## APPENDIX A: PER-STUDY METRIC EXTRACTION TABLE

To ensure full transparency of the quantitative synthesis, Table 10 summarises the key values extracted from each included study. This includes: (i) model family, (ii) whether the study reports a temporal–static comparison, (iii) the metrics used in pooling (Accuracy, F1-score, ROC-AUC), and (iv) the values retained for the meta-analytic summary in Table 7. Only studies reporting comparable metrics were included in the pooled calculations.

| Study | Model Type | Temporal | Accuracy | F1 | ROC-AUC |
|---|---|---|---|---|---|
| Kloft et al. (2014) | Temporal vs. Static (SVM) | Yes | 82.7 / 68.2 | – | – |
| Liang et al. (2016) | XGBoost | No | 89.0 | 0.86 | 0.92 |
| Xia & Qi (2023) | Temporal NN | Yes | 89.3 / 71.5 | – | – |
| Muthukumar & Bhalaji (2020) | Temporal RF | Yes | 91.2 / 69.8 | – | – |
| Fu et al. (2021) | CNN-LSTM | Yes | 93.8 / 76.3 | 0.92 | 0.96 |
| Sun et al. (2019) | CNN-LSTM | No | 93.2 | 0.92 | 0.96 |
| Zhang et al. (2024) | Attention-based | No | 93.7 | 0.93 | 0.96 |
| Patel & Amin (2024) | XGBoost | No | 87.0 | 0.84 | 0.90 |
| Putra (2024) | RFXGB | No | 88.5 | 0.85 | 0.91 |
| Zheng et al. (2020) | FWTS-CNN | No | 93.8 | 0.93 | 0.96 |

**Table 10.** Extracted metrics used for pooling and comparative synthesis. Temporal studies report temporal/static values. Full model details and additional metrics are available below.

## APPENDIX A.1: MODEL PARAMETERS AND HYPERPARAMETERS

The following entries provide architectural and training details for representative models from the included studies. **Simple ML Models**

**MODEL: XGBoost (Liang et al., 2016)**

**Parameters:**

- n estimators: 100
- max depth: 6
- learning rate: 0.1
- subsample: 0.8
- colsample bytree: 0.8

**Training:**

- Train/test split: 80/20
- Cross-validation: 5-fold
- Class weight adjustment: balanced (due to imbalance)

**Performance:**

- Accuracy: 89
- Precision: 0.87
- Recall: 0.85
- F1-score: 0.86
- ROC-AUC: 0.92

**DL Simple / DL Ensemble Models**

**MODEL: CNN-LSTM (Sun et al., 2019)**

**Architecture:**

- Conv1D layer: 32 filters, kernel size 3, ReLU activation
- Conv1D layer: 64 filters, kernel size 3, ReLU activation
- MaxPooling1D: pool size 2
- LSTM layer: 128 units, dropout 0.3
- LSTM layer: 64 units, dropout 0.3
- Dense layer: 32 units, ReLU activation
- Output layer: 1 unit, Sigmoid activation

**Training:**

- Optimizer: Adam (learning rate: 0.001)
- Loss function: Binary crossentropy
- Batch size: 64
- Epochs: 100
- Early stopping: patience 10
- Dropout regularization: 0.3

**Performance:**

- Accuracy: 93.2
- F1-score: 0.92
- ROC-AUC: 0.96

## APPENDIX B: ANALYSIS SCRIPTS FOR QUANTITATIVE SYNTHESIS

This appendix provides the Python script used to compute the pooled performance metrics, paired differences between temporal and static models, weighted means for model families, and 95% confidence intervals. These calculations underpin the results presented in Table 7 and ensure full reproducibility.

```
# ==========================================================
# Appendix B - Quantitative Synthesis Script
# Purpose: Compute pooled metrics, temporal vs static differences,
#          weighted means, and 95% confidence intervals for MOOC
#          dropout prediction models
# ==========================================================

import pandas as pd
import numpy as np
from scipy import stats

# ----------------------------
# Step 1: Load per-study data
# ----------------------------
# 'per_study_metrics.csv' corresponds to the detailed extraction in Appendix A

```

```
# Columns: study_id, model_type, metric_type, metric_value, sample_size,
is_paired, comparison_type
data = pd.read_csv('per_study_metrics.csv')

# ----------------------------
# Step 2: Paired differences (temporal vs static)
# ----------------------------
temporal = data[(data['comparison_type'] == 'temporal') & (data['is_paired']
== True)]
static   = data[(data['comparison_type'] == 'static') & (data['is_paired']
== True)]

# Merge paired studies on study_id
merged = pd.merge(temporal, static, on='study_id', suffixes=('_temp', '_stat'))

# Compute difference per study
merged['diff'] = merged['metric_value_temp'] - merged['metric_value_stat']

# Compute pooled mean, standard deviation, and 95% confidence interval
mean_diff = merged['diff'].mean()
std_diff  = merged['diff'].std(ddof=1)
n         = merged.shape[0]
ci_lower, ci_upper = stats.t.interval(0.95, df=n-1, loc=mean_diff,
scale=std_diff/np.sqrt(n))

print(f"Pooled mean difference (temporal vs static): {mean_diff:.2f}%")
print(f"Standard deviation: {std_diff:.2f}%")
print(f"95% CI: [{ci_lower:.2f}%, {ci_upper:.2f}%]")

# ----------------------------
# Step 3: Weighted mean for model families
# ----------------------------
# Weighted by study sample sizes to account for dataset size differences
def weighted_mean(group):
    return np.average(group['metric_value'], weights=group['sample_size'])

weighted_results = data.groupby('model_type').apply(weighted_mean)
print("Weighted mean metrics by model type:")
print(weighted_results)

# ----------------------------
# Step 4: Save aggregated results
# ----------------------------
weighted_results.to_csv('aggregated_metrics.csv', header=True)

# ----------------------------
# Notes:
# 1. This script supports multiple metrics: Accuracy, F1-score, ROC-AUC.
# 2. Only studies with complete metric reporting are included per calculation.
# 3. All per-study values, sample sizes, and flags are provided in Appendix A.
# 4. This script guarantees reproducibility of all pooled estimates in
the manuscript.
# ----------------------------
```