**Appendix B**. Description of Proposed and Non-proposed Algorithms

| **Study** | **Proposed Algorithm** | **Description** | **Non-Proposed Algorithm** | **Description** |
| --- | --- | --- | --- | --- |
| [R01] | 2M-GWO (SVM, RF, GB, AB, KNN) | Two-Phase Modified Grey Wolf Optimizer combined with SVM (Support Vector Machine); RF (Random Forest); GB (Gradient Boosting); AB (AdaBoost); KNN (K-Nearest Neighbors) classifiers for optimization and classification | HHO, SSO, WO, JO, SCO | HHO: Harris Hawks Optimization, a metaheuristic inspired by the cooperative behavior of hawks to solve optimization problems; SSO: Social Spider Optimization, an optimization algorithm based on the communication and cooperation of social spiders; WO: Whale Optimization, an algorithm bioinspired by the hunting strategy of humpback whales; JO: Jellyfish Optimization, an optimization technique based on the movement patterns of jellyfish; SCO: Sand Cat Optimization, an algorithm inspired by the hunting strategy of desert cats to find optimal solutions. |
| [R02] | ANN, SVM | ANN: Artificial Neural Network, a basic neural network used for classification or regression; SVM: Support Vector Machine, a robust supervised classifier for binary classification problems | n/a | n/a |
| [R03] | LineFlowDP (Doc2Vec+R-GCN+GNNExplainer) | Defect prediction approach based on semantic code representation and neural graphs | CNN, DBN, BoW, Bi-LSTM, CodeT5, DeepBugs, IVDetect, LineVD, DeepLineDP, N-gram | CNN: Convolutional Neural Network, deep neural network used for automatic feature extraction in structured or unstructured data; DBN: Deep Belief Network, neural network based on layers of autoencoders to learn hierarchical data representations; BoW: Bag of Words, text or code representation model based on the frequency of appearance of words without considering the order; Bi-LSTM: Bidirectional Long Short-Term Memory, bidirectional recurrent neural network used to capture contextual information in sequences; CodeT5: Transformer Model, pre-trained transformer-based model for source code analysis and generation tasks; DeepBugs: DeepBugs Defect Detection, deep learning system designed to detect errors in source code; IVDetect: Invariant Violation Detection, a technique that seeks to detect violations of logical invariants in software programs; LineVD: Line-level Vulnerability Detector, automated system that identifies vulnerabilities in specific lines of code; DeepLineDP: Deep Line-based Defect Prediction, a deep learning-based model for predicting defects at the line of code level; N-gram: N-gram Language Model, a statistical model for processing sequences based on the frequency of occurrence of adjacent subsequences. |
| [R13] | CNN | Convolutional Neural Network, a neural network used for automatic feature extraction | n/a | n/a |
| [R22] | SDP-CMPOA (CMPOA+Bi-LSTM+Deep Maxout) | Software Defect Prediction using CMPOA optimized with Bi-LSTM and Deep Maxout activation | CNN, DBN, RNN, SVM, RF, GH+LSTM, FA, POA, PRO, AOA, COOT, BES | RNN: Recurrent Neural Network, a neural network designed to process sequential data using recurrent connections; SVM: Support Vector Machine, a robust supervised classifier for binary and multiclass classification problems; RF: Random Forest, an ensemble of decision trees used for classification and regression, robust to overfitting; GH+LSTM: Genetic Hybrid + Long Short-Term Memory, a combination of genetic optimization with an LSTM neural network to improve learning; FA: Firefly Algorithm, an optimization algorithm inspired by the luminous behavior of fireflies to solve complex problems; POA: Pelican Optimization Algorithm, an optimization technique based on the collective behavior of pelicans; PRO: Progressive Optimization, an optimization approach that iteratively adjusts parameters to improve results; AOA: Arithmetic Optimization Algorithm, a metaheuristic based on arithmetic operations to explore and exploit the search space; COOT: Coot Bird Optimization, an optimization algorithm inspired by the movements of coot-type aquatic birds; BES: Bacterial Foraging Optimization, a metaheuristic inspired by the foraging strategy of bacteria. |
| [R24] | DT, NB, RF, LSVM | DT: Decision Tree, classifier based on decision trees, NB: Naïve Bayes, probabilistic classifier based on Bayes theory, RF: Random Forest, ensemble of decision trees for classification and regression, LSVM: Linear Support Vector Machine, linear version of SVM | n/a | n/a |
| [R10] | PoPL(Hybrid) | Paired Learner Approach, a hybrid technique for handling concept drift in defect prediction | n/a | n/a |
| [R11] | bGWO (ANN, DT, KNN, NB, SVM) | Binary Grey Wolf Optimizer combined with multiple classifiers | ACO | Ant Colony Optimization, técnica metaheurística basada en el comportamiento colectivo de las hormigas para resolver problemas de optimización de rutas o combinatorios |
| [R12] | FMR, FMRT | Fuzzy Min-Max Regression and its variant for prediction | NB, RF, ACN, ACF | NB: Naïve Bayes, a simple probabilistic classifier based on the application of Bayes' theorem with independence between attributes; ACN: Artificial Cognitive Network, an artificial network model inspired by cognitive systems for classification or pattern analysis; ACF: Artificial Cooperative Framework, an artificial cooperative framework designed to improve accuracy in prediction or classification tasks. |
| [R15] | LM, BP, BR, BR+NN | LM: Linear Model, linear regression model, BP: Backpropagation, training algorithm for neural networks, BR: Bayesian Regularization, technique to avoid overfitting in neural networks, BR+NN: Bayesian Regularized Neural Network, Bayesian regularized neural network | SVM, DT, KNN, NN | DT: Decision Tree, a classification or regression model based on a decision tree structure; KNN: K-Nearest Neighbors, a classifier based on the similarity between instances in the feature space; NN: Neural Network, an artificial neural network used for supervised or unsupervised learning in various tasks. |
| [R16] | DEPT-C, DEPT-M1, DEPT-M2, DEPT-D1, DEPT-D2 | Variants of a specific DEPT approach to prioritization or prediction in software testing | DE, GS, RS | DE: Differential Evolution, an evolutionary optimization algorithm used to solve continuous and nonlinear problems; GS: Grid Search, a systematic search method for hyperparameter optimization in machine learning models; RS: Random Search, a hyperparameter optimization technique based on the random selection of combinations. |
| [R42] | MLP | Multilayer Perceptron, a neural network with multiple hidden layers. | n/a |  |
| [R18] | C4.5 +ADB | C4.5 Decision Tree Algorithm Combined with AdaBoost to Improve Accuracy. | ERUS, NB, NB+Log, RF, DNC, SMT+NB, RUS+NB, SMTBoost, RUSBoost | ERUS: Ensemble Random Under Sampling, class balancing method based on combined random undersampling in ensemble; NB+Log: Naïve Bayes + Logistic Regression, hybrid approach that combines Naïve Bayes probabilities with a logistic classifier; DNC: Dynamic Nearest Centroid, classifier based on dynamic centroids to improve accuracy; SMT+NB: Synthetic Minority Technique + Naïve Bayes, combination of class balancing with Bayesian classification; RUS+NB: Random Under Sampling + Naïve Bayes, majority class reduction technique combined with Naïve Bayes; SMTBoost: Synthetic Minority Oversampling Technique Boosting, balancing method combined with boosting to improve classification; RUSBoost: Random Under Sampling Boosting, ensemble method based on undersampling and boosting to improve prediction. |
| [R28] | KPCA+ELM | Kernel Principal Component Analysis combined with Extreme Learning Machine | SVM, NB, LR, MLP, PCA+ELM | LR: Logistic Regression, a statistical model used for binary classification using the sigmoid function; MLP: Multilayer Perceptron, an artificial neural network with one or more hidden layers for classification or regression; PCA+ELM: Principal Component Analysis + Extreme Learning Machine, a hybrid approach that reduces dimensionality and applies ELM for classification. |
| [R47] | rejoELM, IrejoELM | Improved variants of the Extreme Learning Machine applying its own techniques. | rejoNB, rejoRBF | rejoNB: Re-joined Naïve Bayes, an improved variant of Naïve Bayes for classification; rejoRBF: Re-joined Radial Basis Function, a variant based on RBF for classification or regression tasks. |
| [R29] | WPA-PSO+DNN, WPA-PSO+self-encoding | Whale + Particle Swarm Optimization combined with Deep Neural Networks or Autoencoders. | Grid, Random, PSO, WPA | Grid: Grid Search, an exhaustive search technique for hyperparameter optimization; Random: Random Search, a random parameter optimization strategy; PSO: Particle Swarm Optimization, an optimization algorithm inspired by the behavior of particle swarms; WPA: Whale Particle Algorithm, a metaheuristic that combines whale and particle optimization strategies. |
| [R30] | ACO | Ant Colony Optimization, a technique inspired by ant behavior for optimization. | NB, J48, RF | J48: J48 Decision Tree, implementation of the C4.5 algorithm in WEKA software for classification. |
| [R41] | DP+GCNN | Defect Prediction using Graph Convolutional Neural Network | LRC, RFC, DBN, CNN, SEML, MPT, DP-T, CSEM | LRC: Logistic Regression Classifier, a variant of logistic regression applied to classification tasks; RFC: Random Forest Classifier, an ensemble of decision trees for robust classification; SEML: Software Engineering Machine Learning, an approach that applies machine learning techniques to software engineering; MPT: Modified Particle Tree, a tree-based algorithm for optimization; DP-T: Defect Prediction - Tree, a tree-based approach for defect prediction; CSEM: Code Structural Embedding Model, a model that uses structural code embeddings for prediction or classification. |
| [R44] | RNNBDL | Recurrent Neural Network with Bayesian Deep Learning | LSTM, BiLSTM, CNN, SVM, NB, KNN, KStar, Random Tree | LSTM: Long Short-Term Memory, a recurrent neural network specialized in learning long-term dependencies in sequences; BiLSTM: Bidirectional Long Short-Term Memory, a bidirectional version of LSTM that captures past and future context in sequences; KStar: KStar Instance-Based Classifier, a nearest-neighbor classifier with a distance function based on transformations; Random Tree: Random Tree Classifier, a classifier based on randomly generated decision trees. |
| [R50] | Naïve Bayes (GaussianNB) | Naïve Bayes variant using Gaussian distribution | n/a | n/a |
| [R51] | Stacking+MLP (J48,RF,SMO,IBK,BN)+BF,GS,GA,PSO,RS,LFS | Stacking ensemble of multiple classifiers and meta-heuristics | n/a | n/a |
| [R53] | TS-ELA (ELA+IG+SMOTE+INFFC)+(BaG, RaF, AdB, LtB, MtB, RaB, StK, StC, VoT, DaG, DeC, GrD, RoF) | Hybrid technique that combines multiple balancing, selection and induction techniques | DTa, DSt | DTa: Decision Tree (Adaptive), a variant of the adaptive decision tree for classification; DSt: Decision Stump, a single-split decision tree, used in ensemble methods. |
| [R55] | CBA2 | Classification Based on Associations version 2 | C4.5, CART, ADT, RIPPER, DT | C4.5: C4.5 Decision Tree, a classic decision tree algorithm used in classification; CART: Classification and Regression Tree, a tree technique for classification or regression tasks; ADT: Alternating Decision Tree, a tree-based algorithm with alternating prediction and decision nodes; RIPPER: Repeated Incremental Pruning to Produce Error Reduction, a rule-based algorithm for classification. |
| [R57] | HyGRAR (MLP, RBFN, GRANUM) | Hybrid of MLP, radial basis networks and GRAR algorithm for classification. | SOM, KMeans-QT, XMeans, EM, GP, MLR, BLR, LR, ANN, SVM, CCN, GMDH, GEP, SCART, FDT-O, FDT-E, DT-Weka, BayesNet, MLP, RBFN, ADTree, DTbl, CODEP-Log, CODEP-Bayes | SOM: Self-Organizing Map, unsupervised neural network used for clustering and data visualization; KMeans-QT: K-Means Quality Threshold, a variant of the K-Means algorithm with quality thresholds for clusters; XMeans: Extended K-Means, an extended version of K-Means that automatically optimizes the number of clusters; EM: Expectation Maximization, an iterative statistical technique for parameter estimation in mixture models; GP: Genetic Programming, an evolutionary programming technique for solving optimization or learning problems; MLR: Multiple Linear Regression, a statistical model for predicting a continuous variable using multiple predictors; BLR: Bayesian Linear Regression, a linear regression under a Bayesian approach to incorporate uncertainty; ANN: Artificial Neural Network, an artificial neural network used in classification, regression, or prediction tasks; CCN: Convolutional Capsule Network, a convolutional capsule network for pattern recognition; GMDH: Group Method of Data Handling, a technique based on polynomial networks for predictive modeling; GEP: Gene Expression Programming, an evolutionary technique based on genetic programming for symbolic modeling; SCART: Soft Classification and Regression Tree, a decision tree variant that allows fuzzy or soft classification; FDT-O: Fuzzy Decision Tree - Option, a decision tree variant with the incorporation of fuzzy logic; FDT-E: Fuzzy Decision Tree - Enhanced, an improved version of fuzzy decision trees; DT-Weka: Decision Tree Weka, an implementation of decision trees within the WEKA platform; BayesNet: Bayesian Network, a probabilistic classifier based on Bayesian networks; RBFN: Radial Basis Function Network, a neural network based on radial basis functions for classification or regression; ADTree: Alternating Decision Tree, a technique based on alternating decision and prediction trees; DTbl: Decision Table, a simple classifier based on decision tables; CODEP-Log: Code Execution Prediction - Logistic Regression, a defect prediction approach using logistic regression; CODEP-Bayes: Code Execution Prediction - Naïve Bayes, a prediction approach based on Naïve Bayes. |
| [R65] | ME-SFP+[DT], ME-SFP+[MLP] | Multiple Ensemble with Selective Feature Pruning with base classifiers. | Bagging+DT, Bagging+MLP, Boosting+DT, Boosting+MLP, Stacking+DT, Stacking+MLP, Indi+DT, Indi+MLP, Classic+ME | Bagging+DT: Bootstrap Aggregating + Decision Tree, an ensemble method that uses decision trees to improve accuracy; Bagging+MLP: Bagging + Multilayer Perceptron, an ensemble method that applies MLP networks; Boosting+DT: Boosting + Decision Tree, an ensemble method where the weak classifiers are decision trees; Boosting+MLP: Boosting + MLP, a combination of boosting and MLP neural networks; Stacking+DT: Stacking + Decision Tree, a stacked ensemble that uses decision trees; Stacking+MLP: Stacking + MLP, a stacked ensemble with MLP networks; Indi+DT: Individual Decision Tree, an approach based on individual decision trees within a comparison or ensemble scheme; Indi+MLP: Individual MLP, an MLP neural network used independently in experiments or ensembles; Classic+ME: Classic Multiple Ensemble, a classic configuration of ensemble methods. |
| [R66] | AST n-gram+J48, AST n-gram+Logistic, AST n-gram+Naive Bayes | Approach based on AST n-gram feature extraction combined with different classifiers | n/a | n/a |
| [R07] | IECGA (RF+SVM+NB+GA) | Improved Evolutionary Cooperative Genetic Algorithm with Multiple Classifiers | RF, SVM, NB | NB: Naïve Bayes, simple probabilistic classifier based on Bayes theory. |
| [R09] | VESDP (RF+SVM+NB+ANN) | Variant Ensemble Software Defect Prediction | RF, SVM, NB, ANN | ANN: Artificial Neural Network, artificial neural network used in classification or regression tasks |
| [R17] | MLP, BN, Lazy IBK, Rule ZeroR, J48, LR, RF, DStump, SVM | BN: Bayesian Network, classifier based on Bayesian networks, Lazy IBK: Instance-Based K Nearest Neighbors, Rule ZeroR: Trivial classifier without predictor variables, J48: Implementation of C4.5 in WEKA, LR: Logistic Regression, logistic regression, DStump: Decision Stump, decision tree of depth 1 | n/a | n/a |
| [R19] | CONVSDP (CNN), DNNSDP (DNN) | Convolutional Neural Network applied to defect prediction., Deep Neural Network applied to defect prediction | RF, DT, NB, SVM | RF: Random Forest, an ensemble of decision trees that improves accuracy and overfitting control. |
| [R21] | ISDPS (NB+SVM+DT) | Intelligent Software Defect Prediction System combining classifiers | NB, SVM, DT, Bagging, Vouting, Stacking | Bagging: Bootstrap Aggregating, an ensemble technique that improves the stability of classifiers; Vouting: Voting Ensemble, an ensemble method that combines the predictions of multiple classifiers using voting; Stacking: Stacked Generalization, an ensemble technique that combines multiple models using a meta-classifier. |
| [R33] | 2SSEBA (2SSSA, ELM, Bagging Ensemble) | Two-Stage Salp Swarm Algorithm + ELM with Ensemble | ELM , SSA+ELM, 2SSSA+ELM, KPWE, SEBA | ELM: Extreme Learning Machine, a single-layer, fast-learning neural network.  SSA+ELM: Salp Swarm Algorithm + ELM, a combination of the bio-inspired SSA algorithm and ELM; 2SSSA+ELM: Two-Stage Salp Swarm Algorithm + ELM, an improved version of the SSA approach combined with ELM; KPWE: Kernel Principal Wavelet Ensemble, a method that combines wavelet transforms with kernel techniques for classification; SEBA: Swarm Enhanced Bagging Algorithm, an enhanced ensemble technique using swarm algorithms |
| [R38] | MODL-SBP (CNN-BiLSTM+CQGOA) | Hybrid model combining CNN, BiLSTM and CQGOA optimization | SVM-RBF, KNN+EM, NB, DT, LDA, AdaBoost, | SVM-RBF: Support Vector Machine with Radial Basis Function, an SVM using RBF kernels for nonlinear separation; KNN+EM: K-Nearest Neighbors + Expectation Maximization, a combination of KNN classification with an EM algorithm for clustering or imputation; LDA: Linear Discriminant Analysis, a statistical technique for dimensionality reduction and classification; AdaBoost: Adaptive Boosting, an ensemble technique that combines weak classifiers to improve accuracy |
| [R46] | MVFS (MVFS+NB, MVFS+J48, MVFS+IBK) | Multiple View Feature Selection applied to different classifiers | IG, CO, RF, SY | IG: Information Gain, a statistical measure used to select attributes in decision models; CO: Cut-off Optimization, a technique that adjusts cutoff points in classification models; SY: Symbolic Learning, a symbolic learning-based approach for classification or pattern discovery tasks. |
| [R06] | HFEDL(CNN, BiLSTM+Attention) | Hierarchical Feature Ensemble Deep Learning | n/a | n/a |
| [R40] | KELM+WSO | Kernel Extreme Learning Machine combined with Weight Swarm Optimization | SNB, FLDA, GA+DT, CGenProg | SNB: Selective Naïve Bayes, an improved version of Naïve Bayes based on the selection of relevant attributes; FLDA: Fisher Linear Discriminant Analysis, a dimensionality reduction technique optimized for class separation; GA+DT: Genetic Algorithm + Decision Tree, a combination of genetic algorithms with decision trees for parameter selection or optimization; CGenProg: Code Genetic Programming, a genetic programming application for automatic code improvement or repair. |
| [R49] | CCFT+CNN | Combination of Code Feature Transformation + CNN | RF, DBN, CNN, RNN, CBIL, SMO | CBIL: Classifier Based Incremental Learning, an incremental approach to supervised learning based on classifiers; SMO: Sequential Minimal Optimization, an efficient algorithm for training SVMs |
| [R58] | KTC (IDR+NB, IDR+SVM, IDR+KNN, IDR+J48) | Keyword Token Clustering combined with different classifiers | NB, KNN, SVM, J48 | Set of standard classifiers (Naïve Bayes, K-Nearest Neighbors, Support Vector Machine, J48 Decision Tree) applied in various classification tasks. |
| [R45] | Flakify (CodeBERT) | CodeBERT-based model for unstable test detection | FlakeFlagger | FlakeFlagger: Flaky Test Flagging Model, a model designed to identify unstable tests or flakiness in software testing. |
| [R34] | SVM+MLP+RF | SVM: Support Vector Machine + MLP: Multilayer Perceptron + RF: Random Forest, hybrid ensemble that combines SVM, MLP neural networks and Random Forest to improve accuracy. | SVM, ANN, RF | SVM: Support Vector Machine, a robust classifier widely used for supervised classification problems; ANN: Artificial Neural Network, an artificial neural network for classification, regression, or prediction tasks; RF: Random Forest, an ensemble technique based on multiple decision trees to improve accuracy and robustness. |
| [R56] | FRBS | Fuzzy Rule-Based System, a system based on fuzzy rules used for classification or decision making | C4.5, RF, NB | C4.5: Decision Tree, a classic decision tree algorithm used for classification; NB: Naïve Bayes, a simple probabilistic classifier based on the application of Bayes' theorem. |
| [R04] | XCSF-ER | Extended Classifier System with Function Approximation - Enhanced Rule, extended rule-based system with approximation and enhancement capabilities | ANN, RS, XCSF | RS: Random Search, a hyperparameter optimization technique based on random selection; XCSF: Extended Classifier System with Function Approximation, a rule-based evolutionary learning system. |
| [R60] | KNN | K-Nearest Neighbors, a classifier based on the similarity between nearby instances in the feature space | LR, LDA, CART, NB, SVM | LR: Logistic Regression, a statistical model for binary or multiclass classification; LDA: Linear Discriminant Analysis, a method for dimensionality reduction and supervised classification; CART: Classification and Regression Trees, a tree technique used in classification and regression. |
| [R64] | AFSA | Artificial Fish Swarm Algorithm, a bio-inspired metaheuristic based on fish swarm behavior for optimization | GA, K-means Clustering, NSGA-II, IA | GA: Genetic Algorithm, an evolutionary algorithm based on natural selection for solving complex problems; K-means Clustering: K-means Clustering Algorithm, an unsupervised technique for grouping data into distance-based clusters; NSGA-II: Non-dominated Sorting Genetic Algorithm II, a widely used multi-objective evolutionary algorithm; IA: Intelligent Agent, a computational system that perceives its environment and makes autonomous decisions. |
| [R35] | T5 (YOLOv5) | Text-to-Text Transfer Transformer + You Only Look Once v5, combining language processing with object detection in images | n/a |  |
| [R39] | EfficientDet, DETR, T5, GPT-2 | EfficientDet: EfficientDet Object Detector, a deep learning model optimized for object detection in images; DETR: Detection Transformer, a transformer-based model for object detection in computer vision; T5: Text-to-Text Transfer Transformer, a deep learning model for translation, summarization, and other NLP tasks; GPT-2: Generative Pre-trained Transformer 2, a transformer-based autoregressive language model. | n/a |  |
| [R14] | MFO | Moth Flame Optimization, a bio-inspired optimization algorithm based on the behavior of moths around flames | FA, ACO | FA: Firefly Algorithm, a metaheuristic inspired by the light behavior of fireflies; ACO: Ant Colony Optimization, a bio-inspired metaheuristic based on cooperative pathfinding in ants. |
| [R48] | IFROWANN av-w₁ | Improved Fuzzy Rough Weighted Artificial Neural Network, a neural network with fuzzy weighting and approximation | EUSBoost, SMOTE+C4.5, CS+SVM, CS+C4.5 | EUSBoost: Evolutionary Undersampling Boosting, an ensemble technique that balances classes using evolutionary undersampling; SMOTE+C4.5: Synthetic Minority Oversampling + C4.5, a hybrid technique for class balancing and classification; CS+SVM: Cost-Sensitive SVM, a cost-sensitive version of the SVM classifier; CS+C4.5: Cost-Sensitive C4.5, a cost-sensitive version applied to C4.5 trees. |
| [R32] | NN (LSTM+MLP) | Neural Network (LSTM + Multilayer Perceptron), a hybrid neural network that combines LSTM and MLP networks | Hierarchical Clustering | Hierarchical Clustering Algorithm, an unsupervised technique that groups data hierarchically. |
| [R43] | EfficientNet-B1 | EfficientNet-B1, a convolutional neural network optimized for image classification with high efficiency | CNN, VGG-16, ResNet-50, MobileNet-V3 | CNN: Convolutional Neural Network, a deep neural network used for automatic feature extraction in images, text, or structured data; VGG-16: Visual Geometry Group 16-layer CNN, a deep convolutional network architecture with 16 layers designed for image classification tasks; ResNet-50: Residual Neural Network 50 layers, a convolutional neural network with residual connections that facilitate the training of deep networks; MobileNet-V3: MobileNet Version 3, a lightweight convolutional network architecture optimized for mobile devices and computer vision tasks with low resource demands. |
| [R62] | NMT | Neural Machine Translation, a neural network-based system for automatic language translation | n/a |  |
| [R23] | GPT-4o | Generative Pre-trained Transformer 4 Omni, an advanced multimodal language model for processing text, images, and audio | GitHub Copilot, GPT-4 Turbo | GitHub Copilot: GitHub Copilot, an OpenAI-assisted artificial intelligence system for autocompletion and code generation in development environments; GPT-4 Turbo: Generative Pre-trained Transformer 4 Turbo, an advanced language model optimized for text generation, programming assistance, and NLP tasks. |
| [R36] | ACO+NSA | Ant Colony Optimization + Negative Selection Algorithm, a combination of ant-based optimization and immune-inspired negative selection algorithm | Random Testing, ACO, NSA | Random Testing: A software testing technique that randomly generates inputs to uncover errors; NSA: Negative Selection Algorithm, a bio-inspired algorithm based on the immune system used to detect anomalies or intrusions. |
| [R05] | SFLA | Shuffled Frog-Leaping Algorithm, a metaheuristic algorithm based on the social behavior of frogs to solve complex problems | GA, PSO, ACO, ABC, SA | GA: Genetic Algorithm, an evolutionary algorithm based on principles of natural selection for solving complex optimization problems; PSO: Particle Swarm Optimization, an optimization algorithm inspired by swarm behavior for finding optimal solutions; ABC: Artificial Bee Colony, an optimization algorithm bioinspired by bee behavior for finding solutions; SA: Simulated Annealing, a probabilistic optimization technique based on the physical annealing process of materials. |
| [R26] | ERINet | Enhanced Residual Inception Network, improved neural architecture for complex pattern recognition | SIFT, SURF, ORB | SIFT: Scale-Invariant Feature Transform, a computer vision algorithm for keypoint detection and description in images; SURF: Speeded-Up Robust Features, a fast and robust algorithm for local feature detection in images; ORB: Oriented FAST and Rotated BRIEF, an efficient method for visual feature detection and image matching. |
| [R63] | ER  -Fuzz (Word2Vec+LSTM) | Error-Revealing Fuzzing with Word2Vec and LSTM, a hybrid approach for generating and analyzing fault-causing inputs | AFL, AFLFast, DT, LSTM | AFL: American Fuzzy Lop, a fuzz testing tool used to discover vulnerabilities by automatically generating malicious input; AFLFast: American Fuzzy Lop Fast, an optimized version of AFL that improves the speed and efficiency of bug detection through fuzzing; DT: Decision Tree, a classifier based on a hierarchical decision structure for classification or regression tasks; LSTM: Long Short-Term Memory, a recurrent neural network designed to learn long-term dependencies in sequences. |
| [R27] | HashC-NC | Hash Coverage - Neuron Coverage, a test coverage approach based on neuron activation in deep networks | *\*(Criterios de evaluación)* NC, 2-way, 3-way, INC, SC, KMNC, HashC-KMNC, TKNC | (Evaluation criteria) NC, 2-way, 3-way, INC, SC, KMNC, HashC-KMNC, TKNC: Set of metrics or techniques for evaluating coverage and diversity in software testing based on neuron activation, combinatorics and structural coverage. |
| [R20] | NSGA-II, MOPSO | NSGA-II: Non-dominated Sorting Genetic Algorithm II, a multi-objective evolutionary algorithm widely used in optimization; MOPSO: Multi-Objective Particle Swarm Optimization, a multi-objective version of particle swarm optimization | Single-objective GA, PSO | Single-objective GA: Single-Objective Genetic Algorithm, a classic genetic algorithm focused on optimizing a single specific objective |
| [R37] | CVDF DYNAMIC (Bi-LSTM+GA) | Cross-Validation Dynamic Feature Selection using Bi-LSTM and Genetic Algorithm for adaptive feature selection | NeuFuzz , VDiscover , AFLFast | NeuFuzz: Neural Fuzzing System, a deep learning-based system for automated test data generation; VDiscover: Vulnerability Discoverer, an automated vulnerability detection tool using dynamic or static analysis; AFLFast: American Fuzzy Lop Fast, a (repeated) optimized system for efficient fuzz testing. |
| [R52] | ARTDL | Adaptive Random Testing Deep Learning, a software testing approach that combines adaptive sampling techniques with deep learning models | RT | RT: Random Testing, a basic strategy for generating random data for software testing |
| [R25] | MTUL (Autoencoder) | Autoencoder-based Multi-Task Unsupervised Learning, used for unsupervised learning and anomaly detection | n/a |  |
| [R61] | RL | Reinforcement Learning, a reward-based machine learning technique for sequential decision-making | GA, ACO, RS | GA: Genetic Algorithm, ACO: Ant Colony Optimization and RS: Random Search, metaheuristics or search strategies combined or applied individually for optimization or classification. |
| [R08] | FrMi | Fractional Minkowski Distance, an improved distance metric for distance-based classifiers | SVM, RF, DT, LR, NB, CNN | Set of traditional classifiers SVM: Support Vector Machine, RF: Random Forest, DT: Decision Tree, LR: Logistic Regression, NB: Naïve Bayes, CNN: Convolutional Neural Network, applied to different prediction or classification tasks. |
| [R31] | MLP | Multilayer Perceptron, a neural network with multiple hidden layers widely used in classification. | Random Strategy, Total Strategy, Additional Strategy | Test case selection or prioritization strategies based on random, exhaustive, or incremental criteria. |
| [R54] | LSTM | Long Short-Term Memory, a recurrent neural network specialized in learning long-term temporal dependencies | n/a |  |
| [R59] | MiTS | Minimal Test Suite, an approach for automatically generating a minimal set of test cases | n/a |  |

**Appendix C.** Variables used in AI studies for ST

| **Subcategory** | **Variable** | **Description** | **Study** |
| --- | --- | --- | --- |
| Source Code Structures | LOC | Total lines of source code | [R11], [R12], [R15], [R22], [R16], [R18], [R28], [R47], [R44], [R51], [R55], [R65], [R07], [R09], [R17], [R46], [R40], [R66], [R34], [R56], [R64], [R42], [R13], [R10], [R19], [R06] |
| Source Code Structures | v(g) | Cyclomatic complexity of the control graph | [R11], [R12], [R15], [R18], [R28], [R29], [R30], [R44], [R51], [R55], [R46], [R40], [R56], [R36], [R05], [R42], [R10], [R06] |
| Source Code Structures | eV(g) | Essential complexity (EVG) | [R11], [R12], [R15], [R18], [R28], [R29], [R44], [R46], [R40], [R56] |
| Source Code Structures | iv(g) | Information Flow Complexity (IVG) | [R11], [R15], [R18], [R28], [R29], [R30], [R44], [R40], [R56] |
| Source Code Structures | npm | Number of public methods | [R01], [R16], [R28], [R65], [R49], [R34] |
| Source Code Structures | NOM | Total number of methods | [R47], [R46], [R06] |
| Source Code Structures | NOPM | Number of public methods | [R47], [R46] |
| Source Code Structures | NOPRM | Number of protected methods | [R47], [R46] |
| Source Code Structures | NOMI | Number of internal or private methods | [R01], [R47], [R46] |
| Source Code Structures | Loc\_com | Lines of code that contain comments | [R01], [R15], [R11], [R28], [R29], [R44], [R50], [R51], [R21], [R46], [R66], [R56] |
| Source Code Structures | Loc\_blank | Blank lines in the source file | [R01], [R11], [R15], [R28], [R29], [R30], [R50], [R51], [R21], [R46], [R34], [R56] |
| Source Code Structures | Loc\_executable | Lines containing executable code | [R01], [R28], [R51], [R07], [R34], [R56] |
| Source Code Structures | LOCphy | Total physical lines of source code | [R29], [R41] |
| Source Code Structures | CountLineCodeDecl | Lines dedicated to declarations | [R01] |
| Source Code Structures | CountLineCode | Total lines of code without comments | [R01], [R28], [R44], [R46], [R49], [R45] |
| Source Code Structures | Locomment | Number of lines containing only comments | [R15], [R22], [R28], [R29], [R44], [R50], [R51], [R09], [R46], [R66], [R34] |
| Source Code Structures | Branchcount | Total number of conditional branches (if, switch, etc.) | [R15], [R30], [R50], [R51], [R07], [R46], [R34], [R56],. , [R19] |
| Source Code Structures | Avg\_CC | Average cyclomatic complexity of the methods | [R28], [R65], [R34] |
| Source Code Structures | max\_cc | Maximum cyclomatic complexity of all methods | [R16], [R28], [R30], [R07], [R34] |
| Source Code Structures | NOA | Total number of attributes in a class | [R47], [R46] |
| Source Code Structures | NOPA | Number of public attributes | [R47], [R46] |
| Source Code Structures | NOPRA | Number of protected attributes | [R47], [R46] |
| Source Code Structures | NOAI | Number of internal/private attributes | [R47], [R46] |
| Source Code Structures | NLoops | Total number of loops (for, while) | [R29] |
| Source Code Structures | NLoopsD | Number of nested loops | [R29] |
| Source Code Structures | max\_cc | Maximum observed cyclomatic complexity between methods | [R50], [R51], [R65], [R17] |
| Source Code Structures | CALL\_PAIRS | Number of pairs of calls between functions | [R51], [R09], [R56] |
| Source Code Structures | CONDITION\_COUNT | Number of boolean conditions (if, while, etc.) | [R51], [R56] |
| Source Code Structures | CYCLOMATI C\_DENSITY (vd(G)) | Cyclomatic complexity density relative to code size | [R51], [R21], [R56] |
| Source Code Structures | DECISION\_count | Number of decision points | [R51], [R56] |
| Source Code Structures | DECISION\_density (dd(G)) | Proportion of decisions to total code | [R51], [R56] |
| Source Code Structures | EDGE\_COUNT | Number of edges in the control flow graph | [R51], [R56] |
| Source Code Structures | ESSENTIAL\_COMPLEXITY (ev(G)) | Unstructured part of the control flow (minimal structuring) | [R51], [R40], [R34], [R56] |
| Source Code Structures | ESSENTIAL\_DENSITY (ed(G)) | Density of the essence complexityl | [R51], [R56] |
| Source Code Structures | PARAMETER\_COUNT | Number of parameters used in functions or methods | [R51], [R21], [R56], [R02] |
| Source Code Structures | MODIFIED\_CONDITION\_COUNT | Counting modified conditions (e.g., if, while) | [R51], [R56] |
| Source Code Structures | MULTIPLE\_CONDITION\_COUNT | Counting compound decisions (e.g., if (a && b)) | [R51], [R56] |
| Source Code Structures | NODE\_COUNT | Total number of nodes in the control graph | [R51], [R56] |
| Source Code Structures | NORMALIZED\_CYLOMATIC\_COMP (Normv(G)) | Cyclomatic complexity divided by lines of code | [R51], [R56] |
| Source Code Structures | NUMBER\_OF\_LINES | Total number of lines in the source file | [R51], [R56] |
| Source Code Structures | PERCENT\_COMMENTS | Percentage of lines that are comments | [R51], [R17], [R21], [R56] |
| Halstead Metrics | n1, n2 / N1, N2 | Number of operators (n1) and unique operands (n2) | [R24], [R50], [R56] |
| Halstead Metrics | V | Program volume | [R11], [R24], [R15], [R29], [R50], [R55], [R46], [R66], [R56] |
| Halstead Metrics | L | Expected program length | [R11], [R24], [R15], [R44], [R51], [R53], [R55], [R46], [R66], [R56] |
| Halstead Metrics | D | Code difficulty | [R11], [R24], [R15], [R29], [R46], [R66], [R56] |
| Halstead Metrics | E | Implementation effort | [R11], [R24], [R15], [R46], [R66], [R56] |
| Halstead Metrics | N | Total length: sum of operators and operands | [R15], [R29], [R50], [R46], [R66], [R53], [R57], [R11], [R12], [R18], [R66], [R34] |
| Halstead Metrics | B | Estimated number of errors | [R15], [R46], [R66], [R56] |
| Halstead Metrics | I | Required intelligence level | [R11], [R15], [R29], [R46], [R56], [R56] |
| Halstead Metrics | T | Estimated time to program the software | [R11], [R15], [R29], [R46], [R56] |
| Halstead Metrics | uniq\_Op | Number of unique operators | [R11], [R12], [R15], [R28], [R29], [R51], [R53], [R57], [R46], [R34], [R19] |
| Halstead Metrics | uniq\_Opnd | Number of unique operators | [R11], [R12], [R15], [R28], [R29], [R51], [R53], [R57], [R46], [R34], [R19] |
| Halstead Metrics | total\_Op | Total operators used | [R11], [R15], [R28], [R29], [R30], [R51], [R53], [R55], [R21], [R46] |
| Halstead Metrics | total opnd | Total operands used | [R15], [R28], [R29], [R51], [R53], [R55], [R46], [R66] |
| Halstead Metrics | hc | Halstead Complexity (may be variant specific) | [R28] |
| Halstead Metrics | hd | Halstead Difficulty | [R28] |
| Halstead Metrics | he | Halstead Effort | [R28], [R30], [R51], [R07], [R34] |
| Halstead Metrics | hee | Halstead Estimated Errors | [R28], [R51], [R53], [R34] |
| Halstead Metrics | hl | Halstead Length | [R28], [R51], [R34] |
| Halstead Metrics | hlen | Estimated Halstead Length | [R28], [R09] |
| Halstead Metrics | hpt | Halstead Programming Time | [R28], [R51] |
| Halstead Metrics | hv | Halstead Volume | [R28], [R51], [R34] |
| Halstead Metrics | Lv | Logical level of program complexity | [R29], [R34] |
| Halstead Metrics | HALSTEAD\_CONTENT | Content calculated according to the Halstead model | [R51], [R21], [R34] |
| Halstead Metrics | HALSTEAD\_DIFFICULTY | Estimated difficulty of understanding the code | [R51], [R34] |
| OO Metrics | amc | Average Method Complexity | [R16], [R28], [R65], [R33], [R38], [R34] |
| OO Metrics | ca | Afferent coupling: number of classes that depend on this | [R16], [R28], [R65], [R49] |
| OO Metrics | cam | Cohesion between class methods | [R16], [R28], [R65], [R17] |
| OO Metrics | cbm | Coupling between class methods | [R16], [R28], [R65], [R49], [R34] |
| OO Metrics | cbo | Coupling Between Object classes | [R16], [R28], [R47], [R57], [R65], [R46], [R49], [R34] |
| OO Metrics | dam | Data Access Metric | [R16], [R28], [R65], [R49], [R34] |
| OO Metrics | dit | Depth of Inheritance Tree | [R16], [R28], [R47], [R65], [R46], [R49], [R34] |
| OO Metrics | ic | Inheritance Coupling | [R16], [R28], [R65], [R49], [R34] |
| OO Metrics | lcom | Lack of Cohesion of Methods | [R16], [R28], [R47], [R65], [R17], [R46], [R49], [R34] |
| OO Metrics | lcom3 | Improved variant of LCOM for detecting cohesion | [R16], [R28], [R65], [R34] |
| OO Metrics | mfa | Measure of Functional Abstraction | [R16], [R28], [R65], [R34] |
| OO Metrics | moa | Measure of Aggregation | [R16], [R28], [R65], [R34] |
| OO Metrics | noc | Number of Children: number of derived classes | [R16], [R28], [R47], [R17], [R46], [R34] |
| OO Metrics | wmc | Weighted Methods per Class | [R16], [R28], [R47], [R57], [R65], [R46], [R34] |
| OO Metrics | FanIn | Number of functions or classes that call a given function | [R47], [R29], [R44], [R46] |
| OO Metrics | FanOut | Number of functions called by a given function | [R47], [R29], [R44], [R46] |
| Software Quality Metrics | rfc | Fan-in OO: Classes that call this class | [R01], [R16], [R28], [R47], [R57], [R46], [R66], [R34] |
| Software Quality Metrics | ce | OO Fan-out: Classes that this class uses | [R01], [R16], [R28], [R65], [R49], [R34] |
| Software Quality Metrics | DESIGN\_COMPLEXITY (iv(G)) | Composite measure of design complexity | [R51], [R09], [R40], [R34], [R56] |
| Software Quality Metrics | DESIGN\_DENSITY (id(G)) | Density of design elements per code unit | [R51], [R56] |
| Software Quality Metrics | GLOBAL\_DATA\_COMPLEXITY (gdv) | Complexity derived from the use of global data | [R51], [R56] |
| Software Quality Metrics | GLOBAL\_DATA\_DENSITY (gd(G)) | Density of access to global data relative to the total | [R51], [R56] |
| Software Quality Metrics | MAINTENANCE\_SEVERITY | Severity in software maintenance | [R51], [R56] |
| Software Quality Metrics | HCM | Composite measure of complexity for maintenance | [R46] |
| Software Quality Metrics | WHCM | Weighted HCM | [R46] |
| Software Quality Metrics | LDHCM | Layered Depth of HCM | [R46] |
| Software Quality Metrics | LGDHCM | Generalized Depth of HCM | [R46] |
| Software Quality Metrics | EDHCM | Extended Depth of HCM | [R46] |
| Change History | NR | Number of revisions | [R46] |
| Change History | NFIX | Number of corrections made | [R46] |
| Change History | NREF | Number of references to previous errors | [R46] |
| Change History | NAUTH | Number of authors who modified the file | [R46] |
| Change History | LOC\_ADDED | Lines of code added in a review | [R46] |
| Change History | maxLOC\_ADDED | Maximum lines added in a single revision | [R46] |
| Change History | avgLOC\_ADDED | Average lines added per review | [R46] |
| Change History | LOC\_REMOVED | Total lines removed | [R46] |
| Change History | max LOC\_REMOVED | Maximum number of lines removed in a revision | [R46] |
| Change History | avg LOC\_REMOVED | Average number of lines removed per review | [R46] |
| Change History | AGE | Age of the file since its creation | [R46] |
| Change History | WAGE | Weighted age by the size of the modifications | [R46] |
| Change History | CVSEntropy | Entropy of repository change history | [R01], [R44] |
| Change History | numberOfNontrivialBugsFoundUntil | Cumulative number of significant bugs found | [R01] |
| Change History | Entropía mejorada | Refined variant of modification entropy | [R22] |
| Change History | fault | Total count of recorded failures | [R16], [R44] |
| Change History | Defects | Total number of defects recorded | [R15], [R46], [R10] |
| Defect History | Bugs | Count of bugs found or related to the file | [R46] |
| Change Metric | codeCHU | Code Change History Unit | [R46] |
| Change Metric | maxCodeCHU | Maximum codeCHU value in a review | [R46] |
| Change Metric | avgCodeCHU | Average codeCHU over time | [R46] |
| Descriptive statistics | mea | Average value (arithmetic mean) | [R22] |
| Descriptive statistics | median | Central value of the data distribution | [R22] |
| Descriptive statistics | SD | Standard deviation: dispersion of the data | [R22] |
| Descriptive statistics | Curtosis | Measure of the concentration of values in the mean | [R22] |
| Descriptive statistics | moments | Statistical moments of a distribution | [R22] |
| Descriptive statistics | skewness | Asymmetry of distribution | [R22] |
| MPI communication | send\_num | Number of MPI submissions (blocking) | [R24] |
| MPI communication | recv\_num | Number of MPI receptions | [R24] |
| MPI communication | Isend\_num | Non-blocking MPI submissions | [R24] |
| MPI communication | Irecv\_num | Non-blocking MPI receptions | [R24] |
| MPI communication | recv\_precedes\_send | Reception occurs before dispatch | [R24] |
| MPI communication | mismatching\_type, size | Incompatible types or sizes in communication | [R24] |
| MPI communication | any\_source, any\_tag | Using wildcards in MPI communication (MPI\_ANY\_SOURCE, etc.) | [R24] |
| MPI communication | recv\_without\_wait | Reception without active waiting (non-blocking) | [R24] |
| MPI communication | send\_without\_wait | Shipping without active waiting | [R24] |
| MPI communication | request\_overwrite | Possible overwriting of MPI requests | [R24] |
| MPI communication | collective\_order\_issue | Order problems in collective operations | [R24] |
| MPI communication | collective\_missing | Lack of required collective calls | [R24] |
| Syntactic Metrics | LCSAt | Total size of the Abstract Syntax Tree (AST) | [R29] |
| Syntactic Metrics | LCSAr | AST depth | [R29] |
| Syntactic Metrics | LCSAu | Number of unique nodes in the AST | [R29] |
| Syntactic Metrics | LCSAm | Average number of nodes per AST branch | [R29] |
| Syntactic Metrics | N\_AST | Total number of nodes in the abstract syntax tree (AST) | [R41] |
| Textual semantics | Line + data/control flow | Logical representation of control/data flow | [R03] |
| Textual semantics | Doc2Vec vector (100 dimensions) | Vectorized textual embedding of source code | [R03] |
| Textual semantics | Token Vector | Tokenized representation of the codeo | [R24], [R63] |
| Textual semantics | Bag of Words | Word frequency-based representation | [R24] |
| Textual semantics | Padded Vector | Normalized vector with padding for neural networks | [R24] |
| Network Metrics | degree\_norm, Katz\_norm | Centrality metrics in dependency graphs | [R03] |
| Network Metrics | closeness\_norm | Normalized closeness metric in dependency graph | [R03] |
| Concurrency Metric | reading\_writing\_same\_buffer | Concurrent access to the same buffer | [R24] |
| Static code metrics | 60 static metrics (calculated with OSA), originally 22 in some datasets. | Source code variables such as lines of code, cyclomatic complexity, and object-oriented metrics, used to predict defects. | [R42], [R06] |
| Execution Dynamics | Relative execution time | Relationship between test duration and total sum | [R04], [R02] |
| Execution Dynamics | Execution history | binary vector with previous results: 0 = failed, 1 = passed | [R04] |
| Execution Dynamics | Last execution | normalized temporal proximity | [R04] |
| Interface Elements | EIem\_Inter (\*) | Extracted interface elements | [R60], [R35], [R39] |
| Programs | Programs (Source code, test case sets, injected fault points, and running scripts.) | Program content | [R64] |
| Graphical models/state diagrams | State Transition Diagrams | OO Systems: Braille translator, microwave, and ATM | [R14] |
| Textual semantics | BoW | Represents the text by word frequency. | [R48] |
| Textual semantics | TF-IDF | Highlights words that are frequent in a text but rare in the corpus. | [R48] |
| Traces and calls | Function names | Names of the functions called in the trace | [R32] |
| Traces and calls | Return values | Return values of functions | [R32] |
| Traces and calls | Arguments | Input arguments used in each call | [R32] |
| Visuals / images | UI\_images | Screenshots (UI) represented by images. | [R43] |
| Traces and calls | class name | Extracted and separated from JUnit classes in Java | [R62] |
| Traces and calls | Method name | Generated from test methods (@Test) | [R62] |
| Traces and calls | Method body | Tokenized source code | [R62] |
| BDD Scenario / Text | BDD Scenario (Given-When text) | CSV generated from user stories | [R23], [R02] |
| GUI Visuals / Interface Processing | GUI images | Visuals (image) + derived structures (masks) | [R26] |
| Textual semantics | If conditions + tokens | Conditional fragments and tokenized structures for error handling classification. | [R63] |
| Embedded representation | Word2Vec embedding | Vector representation of source code for input to the classifier. | [R63] |
| Supervised labeling | Error-handling tag | Binary variable to train the classifier (error handling/normal) | [R63] |
| Embedded representation | Neural activations | Internal outputs of neurons in different layers of the model under test inputs | [R27] |
| Embedded representation | Active combinations | Sets of neurons activated simultaneously during execution | [R27] |
| Embedded representation | Hash combinations | Hash representation of active joins to speed up coverage evaluation (HashC-NC) | [R27 |
| GUI interaction | Events (interaction sequences) | Clicks, keys pressed, sequence of actions | [R20] |
| Test set | Test Paths | Sets of events executed by a test case | [R20] |
| Textual semantics | Input sequence | Character sequence (fuzz inputs) processed by Bi-LSTM | [R37] |
| Fuzzing | Unique paths executed | Measure of structural effectiveness of the coverage test | [R37] |
| Fuzzing (search-based) | Entry Fitness | Probabilistic evaluation of the input value within GA | [R37] |
| Visuals / images | Activations of conv3\_2 and conv4\_2 layers | Vector representations of images extracted from VGGNet layers to measure diversity in fuzzing. | [R52] |
| Latent representations (autoencoding) | Autoencoder outputs, mutated inputs, latent distances | Mutated autoencoder representations evaluated for their effect on clustering. | [R25] |
| Integration Structure / OO Dependencies | Dependencies between classes, number of stubs generated, graph size | Relationships between classes and number of stubs needed to execute the proposed integration order. | [R61] |
| Mutant execution metrics | Number of test cases that kill the mutant, killability severity, mutated code, operator class | Statistical and structural attributes of mutants used as features to classify their ability to reveal real faults. | [R08] |
| Multisource (history + code) | 04 features (52 code metrics, 8 clone metrics, 42 coding rule violations, 2 Git metrics) | Source code attributes and change history used to estimate fault proneness using MLP. | [R31] |
| Time sequence (interaction) | Sequence of player states (actions, objects, score, time, events) | Temporal game interaction variables used as input to an LSTM network to generate test events and evaluate gameplay. | [R54] |
| Structural combinatorics | Array size, levels per factor, ttt coverage, mixed cardinalities | Combinatorial design parameters (values per factor and interaction strength) used to construct optimal test arrays via tabu search. | [R59] |

(\*)EIem\_Inter: Text Box, E-mail, Text Box, Password, Button Back, Button, Login, Link Forgot Password,Button Create an account, Text Box First Name, Text Box, Last Name, Text Box Password, Text Box Date of Birth, Text Box, e-mail address, Text Box Country, Button Sign Up, Button Back, Item Product, Dropdown colour, Dropdown size, Dropdown, quantity, Button Buy Now.

**Appendix D.** Metrics used in AI studies for ST

| **Discipline** | **Description** | **Metrics / Formula** | **Study** |
| --- | --- | --- | --- |
| Classic performance | Proportion of correct predictions out of the total number of cases evaluated. |  | [R22], [R24], [R11], [R15], [R44], [R51], [R53], [R55], [R57], [R07], [R09], [R17], [R21], [R38], [R40], [R49], [R34], [R43], [R63], [R37], [R08], [R42], [R02], [R10], [R19], [R06] |
| Classic performance | Measures the proportion of true positives among all positive predictions made. |  | [R22], [R24], [R11], [R15], [R16], [R42], [R28], [R29], [R55], [R57], [R65], [R07], [R09], [R21], [R49], [R66], [R60], [R32], [R63], [R08], [R02], [R13], [R10], [R19], [R06] |
| Classic performance | Evaluates the model's ability to correctly identify all positive cases. |  | [R22], [R24], [R11], [R15], [R42], [R18], [R29], [R50], [R55], [R57], [R65], [R07], [R09], [R21], [R40], [R49], [R66], [R60], [R32], [R63], [R08], [R02], [R10], [R19], [R06] |
| Classic performance | Harmonious balance between precision and recall, useful in scenarios with unbalanced classes. |  | [R22], [R11], [R15], [R16], [R42], [R28], [R47], [R29], [R41], [R44], [R51], [R53], [R55], [R65], [R07], [R40], [R49], [R66], [R60], [R63], [R08], [R02], [R10], [R19], [R06] |
| Advanced Classification | Evaluates the quality of predictions considering true and false positives and negatives. |  | [R03], [R22], [R28], [R51], [R53], [R65], [R33], [R66] |
| Advanced Classification | Summarizes the model's ability to discriminate between positive and negative classes at different thresholds |  | [R01], [R03], [R16], [R42], [R18], [R28], [R29], [R30], [R41], [R44], [R51], [R55], [R57], [R65], [R07], [R38], [R40], [R48], [R08], [R19], [R06] |
| Advanced Classification | Averages sensitivity and specificity, useful when classes are unbalanced. |  | [R03] |
| Advanced Classification | Geometric between sensitivity and specificity, measures the balance in binary classification. |  | [R03], [R16], [R18], [R55], [R65], [R33], [R46] |
| Alarms and Risk | Measures the proportion of true negatives detected among all true negative cases. |  | [R22], [R15], [R55], [R57], [R09], [R21], [R40] |

| Discipline | Description | Metrics / Formula | Study |
| --- | --- | --- | --- |
| Alarms and Risk | Proportion of true negatives among all negative predictions. |  | [R22], [R09], [R21] |
| Alarms and Risk | Proportion of false positives among all positive predictions. |  | [R22] |
| Alarms and Risk | Proportion of undetected positives among all true positives. |  | [R22], [R12], [R57], [R09], [R21], [R33] |
| Alarms and Risk | Proportion of negatives incorrectly classified as positives. |  | [R22], [R12], [R18], [R50], [R57], [R65], [R09], [R21], [R33]. [R18], [R37] |
| Alarms and Risk | Ability of the model to correctly identify real positives. |  | [R37] |
| Alarms and Risk | Ability of the model to correctly identify real negatives. |  | [R32] |
| Alarms and Risk | Proportion of true positives correctly identified by the model. |  | [R18] |
| Alarms and Risk | Proportion of negatives incorrectly identified as positives. |  | [R18] |
| Software Testing-Specific Metrics | Measures the effort required (in percentage of LOC or files) to reach 20% recall. |  | [R03] |
| Software Testing-Specific Metrics | Percentage of defects found within the 20% most suspicious lines of code. |  | [R03] |
| Software Testing-Specific Metrics | Number of false positives before finding the first true positive. | IFA = Number of non-defective instances before the first defect found | [R03], [R06] |
| Software Testing-Specific Metrics | Accuracy among the k elements best ranked by the model. |  | [R03] |
| Software Testing-Specific Metrics | Effort metric that combines precision and recall with weighting of the inspected code. |  | [R44] |
| Software Testing-Specific Metrics | It is used to compare how effectively a model detects faults early relative to a baseline model. |  | [R04] |
| Software Testing-Specific Metrics | Expected number of test cases generated until the first failure is detected. |  | [R52] |
| Software Testing-Specific Metrics | Number of rows needed to cover all combinations t |  | [R59] |
| Software Testing-Specific Metrics | Time required by MiTS to build the array |  | [R59] |
| Software Testing-Specific Metrics | Improvement compared to the best previously known values |  | [R59] |
| Cost/Error and Probabilistic Metrics | Measures the mean square error between predicted probabilities and actual outcomes (lower is better). |  | [R16] |
| Cost/Error and Probabilistic Metrics | Distance of the model to an ideal classifier with 100% TPR and 0% FPR. |  | [R16] |
| Cost/Error and Probabilistic Metrics | Root mean square error between predicted and actual values; useful for regression models. |  | [R53] |
| Cost/Error and Probabilistic Metrics | Expected time it takes for the model to detect a positive instance (defect) correctly. |  | [R53] |
| Cost/Error and Probabilistic Metrics | Ratio between the actual effort needed to achieve a certain recall and the optimal possible effort. |  | [R57] |
| Cost/Error and Probabilistic Metrics | Proportion of incorrectly classified instances relative to the total. |  | [R09], [R21], [R56] |
| Coverage, Execution, GUI, and Deep Learning | Evaluates the speed of test point coverage. The closer to 1, the better. |  | [R64] |
| Coverage, Execution, GUI, and Deep Learning | Evaluate the total runtime until full coverage is achieved. The lower the better. |  | [R64] |
| Coverage, Execution, GUI, and Deep Learning | Evaluates the similarity between a generated text (e.g., test case) and a reference text, using n-gram matches and brevity penalties. |  | [R35], [R39], [R62] |
| Coverage, Execution, GUI, and Deep Learning | Measures the average accuracy of the model in object detection at different matching thresholds (IoU). |  | [R39] |
| Coverage, Execution, GUI, and Deep Learning | Measures the total time it takes for an algorithm to generate all test paths. |  | [R14], [R20], [R25], [R27], [R37], [R61] |
| Coverage, Execution, GUI, and Deep Learning | Indicates the proportion of repeated or unnecessary test paths generated by the algorithm. |  | [R14] |
| Coverage, Execution, GUI, and Deep Learning | Fraction of generated step methods that have implementation |  | [R23] |
| Coverage, Execution, GUI, and Deep Learning | Fraction of generated step methods without implementation |  | [R23] |
| Coverage, Execution, GUI, and Deep Learning | Fraction of generated POM methods with functional implementation |  | [R23] |
| Coverage, Execution, GUI, and Deep Learning | Average number of paths covered by the algorithm |  | [R36], [R05] |
| Coverage, Execution, GUI, and Deep Learning | Average number of generations needed to cover all paths |  | [R36], [R05] |
| Coverage, Execution, GUI, and Deep Learning | Percentage of executions that cover all paths |  | [R36], [R05] |
| Coverage, Execution, GUI, and Deep Learning | Average execution time of the algorithm |  | [R36], [R05] |
| Coverage, Execution, GUI, and Deep Learning | It is equivalent to an accuracy metric, applied to a visual matching task. |  | [R26] |
| Coverage, Execution, GUI, and Deep Learning | Measures how many unique neural combinations have been covered |  | [R27] |
| Coverage, Execution, GUI, and Deep Learning | Measures whether a neuron was activated at least once |  | [R27] |
| Coverage, Execution, GUI, and Deep Learning | Coverage of combinations of 2 neurons activated together |  | [R27] |
| Coverage, Execution, GUI, and Deep Learning | Coverage of combinations of 3 neurons activated together |  | [R27] |
| Coverage, Execution, GUI, and Deep Learning | Percentage of test paths covered by the generated test cases |  | [R20] |
| Coverage, Execution, GUI, and Deep Learning | % of unique events covered (equivalent to coverage by GUI widgets) |  | [R20] |
| Coverage, Execution, GUI, and Deep Learning | Percentage of code executed during testing. |  | [R37] |
| Coverage, Execution, GUI, and Deep Learning | Weighted measure of coverage diversity among generated cases. |  | [R37] |
| Coverage, Execution, GUI, and Deep Learning | Proportion of mutants detected per change in system output |  | [R25] |
| Coverage, Execution, GUI, and Deep Learning | Euclidean distance in latent space between original and mutated input |  | [R25] |
| Coverage, Execution, GUI, and Deep Learning | Total number of stubs needed for each order |  | [R61] |
| Coverage, Execution, GUI, and Deep Learning | Reduction in the number of stubs compared to baseline |  | [R61] |
| Coverage, Execution, GUI, and Deep Learning | Evaluate the effectiveness of test case prioritization |  | [R31] |
| Coverage, Execution, GUI, and Deep Learning | Percentage of LSTM predictions that match expected gameplay |  | [R54] |
| Coverage, Execution, GUI, and Deep Learning | Measure of balance between the actions and responses of the game |  | [R54] |