Report

Presented by

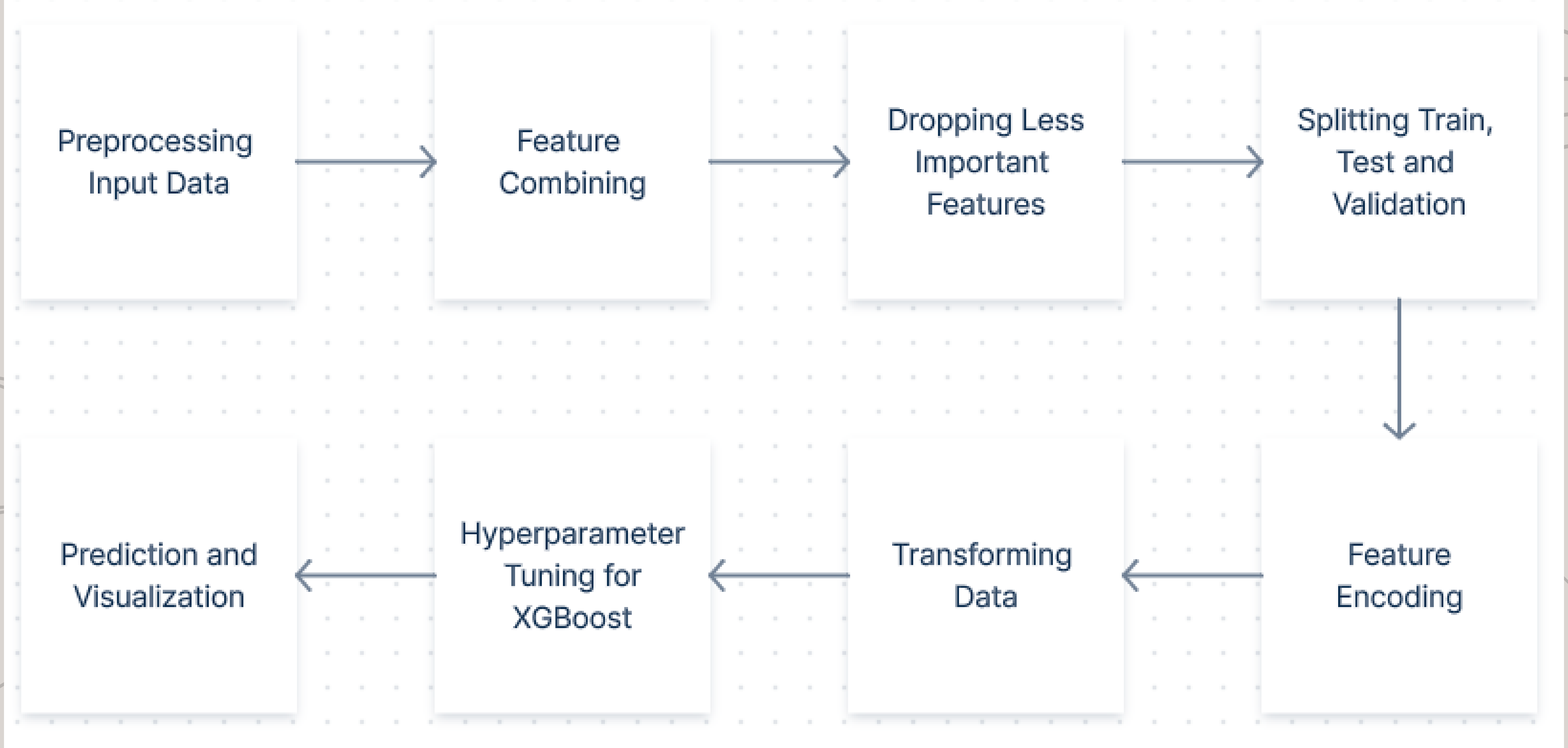
- Pooja Yakkala

Method for Fraud Detection

For the first method, our aim was to minimize the false negatives (Fraud misclassified as not Fraud). We concentrated on balancing both precision and recall. So basically we concentrated on getting a higher F1 score. We performed the following steps for task 1:

- Combined user and merchant location data to calculate distances.
- Dropped less important features (e.g., street, city, zip) with a threshold of 0.01.
- Divided data into train, test, and validation sets (60:20:20).
- Scaled numerical features using a Standard Scaler and encoded categorical features using an Ordinal Encoder.
- Used a hyperparameter grid to determine the best parameters for the XGBoost model.
- The model's objective function was set to binary logistic.
- Stopped once the best F1 score was achieved, balancing precision and recall.

Visualization Overview

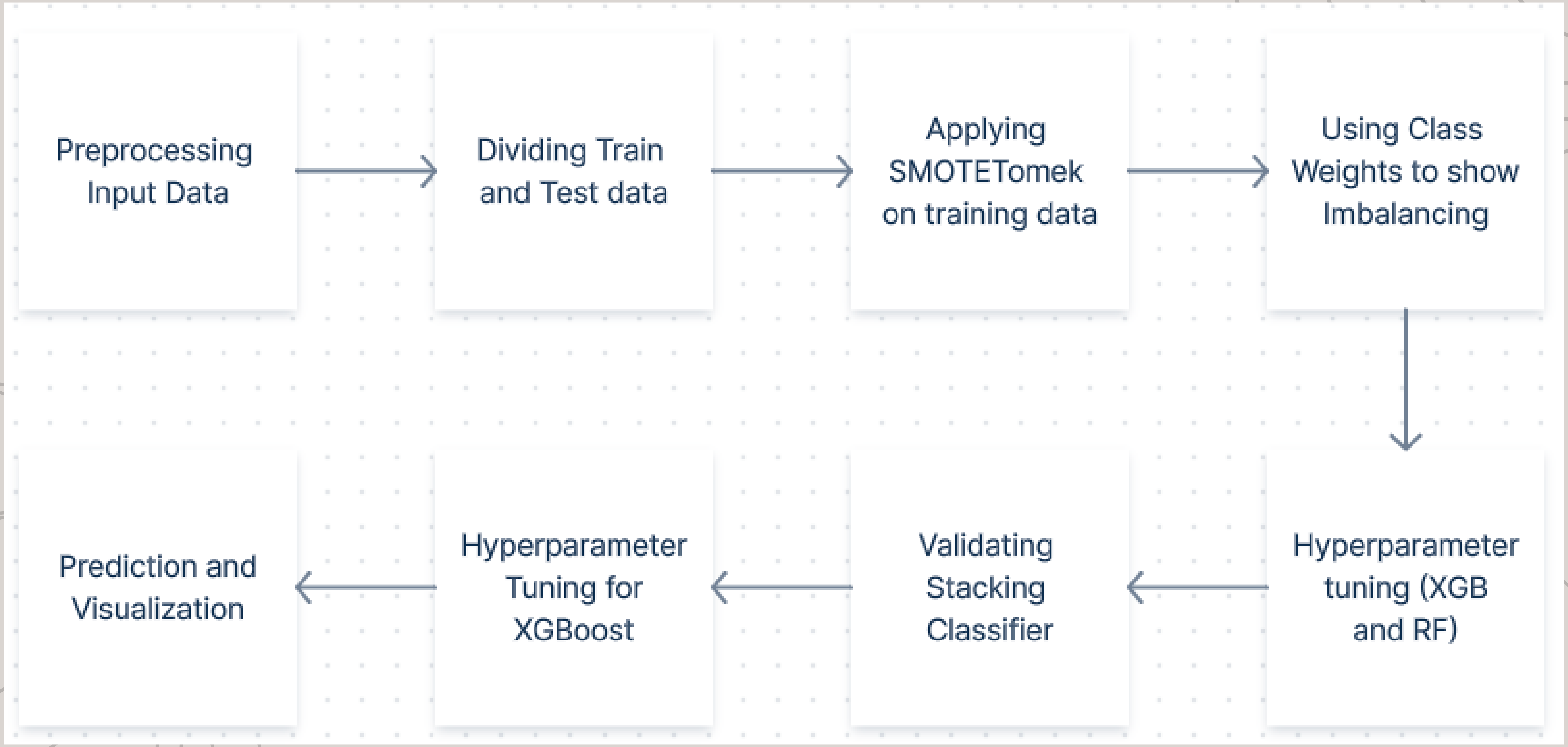


Method for HAR

For the second task, our aim was to classify all the classes correctly. The dataset given was also imbalanced so here we tried to balance the dataset using SMOTETomek. We performed the following steps for task 2,

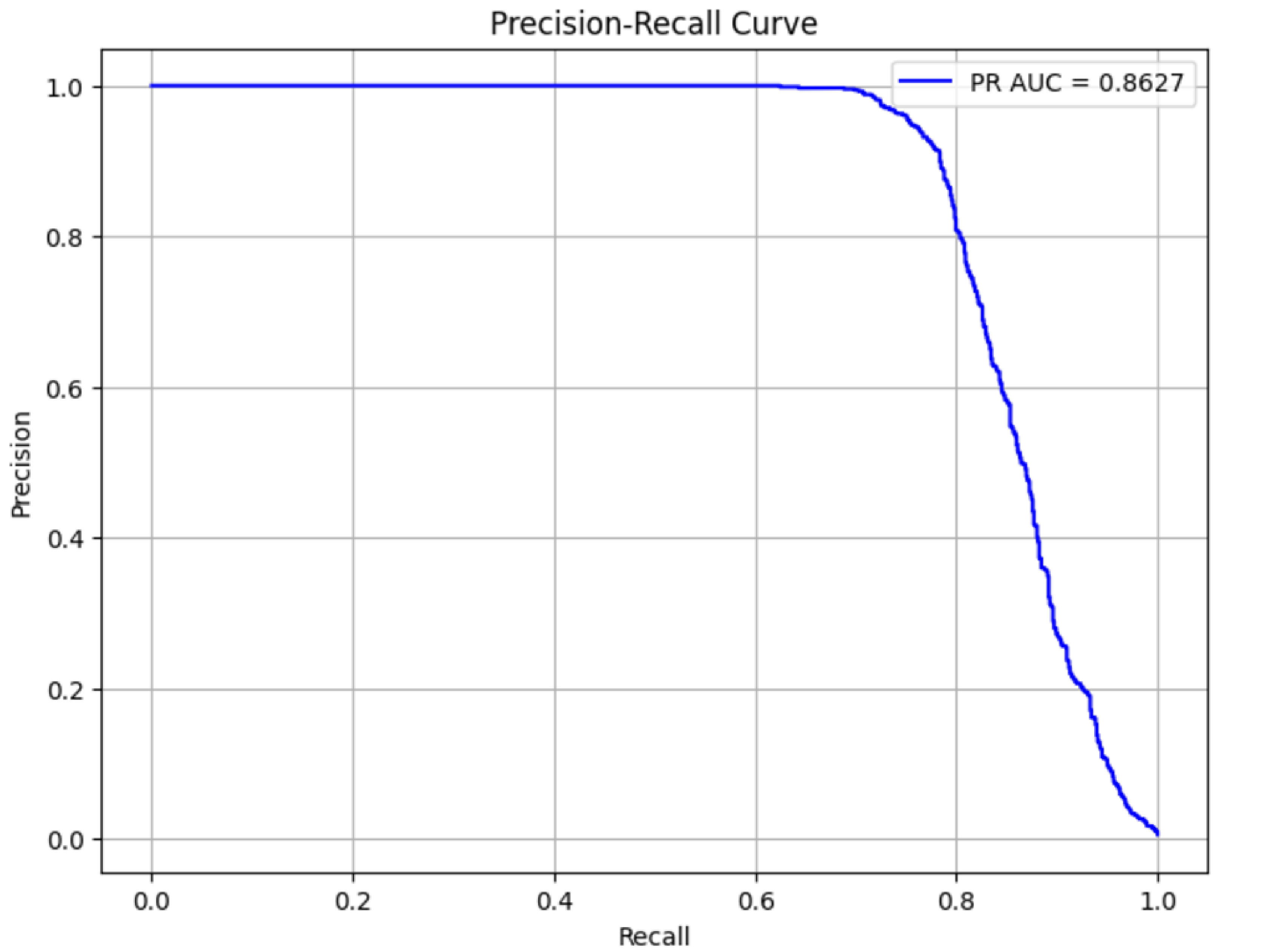
- Performed label encoding for the target variable and Split data into train and test sets (80:20).
- Applied SMOTETomek on the training data to balance the class distribution.
- Used Random Forest and XGBoost as base models, with Logistic Regression as the meta-model in a stacking classifier.
- Applied class weights in Random Forest, assigning higher weights to minority classes.
- Used sample weights in XGBoost to compute balanced weights for each class.
- Optimized parameters for both Random Forest and XGBoost using a hyperparameter grid.
- Cross-validated the stacking classifier and selected the best-performing model.
- Saved both task models in joblib files for deployment.

Visualization Overview



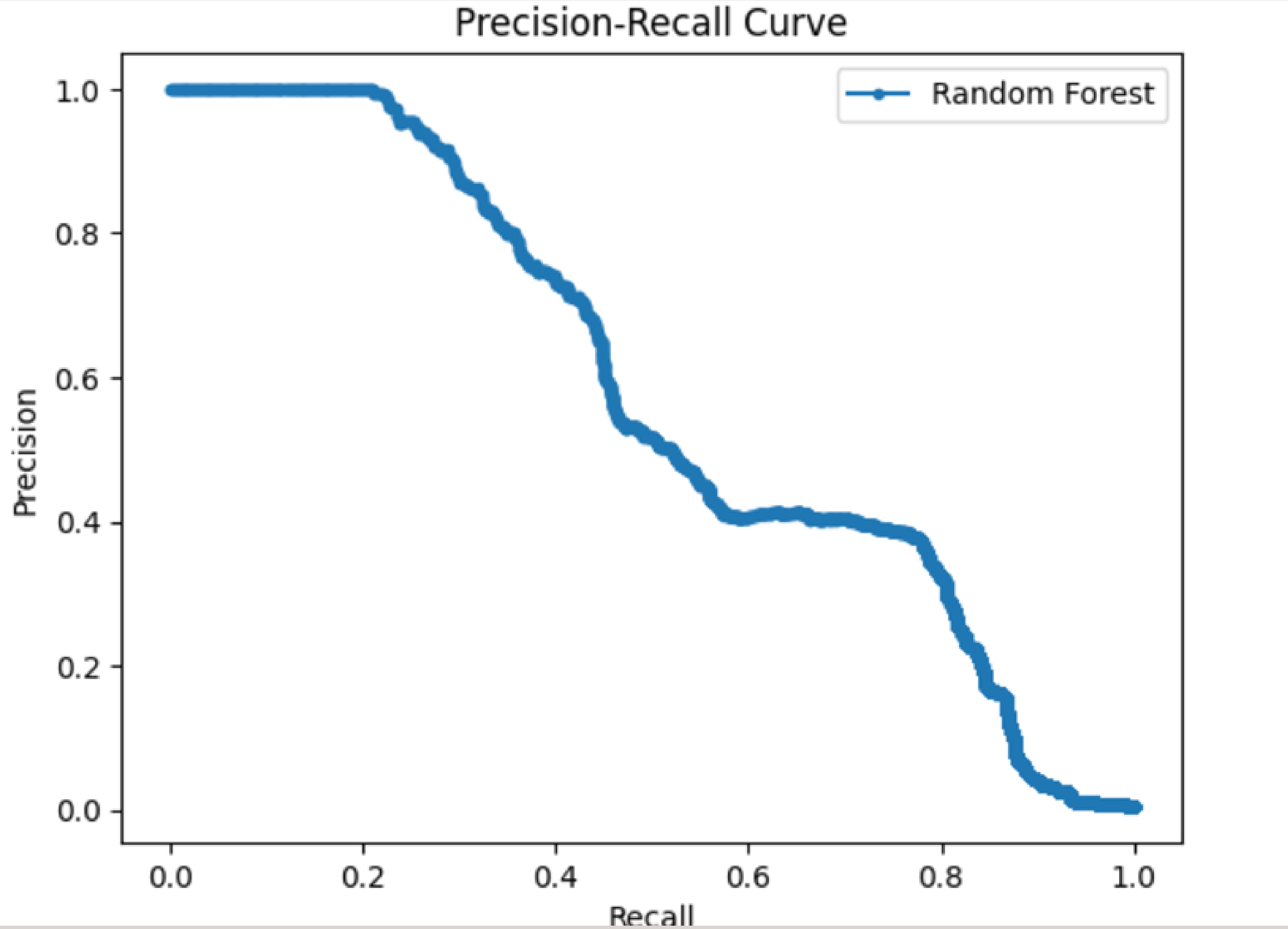
Results – Fraud detection

```
Test Accuracy (Best Model): 0.9984
Confusion Matrix (Best Model):
 [[174200     59]
 [   213    730]]

Classification Report (Best Model):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    174259
           1       0.93      0.77      0.84       943

    accuracy                           1.00    175202
   macro avg       0.96      0.89      0.92    175202
weighted avg       1.00      1.00      1.00    175202
```



Baseline Results

```
Test Set Evaluation:
Confusion Matrix:
 [[174259      0]
 [   804    139]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    174259
           1       1.00      0.15      0.26       943

    accuracy                           1.00    175202
   macro avg       1.00      0.57      0.63    175202
weighted avg       1.00      1.00      0.99    175202


ROC AUC Score: 0.9508566456128366

Precision-Recall AUC Score: 0.580732417507338
```



Baseline – Random Forest

Comparison Table

Model	Accuracy	Macro Average F1	F1 for Fraud
Random Forest	0.9956	0.63	0.26
XGBoost	0.9984	0.92	0.84

Results - HAR

```
Test Accuracy (Best Model): 0.8353
Confusion Matrix (Best Model):
 [[ 43   0   0   0   0   0   1   3   0   1   0]
 [  0  62   0   0   0   0   0   6   0   0   0]
 [  0   0   2   0   0   0   0   0   0   0   0]
 [  0   1   0  14   0   0   0   6   0   0   0]
 [  0   1   0   0   1   0   0   1   0   0   0]
 [  0   0   0   0   0 122   0   1   0   0   0]
 [  0   2   0   0   0   0  13   7   0   0   0]
 [  0  14   0   0   0   1   1  85   1   0   0]
 [  0   0   0   0   0   0   0   0  10   0   0]
 [  1   1   0   0   0   0   1   5   0   2   5]
 [  2   0   0   0   0   0   0   7   0   2   6]]
```

Classification Report (Best Model):

	precision	recall	f1-score	support
0	0.93	0.90	0.91	48
1	0.77	0.91	0.83	68
2	1.00	1.00	1.00	2
3	1.00	0.67	0.80	21
4	1.00	0.33	0.50	3
5	0.99	0.99	0.99	123
6	0.81	0.59	0.68	22
7	0.70	0.83	0.76	102
8	0.91	1.00	0.95	10
9	0.40	0.13	0.20	15
10	0.55	0.35	0.43	17
accuracy			0.84	431
macro avg	0.82	0.70	0.73	431
weighted avg	0.83	0.84	0.82	431

Baseline Results

	precision	recall	f1-score	support
Cycling	0.86	0.88	0.87	48
Football	0.62	0.96	0.75	68
Jogging	1.00	0.50	0.67	2
JumpRope	0.71	0.71	0.71	21
Pushups	1.00	0.33	0.50	3
Sitting	0.98	0.98	0.98	123
Swimming	0.54	0.68	0.60	22
Tennis	0.89	0.62	0.73	102
Walking	0.91	1.00	0.95	10
WalkingDownstairs	0.22	0.13	0.17	15
WalkingUpstairs	0.50	0.35	0.41	17
accuracy			0.79	431
macro avg	0.75	0.65	0.67	431
weighted avg	0.81	0.79	0.78	431

```
Confusion Matrix for XGBoost:
 [[ 42   2   0   0   0   0   2   1   0   1   0]
 [  0  65   0   1   0   0   2   0   0   0   0]
 [  0   0   1   1   0   0   0   0   0   0   0]
 [  0   2   0  15   0   0   0   2   0   1   1]
 [  0   0   0   0   1   0   1   0   1   0   0]
 [  0   0   0   0   0 121   1   1   0   0   0]
 [  1   3   0   0   0   0  15   2   0   1   0]
 [  0  26   0   3   0   2   6  63   0   1   1]
 [  0   0   0   0   0   0   0   0  10   0   0]
 [  3   5   0   0   0   0   1   0   0   2   4]
 [  3   2   0   1   0   0   0   2   0   3   6]]
```

Baseline – XGBoost

Comparison Table

Model	Accuracy	Macro Average F1-Score
XGBoost	0.7933	0.67
Stacking Classifier	0.8353	0.73

Observations

- For Task 1, during preprocessing, we observed that dropping features with a contribution less than 0.01 streamlines the model performance and improves efficiency.
- We selected XGBoost for its robustness in handling imbalanced datasets and its ability to provide feature-importance insights.
- For Task 2, We observed that a Stacking Classifier with Logistic Regression as the meta-model combining the strengths of base models like Random Forest and XGBoost mitigated biases in individual models, ensuring balanced generalization across activity classes.
- The meta-model dynamically weighed predictions, improving class-level performance consistency, especially for underrepresented categories.

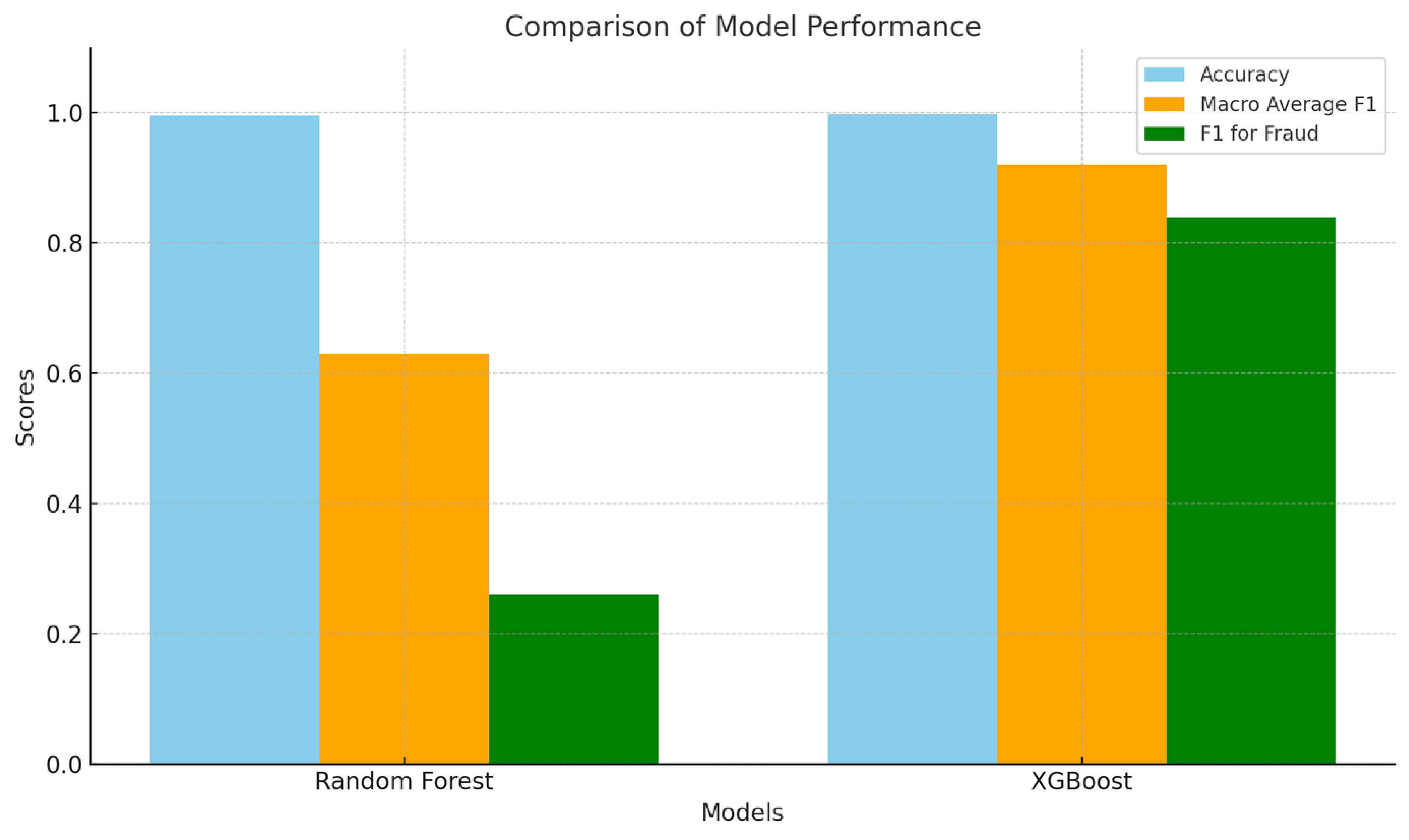
Observations

- Task 1
  - Model Performance: The XGBoost model outperformed the Random Forest baseline, achieving a test accuracy of 0.9984 and a macro F1-score of 0.92.
  - False Negatives Reduction: Recall for the minority class (fraudulent transactions) improved significantly to 0.77, reducing false negatives.
  - Precision: Fraud detection precision reached 0.93, indicating a balanced trade-off between false positives and true positives.
  - Baseline Comparison: The tuned XGBoost model greatly enhanced minority class recognition, with recall increasing from 0.16 (baseline) to 0.77.
- Task 2
  - Model Accuracy: The final stacking classifier achieved an accuracy of 0.84, slightly better than the baseline XGBoost model's 0.79 accuracy. But shows a better improvement in recall.
  - The ensemble approach ensured all precision, recall, and F1 scores were greater than 0 for all classes, showing better balance across the imbalanced dataset.

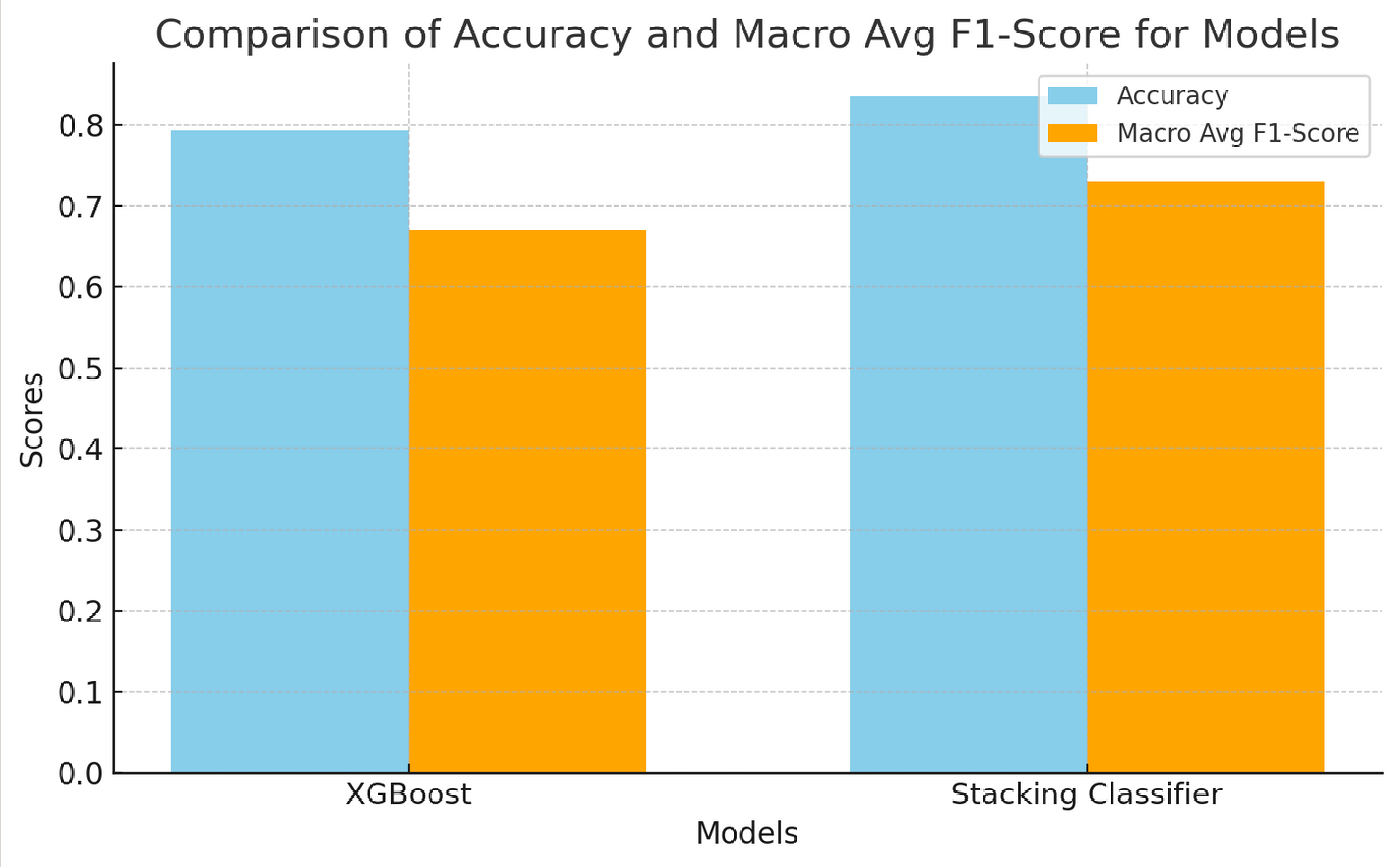
Observations

- Highlights
  - Task 1: Adjusting scale_pos_weight to 7 in XGBoost focused the model on the minority class, significantly reducing false negatives.
  - Task 1: Combining user and merchant location data embedded spatial relationships, enhancing the model's ability to detect fraudulent transactions.
  - Task 2: Applying SMOTETomek resolved class imbalance, improving the training data distribution and enabling better generalization.
  - Task 2: The stacking classifier, integrating Random Forest and XGBoost, achieved higher macro F1 scores and a balanced precision-recall profile for all classes.
- Success
  - Task 1: Achieved a high F1 score and precision for fraud detection, while effectively minimizing false negatives.
  - Task 2: The Stacking Classifier ensured improved precision, recall, and F1 scores for all classes, showcasing balanced and consistent performance.

Visualizations



Visualizations



Challenges

- Task 1:
  - We faced a challenge in deciding whether to balance the dataset or not.
  - Tested various balancing techniques:
    - SMOTE and ADASYN: While these methods ran successfully, they produced poor classification performance.
    - SMOTE ENN and SMOTE TOMEK: These methods handled outliers effectively but required high computational power and could not complete processing for the large dataset.
  - We ultimately opted for class weights to efficiently address the class imbalance issue.
  - No result even after 6 hours due to high computational power : ADASYN + TomekLinks

Challenges

- Task 2:
  - We encountered difficulties selecting a model capable of correctly classifying all classes, especially the minority ones:
    - Random Forest and certain XGBoost configurations resulted in precision and recall values of zero for some minority classes.
    - Evaluated ensemble methods:
      - Voting Classifier: Gave moderate results.
      - Stacking Classifier: We selected this approach as it delivered better performance and balanced classification.
  - We also faced issues with anonymous feature names in the test dataset:
    - The test dataset assigned the first column as the feature name, causing mismatched column names with the training dataset.
    - We resolved this by standardizing column names across training and test datasets.

Conclusion

- The project demonstrated the use of machine learning to solve real-world challenges in fraud detection and human activity recognition.
- In Task 1 we used XGBoost, which achieved a macro F1-score of 0.92, with enhanced recall for fraudulent transactions (0.77) through parameter tuning and feature engineering, including combining user and merchant locations.
- In Task 2 we used Logistic Regression as the meta-model and addressed class imbalance using SMOTETomek and class weights, resulting in balanced precision, recall, and F1 scores.
- Overall our assignment highlights the importance of preprocessing and ensemble learning in achieving robust performance and handling imbalanced datasets.

References

- Ramraj, S., Uzir, N., Sunil, R., & Banerjee, S. (2016). Experimenting XGBoost algorithm for prediction and classification of different datasets. International Journal of Control Theory and Applications, 9(40), 651-662.

- Novaković, Jasmina Dj, et al. "Evaluation of classification models in machine learning." Theory and Applications of Mathematics & Computer Science 7.1 (2017): 39.

- 7 SMOTE Variations for Oversampling - https://www.kdnuggets.com/2023/01/7-smote-variations-oversampling.html (Website that we referred to learn about SMOTE)

- Oversampling and Undersampling: ADASYN vs ENN - https://medium.com/quantyca/oversampling-and-undersampling-adasyn-vs-enn-60828a58db39 (Website that we used to refer about ADASYN)

Thank You