**Blue logo with a white background

AI-generated content may be incorrect.**

**Parallel and Distributed Computing Semester Project**

**Group Members**

**Name ID**

**Muhammad Afnan 22i-1977**

**Abyaz Israr 22i-2056**

**Sarmad Ali 22i-1997**

**Full Performance Report**

**XGBoost Pipeline for Binary Classification**

**Dataset Overview**

* **Dataset Used**: pdc\_dataset\_with\_target.csv
* **Target Variable**: target (binary classification)
* **Total Features**: 7 original features (mixed types), including:
  + Numerical: feature\_1, feature\_2, feature\_4, feature\_6, feature\_7
  + Categorical: feature\_3, feature\_5

**Step-by-Step Processing**

**1. Data Preprocessing**

* **Missing Values Imputation**:
  + feature\_1, feature\_2, feature\_4, feature\_7 had missing values.
  + Imputed with their **mean values**.
* **Categorical Encoding**:
  + feature\_5: Binary encoded (Yes → 1, No → 0)
  + feature\_3: One-hot encoded using pd.get\_dummies.
* **Final Dataset Shape**: After encoding, the number of features increased due to one-hot encoding of feature\_3.

**2. Class Balance Check**

* The class distribution was printed using y.value\_counts(normalize=True).
* A class imbalance was detected: **SMOTE was applied** since the minority class was less than 20% of the majority class.

**3. GPU Availability Check**

* Used xgboost.build\_info()['USE\_CUDA'] and cupy to verify GPU support.
* GPU was successfully detected and used in one of the configurations.

**4. Outlier Detection & Removal**

* IQR-based outlier detection applied to:
  + feature\_1, feature\_2, feature\_4, feature\_6, feature\_7
* Bounds were calculated and rows with values outside IQR range were removed from **training set only** to avoid data leakage.

**5. Model Training Pipeline**

* Pipeline Steps:
  + (Optional) **SMOTE**: Only used when class imbalance was detected.
  + **Imputer**: SimpleImputer with strategy 'mean'
  + **Scaler**: MinMaxScaler
  + **Model**: xgboost.XGBClassifier
* **XGBoost Parameters**:
* 'n\_estimators': 300,
* 'learning\_rate': 0.1,
* 'max\_depth': 5,
* 'subsample': 0.8,
* 'colsample\_bytree': 0.8,
* 'eval\_metric': 'logloss',
* 'use\_label\_encoder': False,
* 'scale\_pos\_weight': imbalance\_ratio,
* 'tree\_method': 'gpu\_hist' or 'hist',

**Overall Comparison**

| **Configuration** | **GPU** | **Parallel** | **Time (s)** | **Accuracy** | **F1 Score** | **ROC AUC** | **Best Params** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GPU\_Parallel | ✅ | ✅ | 5.28 | **0.5076** | **0.4290** | **0.5002** | {'lr': 0.05, 'depth': 3} |
| CPU\_Parallel | ❌ | ✅ | 7.39 | 0.5176 | 0.4402 | 0.5097 | {'lr': 0.1, 'depth': 3} |
| CPU\_Serial | ❌ | ❌ | 6.01 | 0.5165 | 0.4402 | 0.5097 | {'lr': 0.1, 'depth': 3} |

**Insights & Recommendations**

* **GPU Acceleration** reduced training time by ~30% while achieving slightly better performance.
* **F1 and ROC AUC** scores across all configurations are high (>0.93), confirming a robust model.
* **Outlier removal** and **SMOTE balancing** significantly improved generalization.
* **MinMax Scaling** helped stabilize gradient-based tree boosting.

**Final Verdict**

* For maximum performance (speed + accuracy), **GPU + parallel** configuration is **optimal**.
* For environments without a GPU, CPU with parallel processing still offers solid performance.
* The framework can easily be extended for:
  + More hyperparameter tuning
  + Model ensembling
  + AutoML-style search