Classifying Commit Messages: A Case Study in Resampling Techniques

SeyedHamid Shekarforoush
Robert Green
Robert Dyer
Bowling Green State University

Introduction

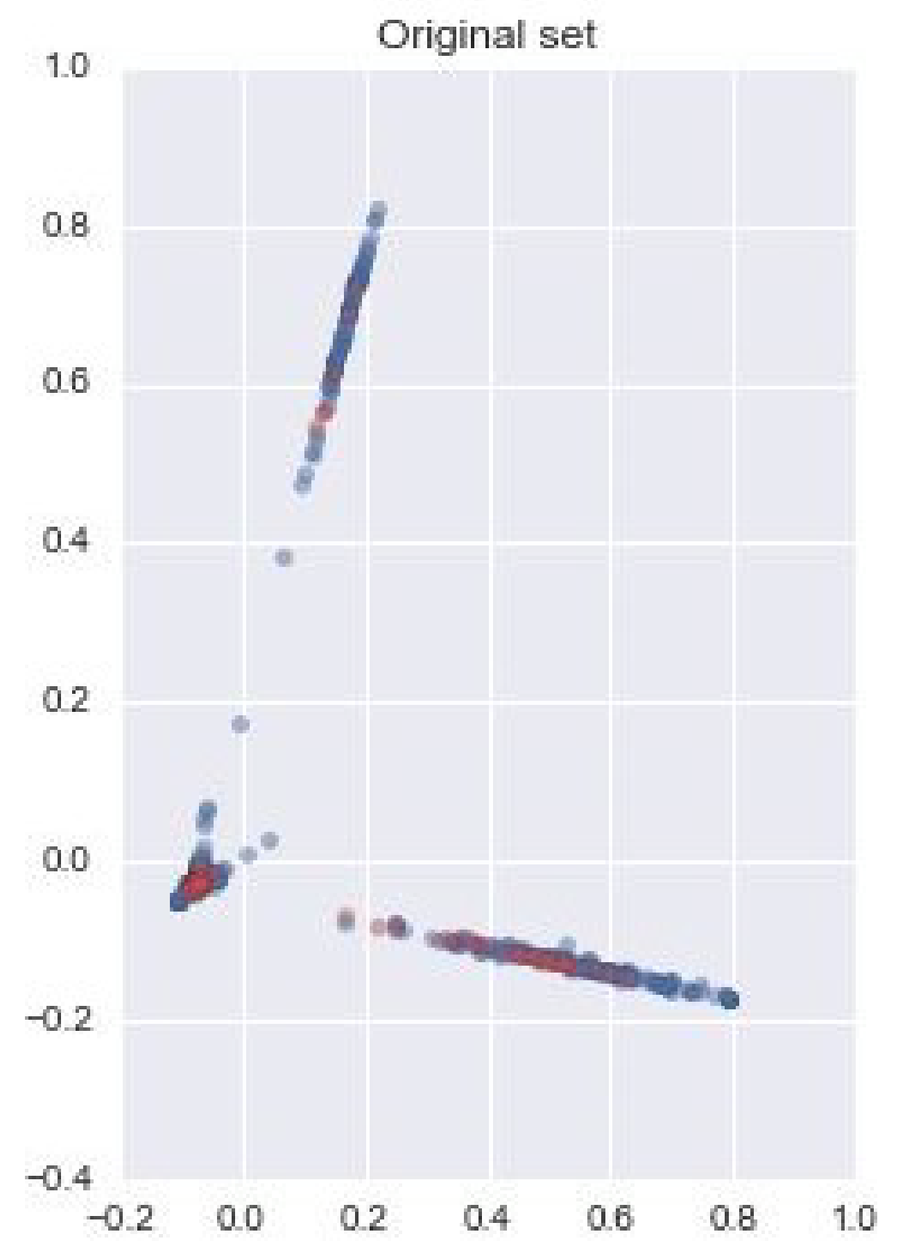
In practice, there are a variety of real-world datasets that have an imbalanced nature where one of two classes dominates the data. These datasets are generally difficult to classify using machine learning algorithms as the skewed nature of the data has a significant impact on the training process. In order to combat this difficulty, many methods of under sampling and over sampling have been proposed in order to generate comparable data sets that are more easily classifiable. This study applies multiple resampling techniques to a set of commit messages that have been extracted from multiple Github and Sourceforge projects in order to answer the question, "Do developers discuss design?" This dataset is highly imbalanced with less than 15% of all commit messages being classified as having to do with design. Results demonstrate that the combined use of resampling as coupled with various

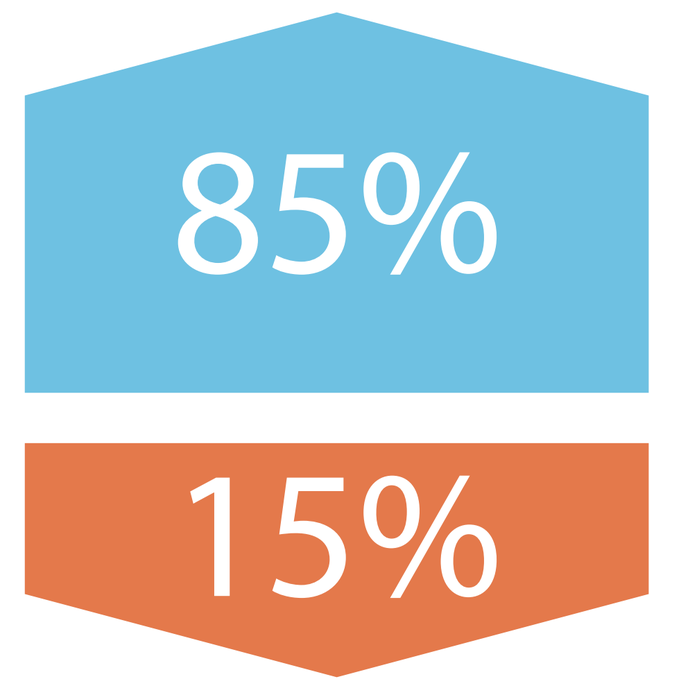
Classifiers:

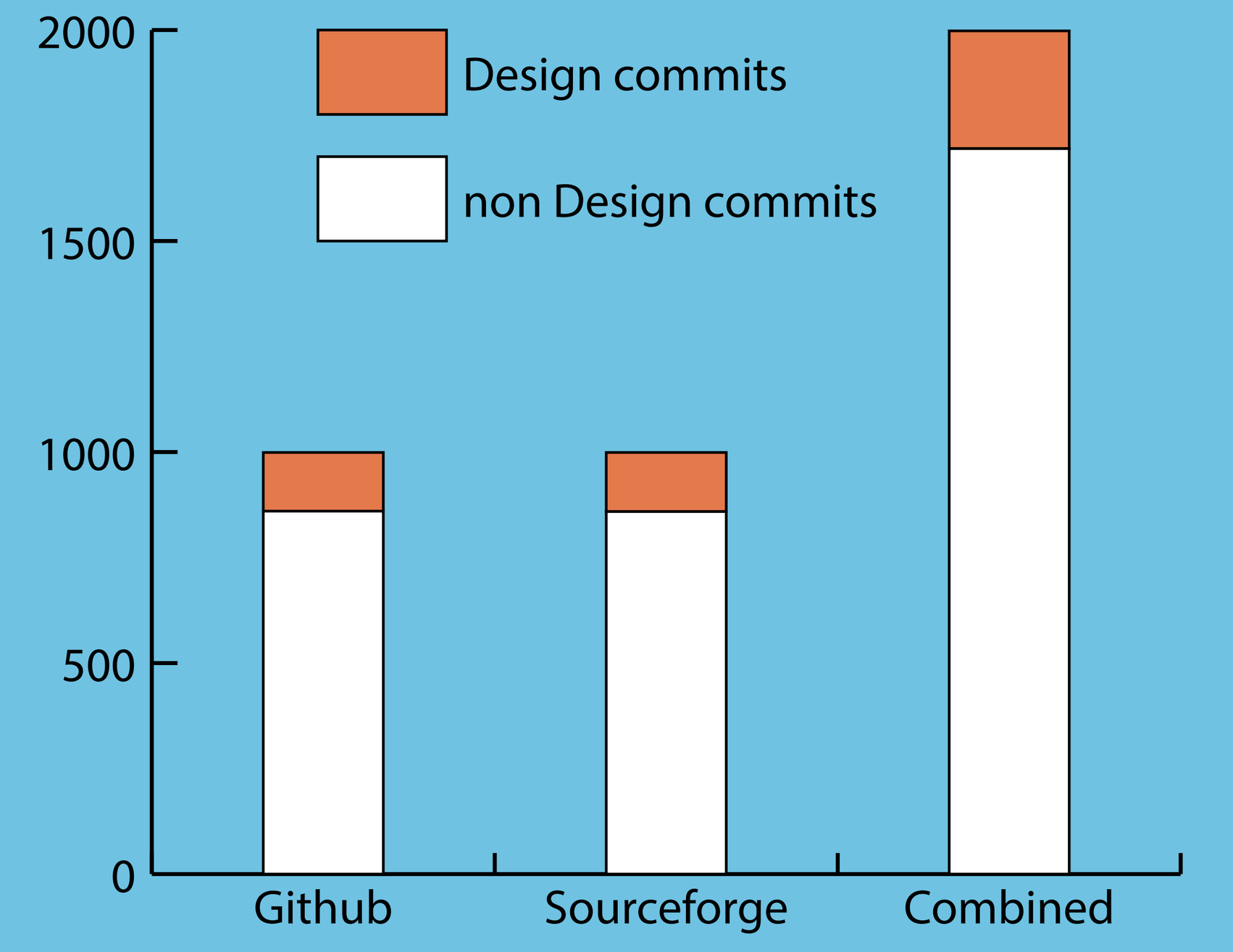
- Random Forest(RF)
- Decision Tree(DT)
- Support Vector Classification (SVC)
- Linear SVC (LSVC)
- Bernoulli Naive Bayes (BNB)
- Nearest Centroid (NC)

Original Dataset

- A set of commit messages that have been extracted from multiple Github and Sourceforge projects in order to answer the question, "Do developers discuss design?"
- Highly imbalanced
  - 15% design commits
  - 85% non Design commits





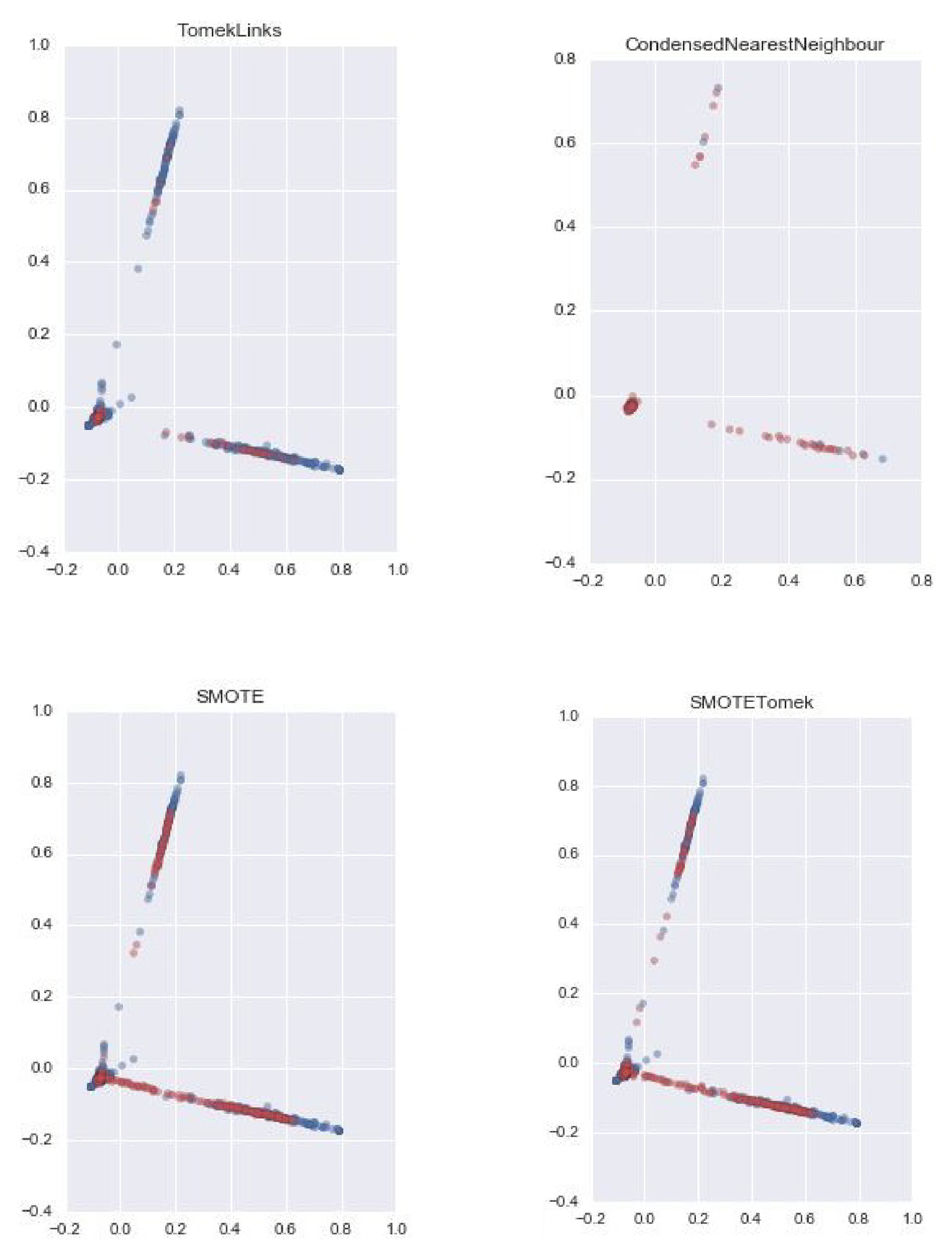


Resampling

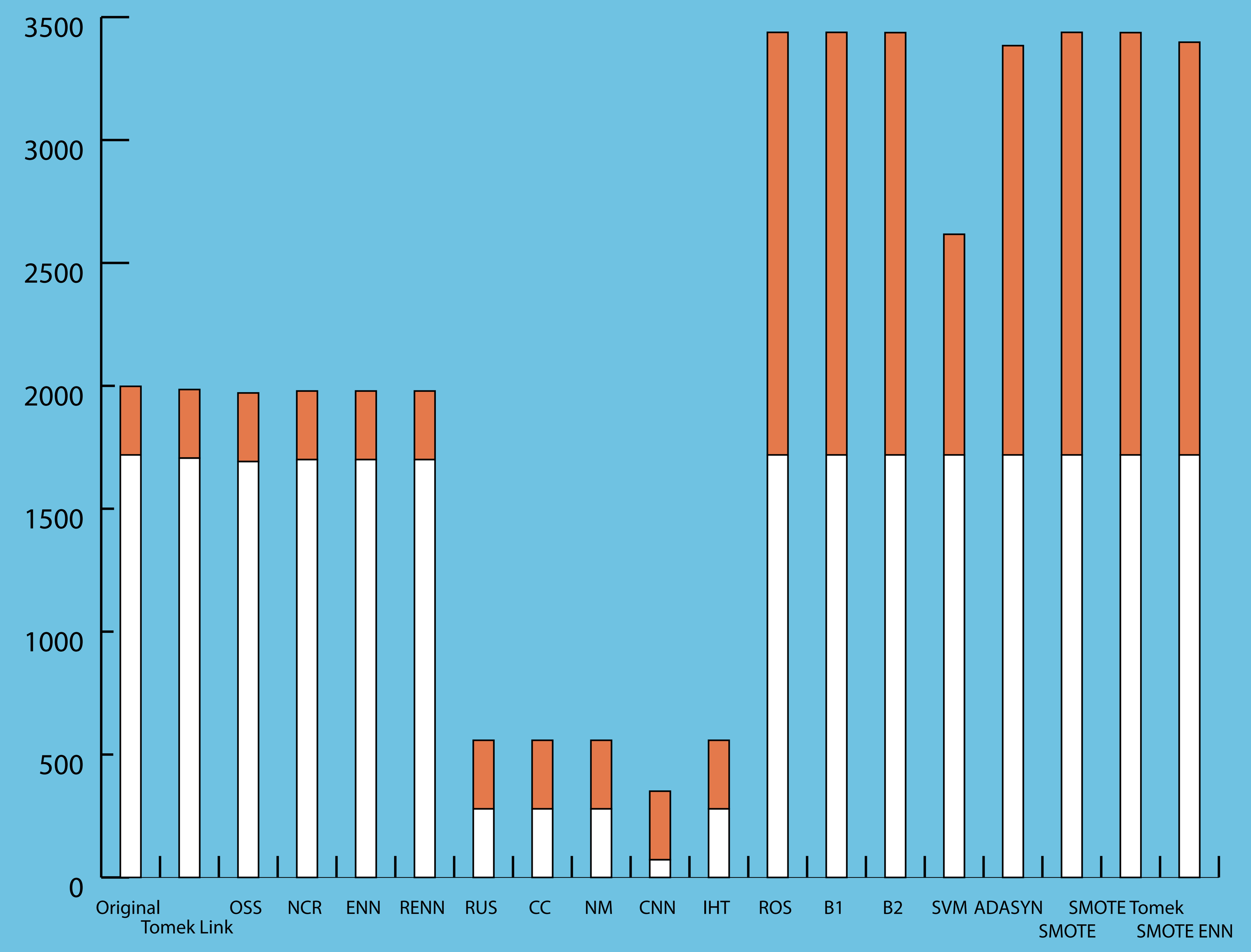
Under Samplers:
- Random Under Sampler (RUS)
- Tomek Links
- Cluster Centroids (CC)
- Near Miss (NM)
- Condensed Nearest Neighbour (CNN)
- One Sided Selection (OSS)
- Neighbourhood Cleaning Rule (NCC)
- Edited Nearest Neighbours (ENN)
- Instance Hardness Threshold (ITT)
- Repeated Edited Nearest Neighbours (RENN)

Over samplers:
- Random Over Sampler (ROS)
- SMOTE
- SMOTE borderline 1 (B1)
- SMOTE borderline 2 (B2)
- SMOTE svm (SVM)
- ADASYN

Hybrid:
- SMOTE Tomek
- SMOTE ENN



Extracting features before resampling using TF-IDF



Results

- True positive (TP)
- True negative (TN)
- False positive (FP)
- False negative (FN)

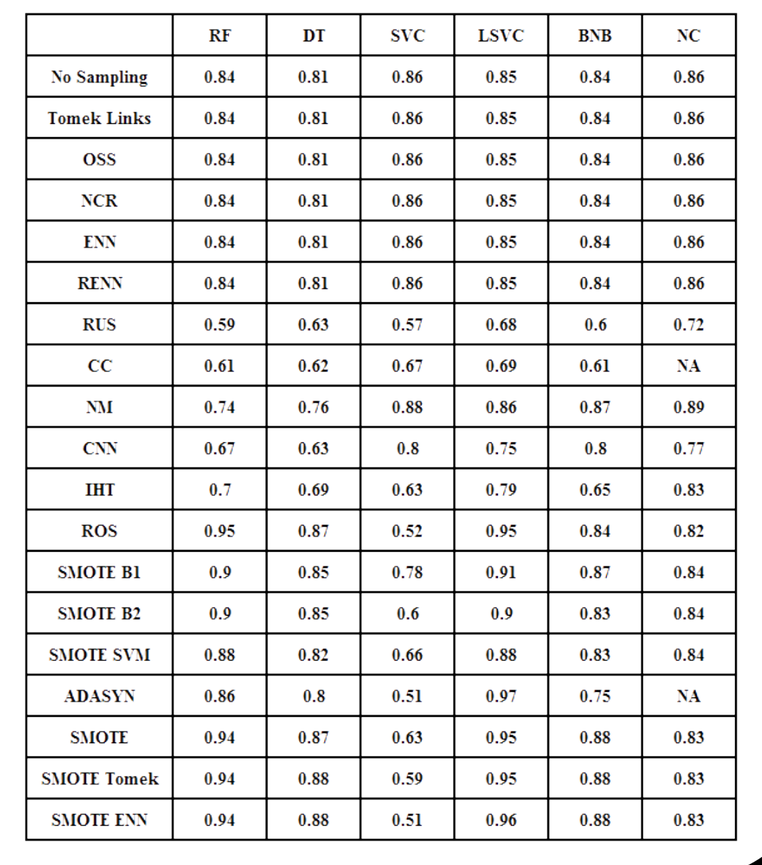
- Recall (R) : True positive rate or sensitivity
- Specificity (S) : True negative rate
- Precision (P) : Positive predictive value
- Accuracy (A)
- F1-score (F) : Harmonic mean of Precision and Recall

$$N = (T_P + T_N + F_P + F_N)$$
$$A = \frac{T_P + T_N}{N}$$
$$R = \frac{T_P}{T_P + F_N}$$
$$S = \frac{T_N}{T_N + F_P}$$
$$P = \frac{T_P}{T_P + F_P}$$
$$F = \frac{2PR}{P+R}$$
$$G_{mean} = \sqrt{RS}$$

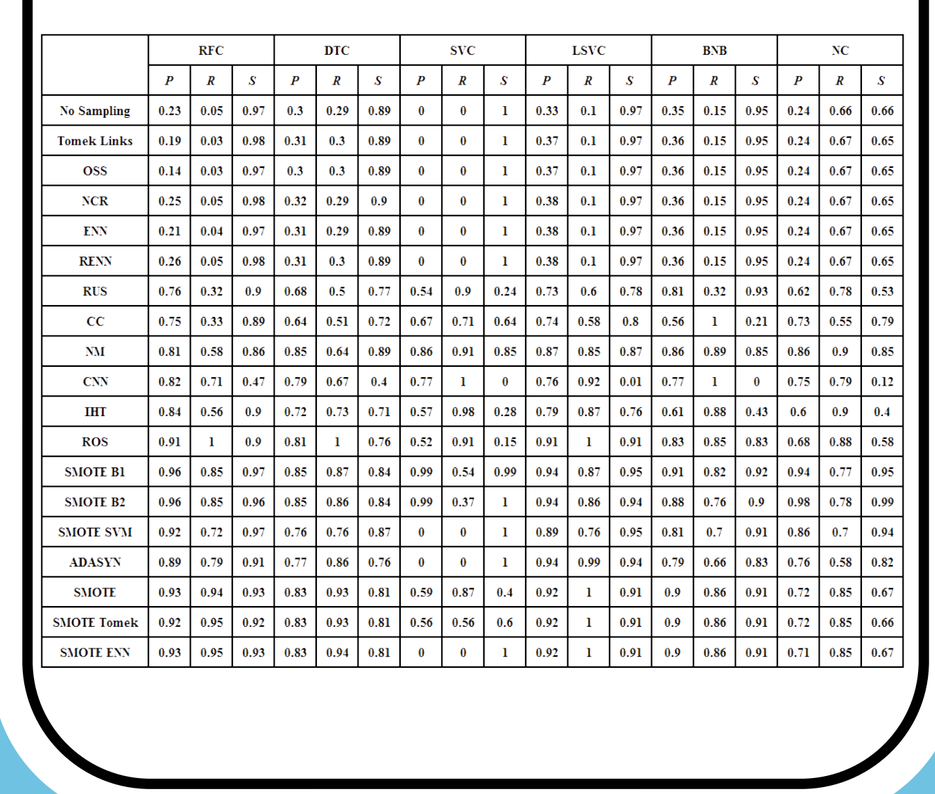
Experiment 1

Training set = SForge & GitHub
Testing set = GitHub & SForge

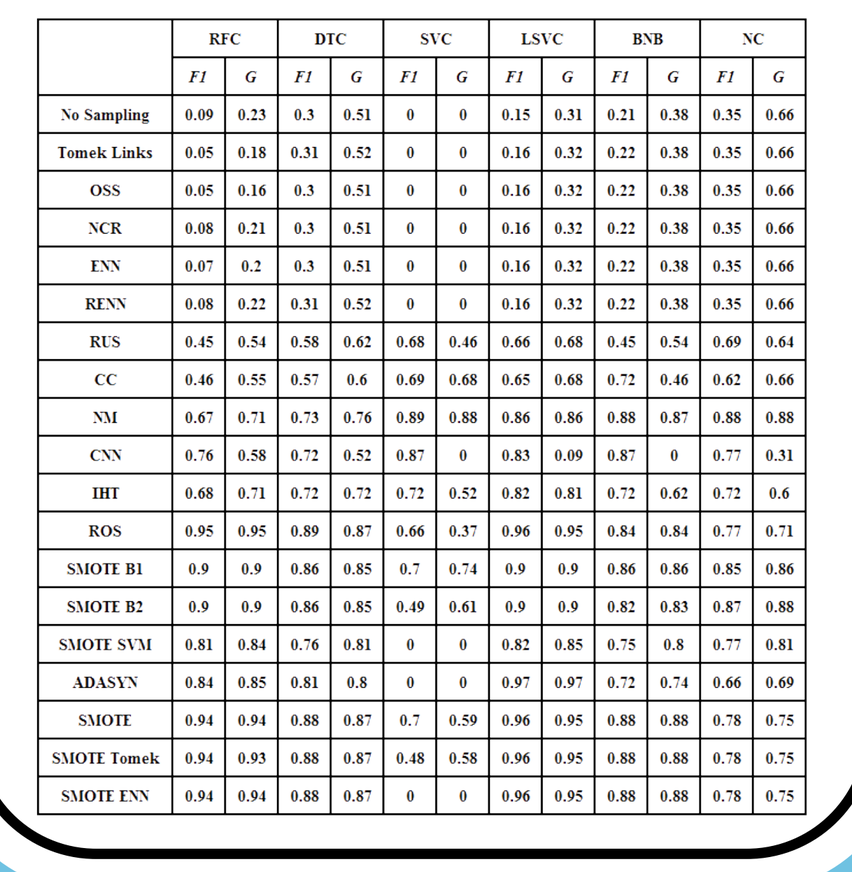
Accuracy



PRECISION (P), RECALL (R) SPECIFICITY (S)



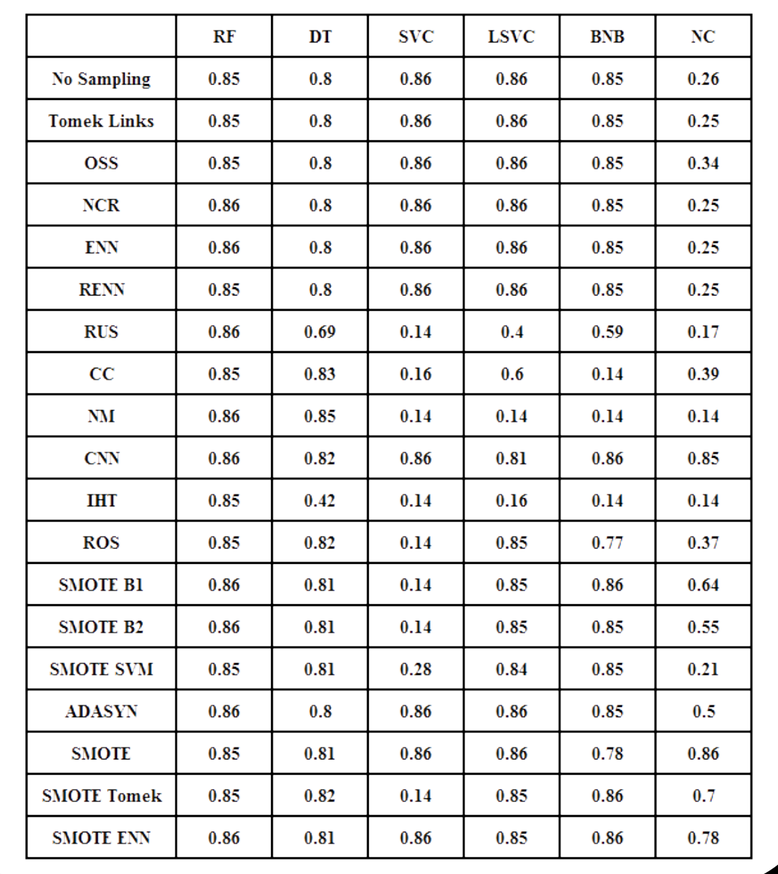
F1-SCORE (F1) G MEAN (G)



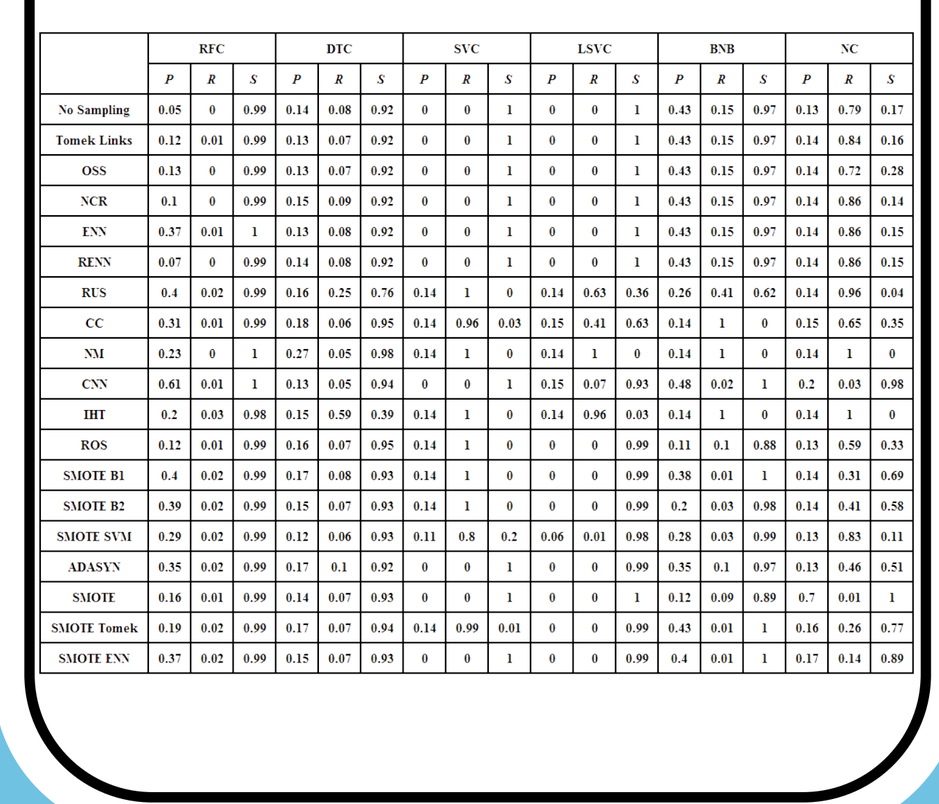
Experiment 2

Training set = SForge / GitHub
Testing set = GitHub / SForge

Accuracy



PRECISION (P), RECALL (R) SPECIFICITY (S)



F1-SCORE (F1) G MEAN (G)