Leakage Risk**: Requires nested cross-validation to avoid overfitting
3. **AP is Ranking-Based**: Calibration doesn't affect instance ordering (AUCPC metric)

```
# WRONG APPROACH (calibration during tuning)
opt = BayesSearchCV(
    CalibratedClassifierCV(base_model),  # ✖ Avoid!
    param_space,
    scoring='average_precision'
)
```

4. Your Code's Calibration Flow

```
# 1. Tune WITHOUT calibration (raw probabilities)
opt.fit(X_train, y_train)  # Uses raw probs for AP scoring

# 2. Calibrate AFTER selecting best model
calibrated_model = CalibratedClassifierCV(opt.best_estimator_)
calibrated_model.fit(X_val, y_val)  # Separate holdout set
```

Key Properties:

- Tuning optimizes **discrimination** (separation of classes)
- Calibration fixes **reliability** (probability = true likelihood)
- Combined result: Model that both ranks well **and** has truthful probabilities

5. When to Calibrate During Tuning?

Only if you need **calibration-sensitive metrics** like Brier Score:

```
from sklearn.metrics import brier_score_loss, make_scorer

# Custom scorer requiring calibration
brier_scorer = make_scorer(brier_score_loss, needs_proba=True)

# Nested calibration approach
opt = BayesSearchCV(
    CalibratedClassifierCV(base_model, cv=3),
    param_space,
    scoring=brier_scorer,
    cv=3
)
```

6. Practical Implications for Your Case

- **Threshold Selection**: Must use calibrated probabilities
- **Performance Reporting**:

  ```
  # Report both calibrated and uncalibrated AP
  raw_ap = average_precision_score(y_test, opt.best_estimator_.predict_proba(X_test)[:,
  cal_ap = average_precision_score(y_test, calibrated_model.predict_proba(X_test)[:, 1]
  print(f"Raw AP: {raw_ap:.3f} → Calibrated AP: {cal_ap:.3f}")
  ```

- **Deployment**: Always use the calibrated model

Key Takeaway

Tuning with AP **doesn't require calibration** because it's a ranking metric. Calibration is a **post-hoc adjustment** to make probabilities meaningful for business decisions (e.g., "What threshold gives 90% precision?").