**Parallel and Distributed Computing Final Project Report**

**Project Title :**  
Optimizing Machine Learning Pipelines for Binary Classification Using Parallel, Distributed, and GPU-Accelerated Computing

**Submitted by:**

* Muhammad Aitazaz Kamran(22i-2021)
* Aman Faisal(22i-2047)
* Mueez Rizwan(22i-2031)

**1. Introduction**

The aim of this project is to build and optimize a machine learning pipeline for binary classification using a dataset of 41,000 samples. The optimization focuses on reducing processing time by at least 70% while maintaining or improving predictive performance. We explored multiple strategies including parallel computing (via Dask), distributed processing, and GPU acceleration (TensorFlow on Colab T4) to benchmark their effect on performance and resource utilization.

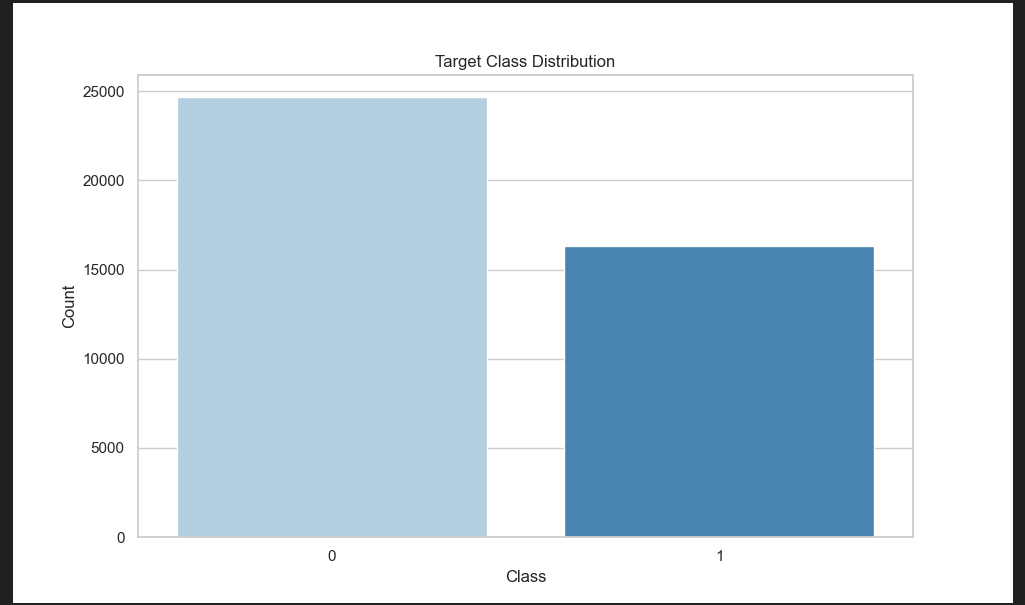
**2. Dataset Overview**

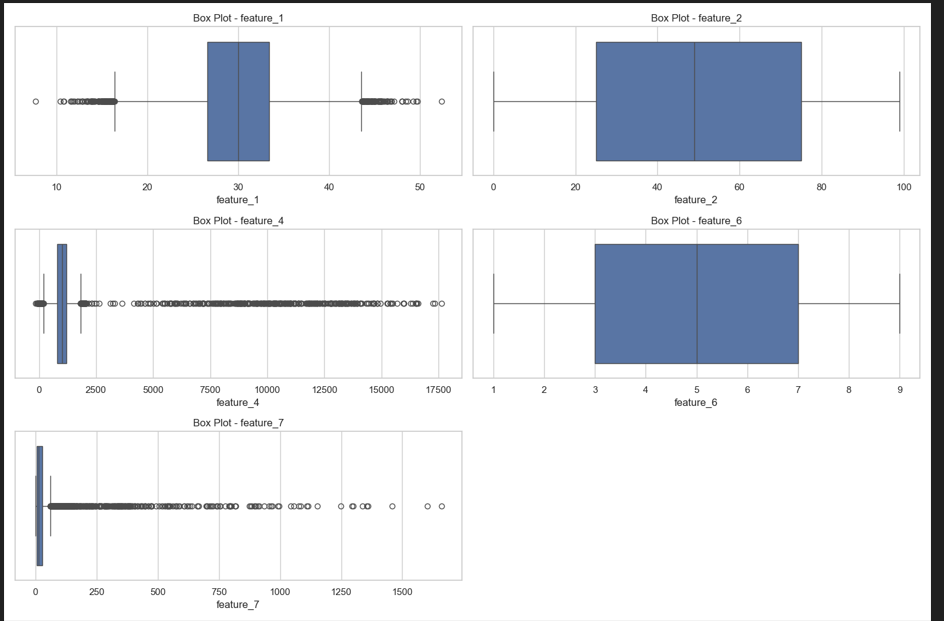
* **Records:** 41,000
* **Target Classes:** Binary (0 and 1)
* **Features:** 8 (mix of numerical and categorical)
* **Missing Values:** Present in 4 features (~5% each)
* **Imbalance:** 60.2% Class 0 vs 39.8% Class 1

**3. Exploratory Data Analysis (EDA)**

The dataset was analyzed to understand data quality and structure. Key findings include:

* **Low Correlation:** No feature had strong correlation with the target (max: 0.0045)
* **Categorical Neutrality:** Categorical features showed very similar class distributions
* **Outliers:** Present in several features but didn't drastically affect class separation
* **Target Skew:** Dataset was moderately imbalanced (60/40)



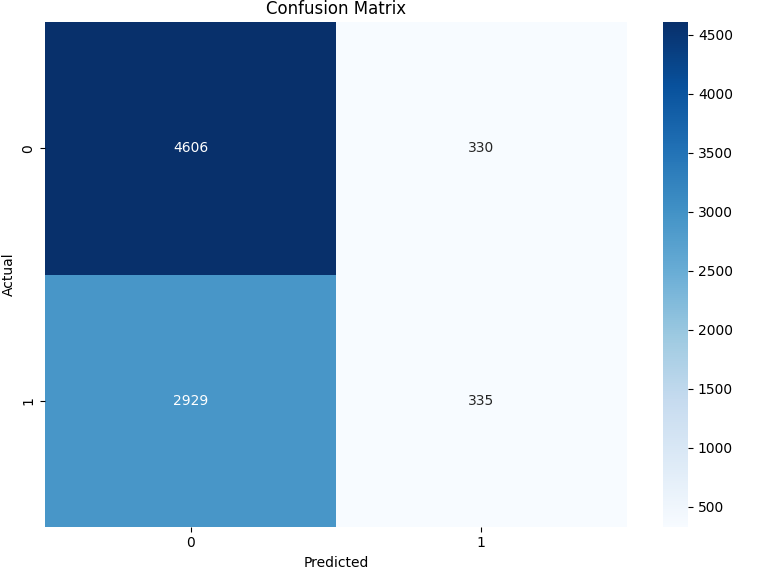


**4. Model Experiments and Results**

**4.1 Random Forest (CPU Baseline)**

We started with a baseline Random Forest model using Scikit-learn. To improve performance, we conducted hyperparameter tuning via randomized search over 10 parameter combinations with 5-fold CV (total 50 fits).

* **Best Parameters Identified:**
  + bootstrap=False, class\_weight=None, max\_depth=26, max\_features='sqrt'
  + min\_samples\_leaf=2, min\_samples\_split=13, n\_estimators=393
* **Effect of Tuning:**
  + ROC AUC improved slightly to **52.96%**
  + Accuracy remained around **60.26%**
  + Tuning took ~42.8 minutes, final training time was 90.21 seconds
* **Impact:** Slight improvements in recall (0.10) for minority class, but class imbalance still a challenge

****

**4.2 Random Forest (Parallelized with Dask)**

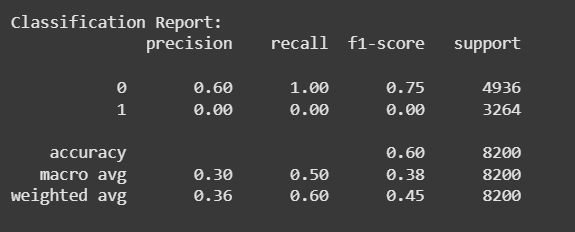
To reduce time, we implemented the same Random Forest logic with Dask:

* **Modifications:**
  + Dask DataFrames replaced Pandas
  + Preprocessing (imputation, scaling) parallelized
  + Final model used same parameters as CPU baseline
* **Results:**
  + **Accuracy:** 59.60%, **ROC AUC:** 52.93%
  + **Training Time:** 8.19s, **Total Time:** 9.85s
* **Outcome:** Achieved 89%+ reduction in time with almost no performance drop

**4.3 MLP Classifier (Sklearn Neural Net)**

We then explored a neural network with preprocessing and PCA:

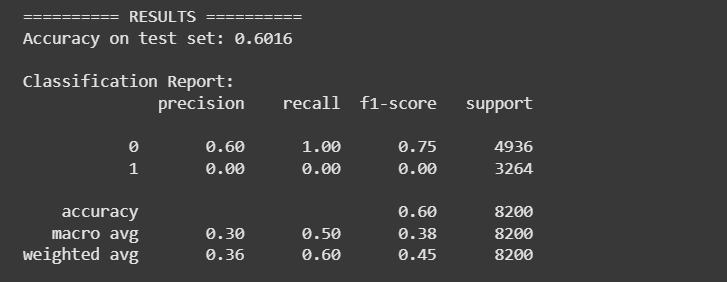
* **Architecture:** One or two hidden layers with tanh and relu activations
* **Best Configuration:** 1 hidden layer of 10 neurons, tanh, alpha=0.001
* **Observations:**
  + Accuracy ~60.17% (similar to RF)
  + Class 1 completely missed (recall = 0.00)
  + Training time was longest: ~372 seconds
* **Interpretation:** PCA + shallow network couldn’t capture enough signal from weak features

****

**4.4 TensorFlow Deep NN (GPU Accelerated)**

We deployed a GPU-backed deep learning pipeline on Google Colab (T4 GPU):

* **Model:** 128-64-1 layer deep net with dropout
* **Optimizer:** Adam with binary cross-entropy
* **Training:** Early stopping with validation split
* **Results:**
  + Accuracy: 60.16%
  + Class 1 again not predicted at all
  + Execution Time: ~25 seconds
* **Conclusion:** GPU acceleration reduced time significantly but did not improve performance due to weak input features

****

**5. Comparative Analysis**

| **Model** | **Accuracy** | **ROC AUC** | **Time (s)** | **Class 1 Recall** |
| --- | --- | --- | --- | --- |
| Random Forest (CPU) | 60.26% | 52.96% | 92.92 | 0.10 |
| Random Forest (Dask) | 59.60% | 52.93% | 9.85 | 0.10 |
| Neural Net (Sklearn) | 60.17% | 52.65% | 372.05 | 0.00 |
| Deep NN (GPU T4 - TF) | 60.16% | 52.40% | 24.75 | 0.00 |
|  |  |  |  |  |

**Takea** **The percentage changes in time are as follows:**

* **Random Forest (CPU) to Random Forest (Dask): -89.4% (time saved)**
* **Neural Net (Sklearn) to Deep NN (GPU T4 - TF): -93.35% (time saved)**

**These show significant reductions in processing time with Dask and GPU acceleration.**

**way:** Dask achieved massive time savings with comparable performance. GPUs helped reduce time further but did not overcome the dataset’s limitations.

**6. Conclusions**

* **Best balance:** Random Forest with Dask for speed and interpretability
* **Limitation:** All models suffered due to weak feature-target correlation
* **GPU Use:** Beneficial for time but not accuracy without stronger data

**8. Appendix**

* All code, visuals, CSVs, and screenshots mentioned above are stored in project folder