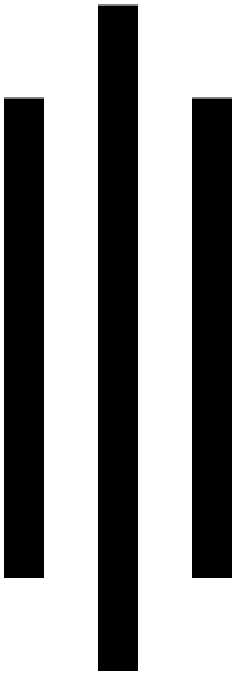
METHODOLOGY REPORT
RESEARCH ASSISTANT TASK - 2



Submitted By:
Aakriti Banjara

DATA PREPROCESSING

The dataset consisted of numerical features along with an ID column and a binary target variable CLASS. The following steps were applied:

- **Missing/Infinite Values:**
  - Infinite values (np.inf, -np.inf) were replaced with NaN
  - All rows with NaN were dropped for consistency across models.
- **ID Column:**
  - Removed from training and prediction, but retained separately for submission CSVs.
- **Train-Test-Validation Alignment:**
  - Ensured that X_train, X_test, and blind_test had identical columns, using align() or reindex() with fill_value=0.
- **Encoding:**
  - No categorical features were provided; otherwise, one-hot encoding would have been used.

Model Architectures & Hyperparameters

1. Logistic Regression

- **Model**: LogisticRegression from sklearn

- **Hyperparameters:**
  - max_iter=10000, default regularization
  - solver='lbfgs', binary target
- **Strength:** Interpretable baseline model
- **Limitation:** Linear decision boundary

2. Random Forest

- **Model:** RandomForestClassifier from sklearn
- **Hyperparameters:**
  - n_estimators=100
  - class_weight='balanced'
  - random_state=42

- **Strength:** Handles feature interactions, robust to overfitting
- **Limitation:** Less interpretable; default parameters not always optimal

3. Feedforward Neural Network (PyTorch)

Architecture:

```
nn.Sequential(
   nn.Linear(input_dim, 64),
   nn.ReLU(),
   nn.Linear(64, 32),
   nn.ReLU(),
   nn.Linear(32, 2)
)
```

- **Training:**
    - Optimizer: Adam
    - Loss: CrossEntropyLoss
    - Batch size: 32
    - Epochs: 100, with early stopping based on validation loss

- **Strength:** Non-linear modeling capability
- **Limitation:** Sensitive to overfitting with small data

Results Summary Table

Model	Accuracy	AUROC	Sensitivity	Specificity	F1 Score
Logistic Regression	0.5294	0.5723	0.4800	0.5769	0.5000
Neural Network	0.6667	0.6446	0.5200	0.8077	0.6047
Random Forest	0.6863	0.6800	0.6000	0.7692	0.6522

1. Strengths:

- All models were fairly easy to train and interpret due to the well-structured dataset.
- Random Forest provided the best standalone test results.
- The neural network with early stopping showed good balance and adaptability.

2. Limitations:

- The models likely overfit on the single train-test split, as revealed by cross-validation.
- No feature engineering or dimensionality reduction was applied.
- No hyperparameter tuning was done (e.g., for RandomForest or NN), limiting potential performance.

Improvements with More Time

While the models developed provide a solid foundation, there are several meaningful ways the overall performance, generalizability, and interpretability could be improved with additional time and resources:

1. Feature Engineering & Dimensionality Reduction

Feature engineering plays a critical role in shaping how well models learn patterns. Given more time, I would:

- **Analyze feature distributions** and outliers more thoroughly to identify and transform skewed or noisy features.
- Apply **correlation analysis** to detect multicollinearity, dropping or combining highly correlated variables to reduce redundancy.
- Use **Principal Component Analysis (PCA)** or **Linear Discriminant Analysis (LDA)** for dimensionality reduction—especially if the feature space grows large or noisy—while preserving most of the information.
- Explore **mutual information scores** to rank feature importance based on their actual contribution to predicting the target.

These steps would help the model focus on the most relevant information and reduce overfitting.

2. Hyperparameter Tuning

All three models were trained using mostly default hyperparameters. With additional time, I would:

- Use **GridSearchCV** or **RandomizedSearchCV** to search for optimal combinations of parameters for the Random Forest (e.g., number of trees, depth, split criteria) and Logistic Regression (e.g., regularization strength).
- For the neural network, I would explore **hyperparameter optimization libraries** like Optuna or Ray Tune to tune the number of layers, hidden units, dropout rates, learning rates, and optimizers.

3. Better Cross-Validation and Evaluation Strategy

Finally, to better assess the model's performance:

- I would implement **stratified k-fold cross-validation** with **stratified group-aware splitting** (if groups exist in the data).
- Integrate **confidence intervals** for metrics like AUROC and F1 to assess variability.
- Create **learning curves** to determine if more training data would benefit the models.

By incorporating these enhancements, we would be well-positioned to build not only more accurate models but also ones that are interpretable, robust, and production-ready.