A tabular-data classification problem

1. Dataset Description:

- **Training set:** Utilized for constructing and fine-tuning the models.
- **Test set:** Contains unseen data used solely for assessing the model's performance; it is not involved in training or tuning.
- **Blinded test set:** Used to produce the final class-probability predictions.

2. Data preprocessing

a) Remove Duplicate Records

Duplicates in the dataset are eliminated to ensure data integrity.

b) Address Missing Values

There are 23 columns with approximately 36.8% missing data each. Since this is a substantial amount, it's crucial to handle it carefully to prevent bias or significant information loss. After checking their correlation with the target and assessing their variance—both of which were found to be low, we opted to drop these columns using:
train.drop(columns=cols_with_high_missing, inplace=True)

c) Define Features and Target Variable

The target variable (CLASS) is separated from the features. The ID column is also removed, as it serves only as an identifier and holds no predictive value.

d) Outlier Clipping

Instead of removing rows, we clip extreme values to reduce their impact, assuming this is a safer approach. Values are clipped at the 1st and 99th percentiles based on the training set.

e) Standardization

A StandardScaler is applied to normalize the features, with the scaler fitted only on the training data to avoid data leakage.

3. feature engineering

We compare different feature engineering techniques and compare them with a result obtained before feature engineering.

- For dimension reduction we take **PCA (Principal Component Analysis)** in order to reduces features to a few orthogonal components while retaining most variance and **Feature Agglomeration** (Clustering-based) to groups similar features into meta-features.
- **The other techniques are Variance Thresholding:** it removes low-variance (uninformative) features.
- **But the highest result is happen using only PCA (Principal Component Analysis).**

4. Models and Hyperparameters:

- **Logistic Regression** (baseline linear model):
  Used with regularization strength **C = 0.1** and **liblinear** solver.

- **Support Vector Machine (Linear Kernel)**:
  Configured with **C = 0.1**.
  Chosen for its ability to handle high-dimensional feature spaces effectively, thanks to its built-in regularization.

- **Random Forest**:
  Parameters: `'max_depth': 5, 'max_features': 'log2', 'n_estimators': 100.`
  A nonlinear ensemble model known for handling high-dimensional data efficiently and p erforming implicit feature selection.

5. Cross-Validation:

- Performed **5-fold cross-validation** on the training set using **GridSearchCV**.
- Evaluation metrics were **averaged across all folds** to ensure reliable performance estimation.

6. Results Table

Model	Accuracy(%)	Sensitivity(%)	Specificity(%)	F1-score(%)	AUROC(%)
Logistic Regression	67	45.2	82.7	53.5	69.2
SVM (Linear)	59	38.1	74.1	43.8	33.7
Random Forest	57	2	96.5	4	61.9

7. Discussion

- **Strengths:** Effective preprocessing, appropriate model selection, and good generalization.
- **Limitations:** Small dataset with numerus features, class imbalance, and due to limited time there is a limited feature engineering. All this limitation make the result very low.
- **Future Work:**Try ensemble stacking, SMOTE, SHAP for explainability, and AutoML for tuning.