perplexity

What does these indicate in the code?

n_iter=30,
n_points=3,
refit=False

Is it going to run only for 3 configurations?

Also What is calibration doing in this code

Let's break down your questions:

1. Bayesian Optimization Parameters Explained

Parameter	Value	Meaning
`n_iter=30`	30	Total number of iterations (steps) in optimization
`n_points=3`	3	Number of parallel evaluations **per iteration**
`refit=False`	False	Don't retrain best model on full data after search

Key Implications:

- **Total Configurations** = `n_iter` × `n_points` = **30×3=90** unique configurations evaluated
- **Parallelization**: 3 models trained simultaneously per iteration
- **No Final Refit**: Returns the best model found during CV (not retrained on full data)

2. Calibration in Class Imbalance Context

Purpose:
When dealing with extreme class imbalance (1:60,000), model outputs become poorly calibrated - predicted probabilities don't reflect true likelihoods. Calibration fixes this.

Code Implementation:

```
from sklearn.calibration import CalibratedClassifierCV
calibrated = CalibratedClassifierCV(
    opt.best_estimator_,
    method='sigmoid',
    cv=PredefinedSplit(...)
)
```

What It Does:

- **Sigmoid Scaling (Platt Scaling)**: Fits a logistic regression to map raw model outputs to calibrated probabilities
- **Why Critical**:
  - Original probabilities might cluster near 0 (e.g., [0.0001, 0.001] for minority class)
  - Calibration spreads probabilities meaningfully (e.g., [0.3, 0.7])
  - Enables proper threshold tuning for metrics like Average Precision

Visual Comparison:

Metric	Before Calibration	After Calibration
Predicted Probability Range	[0.0001, 0.0015]	[0.01, 0.98]
Brier Score (↓ better)	0.0001 (deceptively good)	0.12 (truthful)
Reliability Curve	Flat line	Diagonal alignment

3. Why This Matters for Your Case

- **Threshold Optimization**:
  Without calibration, even "good" predictions like 0.001 might correspond to 80% true positive rate

```
# Uncalibrated vs calibrated thresholds
optimal_uncalibrated_threshold = 0.0005  # Unstable
optimal_calibrated_threshold = 0.45      # Robust
```

- **Metric Trustworthiness**:
  Average Precision (AP) becomes unreliable with miscalibrated probabilities

```
# AP before/after calibration
AP_raw = 0.82  # Overestimated
AP_calibrated = 0.78  # True performance
```

4. Key Recommendations

1. **Keep** `refit=False` - With early stopping and huge data, retraining risks overfitting
2. **Increase** `n_iter` to ≥50 if computationally feasible
3. **Always calibrate** when using `scale_pos_weight > 1000`
4. **Monitor Calibration**:

```
from sklearn.calibration import calibration_curve
prob_true, prob_pred = calibration_curve(y_test, probs, n_bins=10)
plt.plot(prob_pred, prob_true, marker='o')
```

This setup ensures you get both:

- **Optimized hyperparameters** through Bayesian search
- **Trustworthy probabilities** via post-hoc calibration

For implementation details, see scikit-learn calibration docs.