Digital Assignment 1

A Case Study on Tools and Datasets for EDA

Sparsh Karna 23BDS1172
Lavanaya Malhotra 23BDS1169

Course Name: Data Mining
Course Code: BCSE208L

Contents

1 Understanding Statistical Data Analytical Tools (5 marks)

1.1 RStudio

RStudio is an integrated development environment (IDE) for the R programming language. It provides a console, syntax-highlighting editor, and tools for plotting, package management, and workspace management. R is open-source, script-driven, and supports a vast ecosystem of packages (CRAN hosts over 20,000). For data mining, key packages include `caret` (unified modelling), `randomForest`, `rpart`, `ggplot2` (visualization), and `dplyr` (data wrangling). Its strengths lie in reproducibility (scripted pipelines), statistical rigour, and extensibility. However, it has a steeper learning curve for users unfamiliar with programming.

1.2 JMP

JMP (pronounced "jump") is a proprietary statistical software developed by SAS Institute. It emphasises interactive, GUI-driven exploration with dynamic linking between data tables and visualizations. JMP supports a wide range of analyses—from basic EDA (distribution, correlation) to advanced modelling (logistic regression, neural networks, decision trees). Its drag-and-drop interface makes it accessible to non-programmers, and its interactive plots allow rapid hypothesis exploration. Limitations include high licensing cost, less flexibility for custom algorithms, and limited scripting compared to R.

1.3 Weka

Weka (Waikato Environment for Knowledge Analysis) is a free, open-source machine-learning toolkit written in Java. It provides a GUI (Explorer, Experimenter, KnowledgeFlow) and a command-line interface. Weka includes implementations of numerous classifiers (J48, Random Forest, Logistic Regression, SVM, Naïve Bayes, etc.), clustering algorithms, and feature-selection methods. It natively reads ARFF and CSV files and offers built-in cross-validation. Weka is widely used in academic settings for its ease of use and breadth of algorithms but is less suited for large-scale production deployment and has limited visualization capabilities compared to R or JMP.

2 Exploration of IEEE Data Portal & Dataset Selection (5 marks)

2.1 Dataset Categories Explored

Web-based data
: Data collected from web interactions such as clickstreams, server logs, and social media feeds. Typically high-volume, semi-structured, and suitable for text mining and user-behaviour analysis.

Statistical data
: Structured numerical/categorical datasets amenable to classical statistical analysis—hypothesis testing, regression, ANOVA. Often tabular with well-defined attributes.

Spatial data
Data with geographic or positional attributes (GPS coordinates, geofences). Requires specialised tools for spatial indexing, map visualization, and geospatial modelling.

Temporal data
Time-series or time-stamped data capturing evolution over time. Requires handling of trends, seasonality, and autocorrelation.

2.2 Selected Dataset

Source: IEEE DataPort — "IoT Nodes Block Storage Acceptance Dataset"
DOI: https://dx.doi.org/10.21227/eyc1-a041

The selected dataset is a **statistical dataset** comprising 30,000 records and 11 attributes. It is synthetically generated for binary classification of IoT edge nodes to determine whether each node can accept an additional blockchain block for storage. The scenario addresses edge blockchain applications where an edge miner, having exhausted its internal storage, may delegate block storage to nearby IoT nodes, provided those nodes' configurations support it without disrupting normal operation.

2.2.1 Attributes

Table 1: Dataset Attributes and Descriptions

Attribute	Description
Block Size (KB)	Size of the block to be stored (10–1024 KB)
Device ID	Unique identifier for each IoT node (30,000 unique devices)
Device Type	Application type of the device (30 categories, e.g. Smart Light Bulb, Smart Thermostat)
Microcontroller	Microcontroller architecture used (11 types after normalization, e.g. ESP32, STM32F4)
Critical Score	Criticality of the application (integer, 1–5)
Total Storage (KB)	Total EEPROM/secondary storage capacity (128–1,048,576 KB)
Current Available Storage (KB)	Free space at time t (0–1,048,576 KB)
Blockchain Load (KB)	Current blockchain data on the device at time t (0–104,305 KB)
Stability Score	Uptime/reliability score at time t (integer, 1–5)
Projected Growth (KB/day)	Daily on-device storage growth rate
Output (0/1)	Binary target: 1 = node suitable for block storage, 0 = not suitable

2.2.2 Potential Use

This dataset enables research into intelligent storage delegation in edge blockchain systems. Accurate classification helps edge miners quickly identify suitable IoT nodes for block storage, thereby optimizing blockchain performance, reducing latency, and avoiding disruption of critical IoT applications.

3 Comparative Analysis: R vs. Weka vs. JMP (5 marks)

3.1 Strengths and Limitations

Table 2: Comparative Analysis of R, Weka, and JMP

Criterion	R (RStudio)	Weka	JMP
Cost	Free, open-source	Free, open-source	Proprietary (paid license)
Usability	Script-based; steep learning curve	GUI-driven; moderate learning curve	GUI-driven; easy for beginners
Visualization	Excellent (ggplot2, corrplot, GGally); highly customizable	Basic; limited plot types	Excellent; interactive, dynamically linked
Modelling	Vast ecosystem (caret, randomForest, glmnet, etc.)	Broad built-in classifiers (J48, RF, SVM, etc.)	Good (logistic, decision tree, neural net)
Reproducibility	Excellent (scripted pipeline, version control)	Moderate (GUI-based, exportable configs)	Low (GUI-driven, limited scripting)
Scalability	Handles large datasets with optimized packages	Slower on very large datasets (Java memory)	Handles moderate datasets well
Extensibility	20,000+ CRAN packages	Plugin architecture; less flexible	JSL scripting; limited

3.2 Differences in Usability, Visualization, and Modelling

Usability: JMP and Weka offer point-and-click interfaces suitable for rapid prototyping and users without programming background. R requires coding but provides full control over the analytical pipeline, enabling complex data transformations and custom modelling workflows.

Visualization: R's `ggplot2` ecosystem produced the richest visualizations in our analysis—histograms, boxplots stratified by target, correlation heatmaps, pairwise scatter matrices, ROC curves, and variable importance plots. JMP's interactive plots (distribution overlays, correlation matrices) were visually appealing and allowed instant exploration. Weka's visualization was limited to basic margin curves and class-conditional plots.

Modelling: All three tools successfully trained classifiers on the IoT dataset. R (via `caret`) allowed fine-grained control over cross-validation, hyperparameter tuning, and model comparison. Weka provided one-click access to J48, Random Forest, and Logistic Regression with 10-fold CV built in. JMP's Nominal Logistic regression was straightforward but offered fewer classifier options out of the box.

3.3 Influence of Dataset Type on Tool Choice

Our dataset is a **statistical** dataset—structured, tabular, with numerical and categorical features and a binary target. For such data:

- **R** is ideal: it handles the full pipeline (cleaning, EDA, modelling, evaluation) in a reproducible script and supports the widest range of classifiers.
- **Weka** is well-suited for quick classifier comparison via its GUI and built-in cross-validation.
- **JMP** is effective for interactive EDA but less flexible for advanced modelling.
- For *spatial* data, R (with `sf`, `leaflet`) or specialised GIS tools would be preferred. For *temporal* data, R (with `forecast`, `tseries`) excels. For *web-based* data, R or Python are preferred for text parsing and NLP.

3.4 Tool Selected for the Case Study

R (RStudio) was chosen as the primary tool because:

1. It supports the entire pipeline (data cleaning, EDA, modelling, evaluation) in a single reproducible script.
2. `caret` provides a unified interface for Logistic Regression, Decision Tree, and Random Forest with consistent cross-validation and metric computation.
3. Visualization via `ggplot2` produces publication-quality figures.
4. Open-source and free.

Results from JMP and Weka are included for comparative validation.

4 Mini Case Study (15 marks)

4.1 Problem Statement

Can we accurately predict whether an IoT edge node is suitable for storing an additional blockchain block, based on the node's hardware specifications, current resource utilization, and application characteristics?

This is a binary classification problem where the target variable `Output (0/1)` indicates storage acceptance (1) or rejection (0). Accurate prediction enables intelligent block delegation in edge blockchain systems, minimizing the risk of overloading critical or resource-constrained devices.

4.2 Related Research

The problem of deciding whether an IoT edge node can safely store an additional blockchain block sits at the intersection of blockchain-enabled edge computing, IoT resource management, and intelligent node selection. Below is a review of recent research organized by methodological approach.

4.2.1 Reinforcement-Learning–Based Resource Allocation and Offloading

He et al. [1] propose a blockchain-based edge computing architecture in which smart contracts use the Asynchronous Advantage Actor–Critic (A3C) deep reinforcement learning algorithm to allocate edge computing resources to IoT devices. The system observes

network states such as task arrival rates and available resources at edge nodes, and the RL agent learns policies that maximize long-term utility (e.g., low delay and energy consumption) while ensuring security and trust via blockchain. Experiments on simulated IoT scenarios show that their RL-based allocation outperforms heuristic baselines in terms of latency and resource utilization. Conceptually, this work is close to our dataset: it also decides which edge nodes should take on extra workload (computation tasks instead of storage blocks), but uses RL and smart contracts rather than a supervised classifier.

Xu et al. [2] propose BCO, a blockchain-based computation offloading method for edge computing in 5G networks. They design a blockchain-backed EC framework and formulate multi-objective offloading decisions (balancing delay, data loss, and edge node load), solved using NSGA-III (a multi-objective evolutionary algorithm) followed by multi-criteria decision making (SAW + MCDM) to choose the best strategy. While their focus is on offloading from user equipment to edge nodes, the underlying problem is again selecting which nodes can accept extra tasks given their limited resources—analogous to predicting `storage_acceptance` in our dataset.

Heidari and Jamali [3] propose a deep Q-learning approach for offline/online computation offloading in blockchain-enabled green IoT–edge scenarios. They model the offloading process as a Markov Decision Process (MDP), where the RL agent chooses offloading destinations based on state variables such as block size, available processing capability, and network conditions. Simulation results indicate significant improvements in cost, computational overhead, energy use, failure rate, and latency compared to several benchmark strategies. This reinforces the idea that intelligent node/task assignment in blockchain–IoT systems is a key research challenge, and that block size and node resources are central features—exactly those present in our dataset (`block_size`, `total_storage`, `current_available_storage`).

4.2.2 Fuzzy and Multi-Criteria Decision-Making Approaches

Gardas et al. [4] propose a fuzzy-based method for object selection in blockchain-enabled edge–IoT platforms using a hybrid multi-criteria decision-making (MCDM) model. They address the need to choose appropriate IoT nodes to ensure sufficient network lifetime, reliable data, and efficient resource allocation. Their method combines fuzzy logic with the TOPSIS MCDM technique: input criteria (resource availability, trust, and performance metrics) are fuzzified, aggregated, and ranked to select the most suitable IoT object. Experiments on synthetic scenarios show that the fuzzy-MCDM approach improves selection quality compared to simpler methods. This is conceptually very close to our problem: our dataset also encodes multiple criteria (storage capacity, criticality, stability, projected growth), but instead of fuzzy TOPSIS, we apply standard classification algorithms in R, Weka, and JMP.

Imamguluyev et al. [5] focus specifically on node selection in blockchain-enabled edge IoT using a fuzzy logic approach. In their I-SMAC conference paper, they design a fuzzy-rule-based system for selecting nodes that will participate in blockchain operations at the edge, considering parameters such as node performance, resource availability, and network constraints. Their experiments demonstrate improved efficiency and reliability in node selection compared to baseline strategies. This reinforces the relevance of our dataset's features such as `stability_score`, `current_available_storage`, and `projected_growth`: these are exactly the types of inputs that fuzzy systems and MCDM

schemes use to calculate node suitability.

Cuka et al. [6] study IoT node selection in opportunistic networks using fuzzy-based simulation systems and a physical testbed. Although their context is not blockchain, they address a similar decision problem: selecting which IoT nodes should carry tasks in intermittently connected networks. Their fuzzy systems incorporate inputs such as node distance to task, remaining energy, buffer occupancy, and inter-contact time. This underscores the general pattern that node-selection problems in IoT often rely on a small set of interpretable node attributes, similar to those present in our dataset.

4.2.3 Integration of Blockchain and Edge Computing for IoT Data and Storage

Haque et al. [7] propose a scalable blockchain-based framework for efficient IoT data management using lightweight consensus. They observe that traditional blockchain protocols struggle with scalability when faced with huge data volumes and numbers of devices in IoT networks. Their framework adopts a lightweight consensus mechanism and a hierarchical architecture to improve throughput and reduce latency and energy consumption. While they focus more on consensus and throughput than on individual node admission decisions, their work motivates why edge nodes cannot all act as full blockchain storage nodes—selection is needed, which our dataset explicitly tackles via the `storage_acceptance` label.

Zubaydi et al. [8] review blockchain-based approaches for securing IoT systems, covering data integrity, authentication, and access control. They highlight the tension between resource-constrained IoT devices (limited memory, processing, bandwidth) and the heavy resource requirements of many blockchain protocols, arguing that careful architectural design and node role assignment are necessary. This supports the rationale for our dataset's attributes: total and available storage, growth rate, and criticality are exactly the types of constraints that security-oriented IoT–blockchain deployments must consider.

Liu et al. [9] survey the integration of blockchain and edge computing in IoT (IBEC), analyzing architectures that combine blockchain with edge computing to enhance data security and resource utilization. The survey outlines a generic IBEC architecture and reviews optimization strategies for resource management, performance, and security. It emphasizes challenges such as scalability, latency, and heterogeneous resource constraints of edge nodes. This positions our problem within a broader research landscape: the binary classification of "can this node store an extra block?" can be seen as a micro-decision embedded within larger IBEC resource management frameworks.

4.2.4 Synthesis and Relation to Our Dataset

Across these works, several common themes emerge:

- **Node selection / resource allocation as a central challenge:** Whether through deep RL, fuzzy MCDM, or evolutionary optimization, the literature consistently frames "which node should handle this extra workload?" as a key problem. Our dataset operationalizes exactly this question for storage: given attributes of an edge IoT node and the incoming block size, decide whether the node can safely store the block.

- **Multi-criteria nature of the decision:** Prior work typically considers multiple criteria simultaneously—energy, remaining buffer/storage, latency, trust/reputation, and application-specific importance. Our dataset encodes analogous dimensions: `critical_score`, `total_storage`, `current_available_storage`, `blockchain_load`, `stability_score`, and `projected_growth`. This alignment suggests that our feature set is well grounded in the factors the research community considers important.
- **Handling of uncertainty and dynamics:** Fuzzy logic and MCDM are used when system states are uncertain, while RL methods explicitly model dynamic environments. Our dataset abstracts away temporal complexity by capturing snapshots of node state at time t and uses a static binary label, making it well suited for supervised learning experiments while remaining conceptually consistent with dynamic selection frameworks.
- **Lack of openly documented node-selection datasets:** Most of the above works rely on custom simulation environments or proprietary scenarios and rarely publish reusable datasets. The IEEE DataPort dataset we use explicitly publishes a synthetic but richly annotated binary-classification dataset for IoT node storage acceptance, filling a practical gap and enabling benchmarking of classical ML models against heuristic, fuzzy, and RL-based methods.
- **Focus of prior work vs. our case study:** Existing studies often combine architecture design, security mechanisms, and intelligent decision algorithms (RL, fuzzy logic, evolutionary optimization). Our case study is narrower: it focuses on applying standard statistical/ML tools to a well-defined binary classification dataset and comparing software environments (R, Weka, JMP). The novelty in our context is mapping a realistic blockchain–edge–IoT decision problem into a clean supervised-learning task and empirically analyzing which modelling approaches and tools are most effective.

References

[1] Y. He et al., "Blockchain-based Edge Computing Resource Allocation in IoT: A Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, 2020. DOI: 10.1109/JIOT.2020.3035437.

[2] X. Xu et al., "Blockchain-based computation offloading for edge computing in 5G networks," *Software: Practice and Experience*, vol. 50, no. 9, pp. 1631–1648. DOI: 10.1002/spe.2749.

[3] A. Heidari and M. A. Jamali, "Deep Q-Learning Technique for Offloading Offline/Online Computation in Blockchain-Enabled Green IoT-Edge Scenarios," *Semantic Scholar*, 2021.

[4] B. Gardas et al., "A Fuzzy-Based Method for Objects Selection in Blockchain-Enabled Edge-IoT Platforms Using a Hybrid Multi-Criteria Decision-Making Model," *DOAJ*, 2022.

[5] R. Imamguluyev et al., "Node Selection in Blockchain-Enabled Edge IoT Using Fuzzy Logic," *Proc. I-SMAC*, IEEE, 2023. DOI: 10.1109/I-SMAC58438.2023.10290320.

[6] M. Cuka et al., "IoT node selection in opportunistic networks: Implementation of fuzzy-based simulation systems and a testbed," *Internet of Things*, vol. 8, 100105, 2019. DOI: 10.1016/j.iot.2019.100105.

[7] R. Haque et al., "A scalable blockchain-based framework for efficient IoT data management using lightweight consensus," *Scientific Reports*, 2024. PMC: 10991409.

[8] H. Zubaydi et al., "Blockchain-based approaches for securing IoT systems: A survey," *Sensors*, 2023. PMC: 9867322.

[9] Y. Liu et al., "Integration of Blockchain and Edge Computing in Internet of Things: A Survey," *Future Generation Computer Systems*, 2022. DOI: 10.1016/j.future.2022.10.029.

4.3 Methodology

4.3.1 Dataset & Preprocessing

- **Dataset:** 30,000 records, 11 attributes (after dropping Device ID as a non-predictive identifier).
- **Missing values:** None (verified across all columns—see Table 3).
- **Categorical encoding:** Device Type (30 categories) and Microcontroller (16 raw categories, normalized to 11 after merging duplicates such as "Nordic nRF52832" → "nRF52832" and "ESP32 CAM" → "ESP32") were one-hot encoded by R's `caret` framework.
- **Target variable:** Converted to factor with levels `no` (0) and `yes` (1).
- **Train/Test split:** 80/20 stratified split (24,000 train / 6,000 test).
- **Cross-validation:** 10-fold CV on the training set, optimizing AUC (ROC).

Table 3: Missing Value Summary (all zero)

Variable	Missing Count
block_size_kb	0
device_type	0
microcontroller	0
critical_score	0
total_storage_kb	0
current_available_storage_kb	0
blockchain_load_kb	0
stability_score	0
projected_growth_kb_day	0
output_0_1	0

4.3.2 Exploratory Data Analysis (EDA)

Descriptive Statistics. Table 4 presents the five-number summary of numerical features.

Table 4: Descriptive Statistics of Numerical Features

Feature	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Block Size (KB)	10.0	148.2	303.5	371.6	553.5	1024.0
Critical Score	1	2	3	2.93	4	5
Total Storage (KB)	128	512	1024	23128	4096	1048576
Available Storage (KB)	0.0	190.3	467.1	9824.8	2025.7	995810.0
Blockchain Load (KB)	0.0	95.0	215.0	1052.1	426.1	104305.7
Stability Score	1	2	3	3.08	4	5
Projected Growth (KB/day)	3.6	43.7	80.6	1531.0	197.1	200000.0

Key observations: Total Storage, Available Storage, Blockchain Load, and Projected Growth are heavily right-skewed (mean $\gg$ median), indicating a few high-capacity devices dominating the upper range.

Correlation Analysis. The correlation matrix (from both R and JMP) reveals:

- **Total Storage** $\leftrightarrow$ **Available Storage**: $r = 0.895$ (strong positive)—devices with higher total capacity tend to have more free space.
- **Total Storage** $\leftrightarrow$ **Projected Growth**: $r = 0.750$ (strong positive)—larger devices tend to have higher application growth rates.
- **Blockchain Load** $\leftrightarrow$ **Projected Growth**: $r = 0.551$ (moderate positive).
- **Critical Score** $\leftrightarrow$ **Stability Score**: $r = -0.500$ (moderate negative)—higher-criticality devices tend to have lower stability scores, possibly reflecting more demanding operating conditions.
- **Block Size** shows weak correlations with all other features ($|r| < 0.27$).

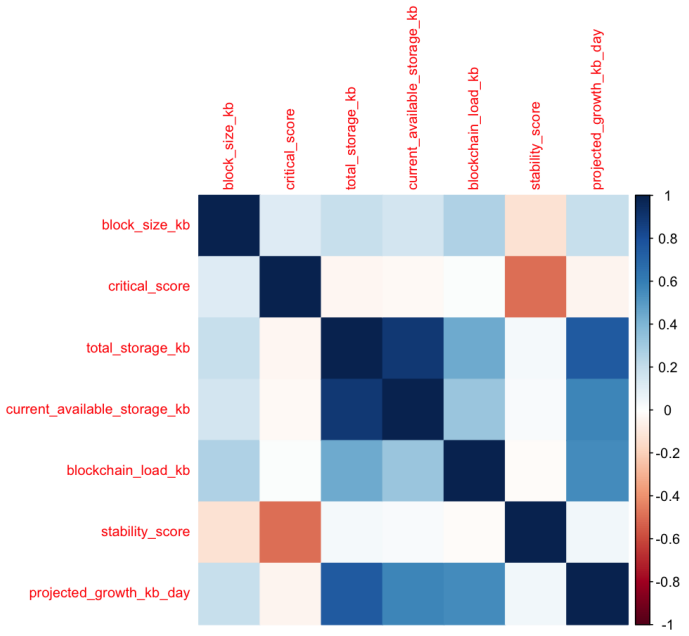


Figure 1: Correlation Heatmap (R)

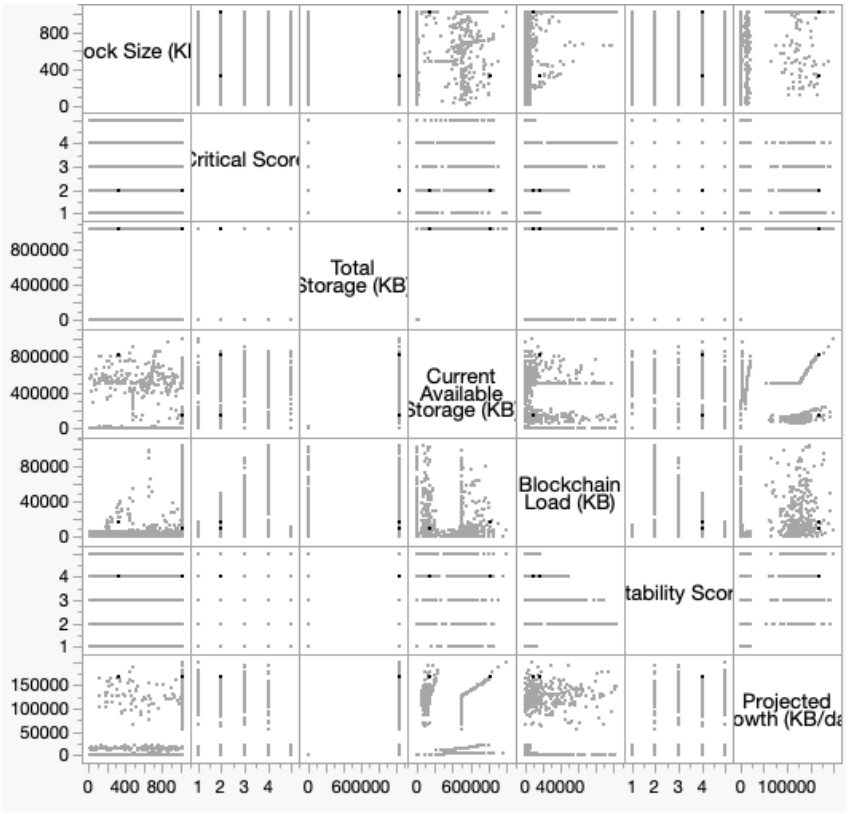
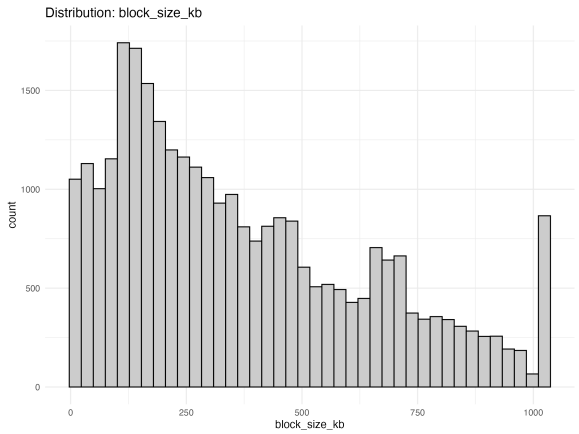
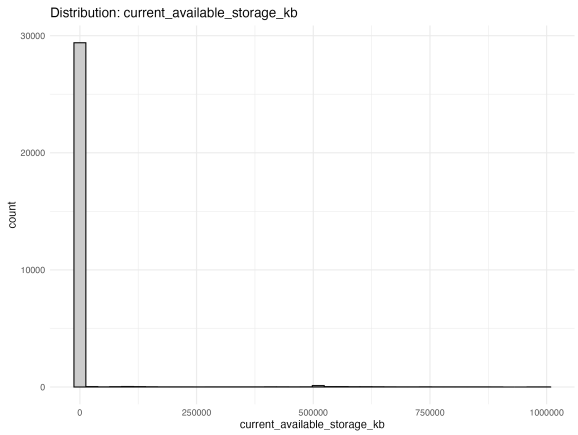


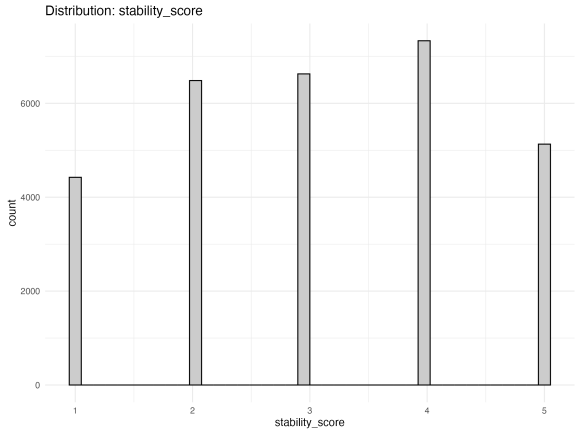
Figure 2: Correlation Matrix (JMP)



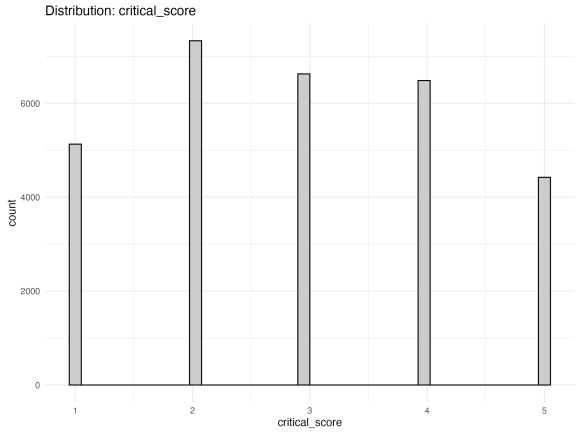
(a) Block Size Distribution



(b) Available Storage Distribution



(c) Stability Score Distribution



(d) Critical Score Distribution

Figure 3: Feature Distributions (R)

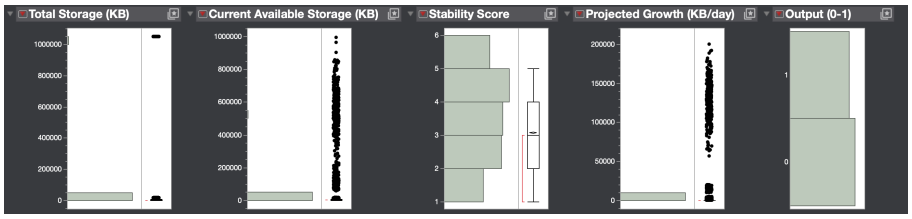
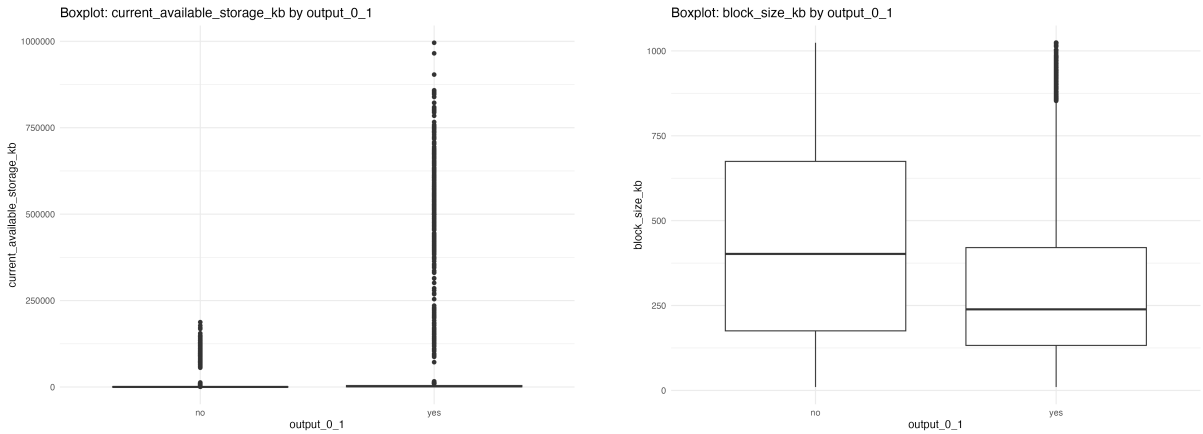


Figure 4: Feature Distributions (JMP)

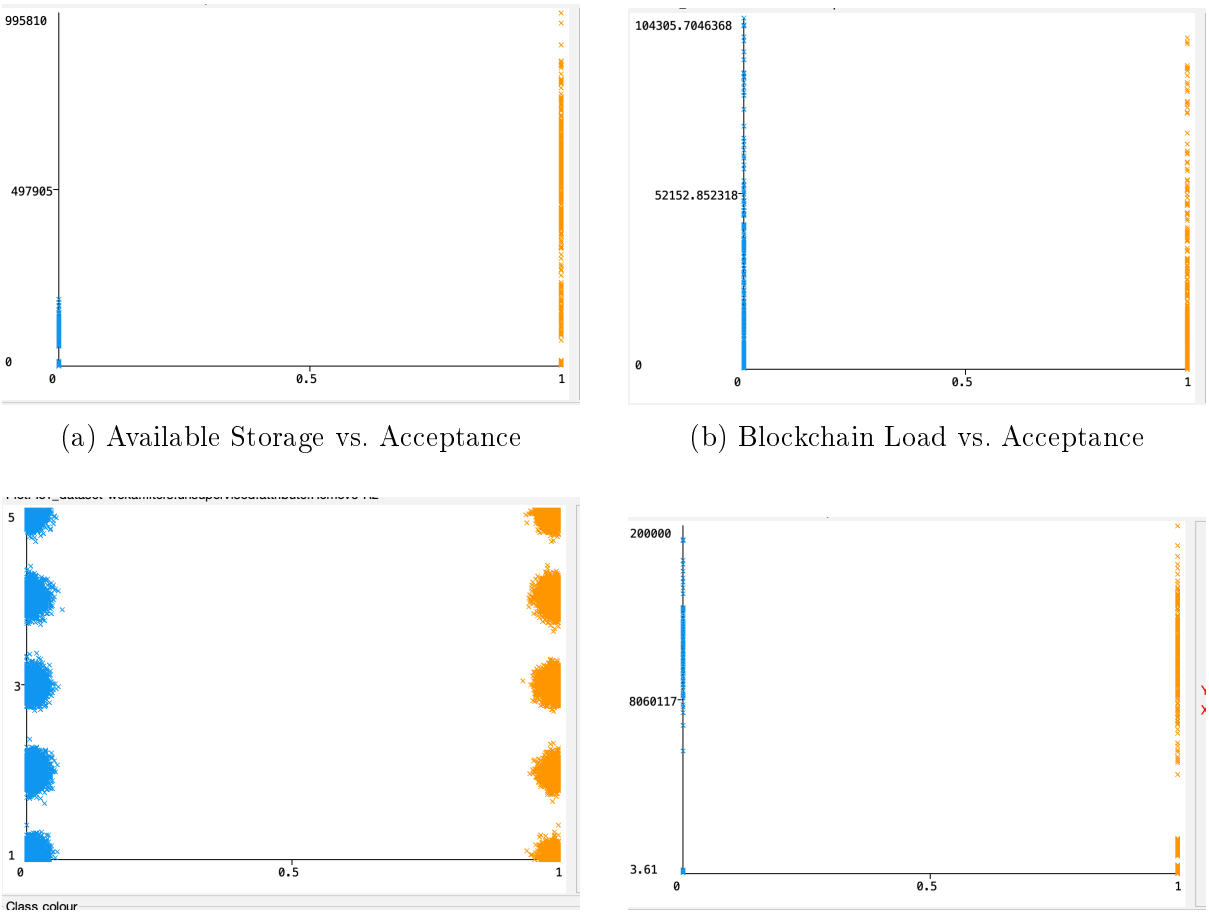
Distribution Analysis.



(a) Available Storage by Acceptance (b) Block Size by Acceptance

Figure 5: Boxplots of Key Features by Target Class (R)

Boxplots by Target Class. The boxplots confirm that accepted nodes (Output=1) tend to have *higher available storage* and *lower block sizes* compared to rejected nodes, aligning with domain expectations.



(a) Available Storage vs. Acceptance (b) Blockchain Load vs. Acceptance

(c) Stability Score vs. Acceptance (d) Projected Growth vs. Acceptance

Figure 6: Feature vs. Acceptance Visualizations (Weka)

Weka EDA: Feature vs. Acceptance.

4.3.3 Models Trained

Three classifiers were trained in R (with 10-fold cross-validation on the 80% training set and evaluated on the 20% hold-out test set), and the same algorithms were independently run in Weka (with 10-fold CV on the full dataset) and JMP (Nominal Logistic on the full dataset) for cross-validation of results.

1. **Logistic Regression (GLM)** — baseline linear classifier.
2. **Decision Tree (RPART / J48)** — interpretable, non-linear classifier.
3. **Random Forest** — ensemble method for robust classification.

4.3.4 Results

Table 5: Model Comparison — R (Test Set)

Model	Accuracy	Kappa	Sensitivity	Specificity	AUC
Logistic Regression (GLM)	0.757	0.517	0.837	0.685	0.761
Decision Tree (RPART)	0.900	0.800	0.916	0.886	0.943
Random Forest	**0.975**	**0.951**	**0.975**	**0.976**	**0.997**

R Results (Test Set: 6,000 observations).

Table 6: Model Comparison — Weka (10-fold CV)

Model	Accuracy	Kappa	Precision	Recall	F1	AUC
Logistic Regression	0.863	0.725	0.863	0.863	0.863	0.939
J48 (Decision Tree)	0.963	0.926	0.963	0.963	0.963	0.974
Random Forest	**0.973**	**0.946**	**0.973**	**0.973**	**0.973**	**0.997**

Weka Results (10-fold CV on 30,000 observations).

Table 7: JMP Nominal Logistic Regression Results

Metric	Value
Entropy RSquare	0.5426
Generalized RSquare	0.7047
Misclassification Rate	0.1363
RASE	0.3128
Mean Abs Dev	0.1982
N	30,000

JMP Results (Nominal Logistic Regression on 30,000 observations). JMP's Nominal Logistic model achieved a misclassification rate of 13.63% (accuracy ≈ 86.4%), which falls between the R GLM (75.7%) and Weka Logistic (86.3%). The Generalized R^2 of 0.70 indicates a reasonable fit.

GLM

		Predicted	
		no	yes
Actual	no	2146	988
	yes	468	2398

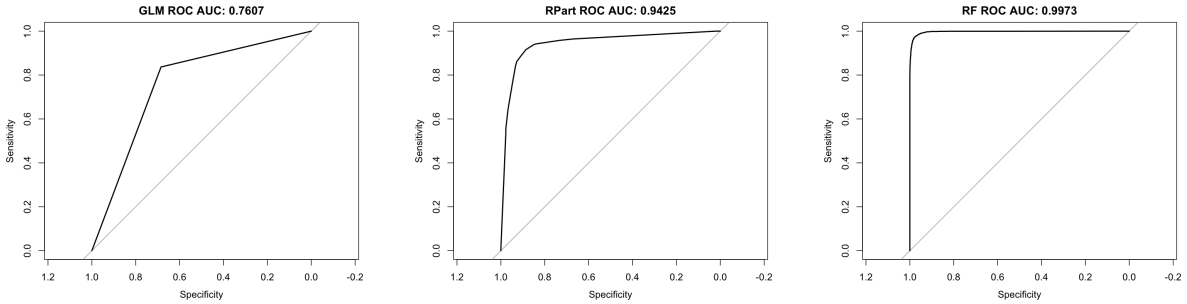
RPART

		Predicted	
		no	yes
Actual	no	2776	358
	yes	242	2624

Random Forest

		Predicted	
		no	yes
Actual	no	3059	75
	yes	73	2793

Confusion Matrices (R — Test Set).



(a) GLM (AUC = 0.761) (b) RPART (AUC = 0.943) (c) RF (AUC = 0.997)

Figure 7: ROC Curves for the Three Models (R)

ROC Curves (R).

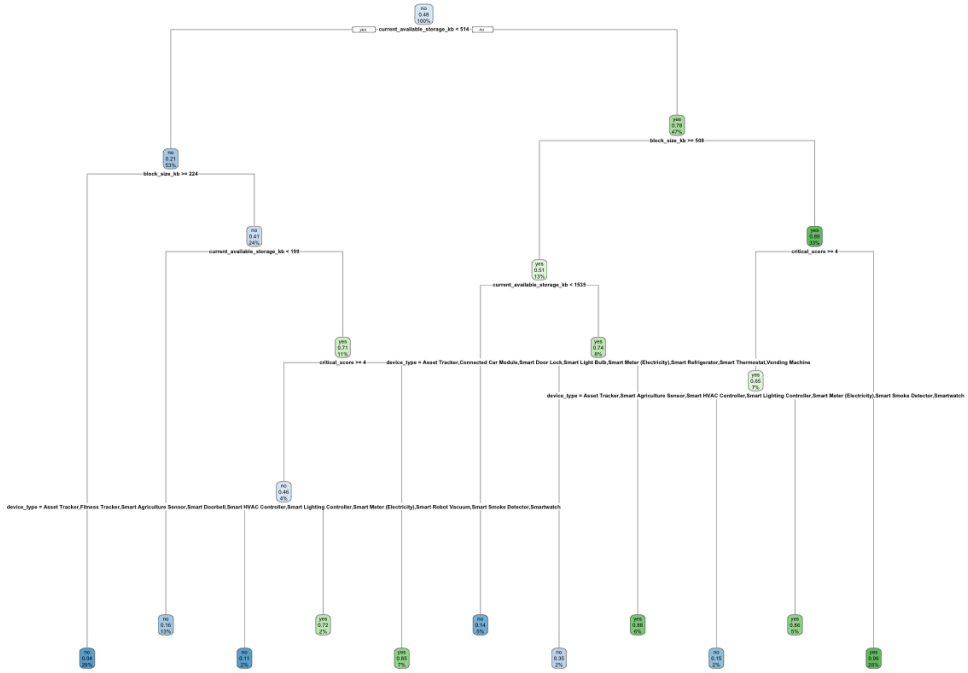


Figure 8: Decision Tree (RPART) — R

Decision Tree Visualization (R).

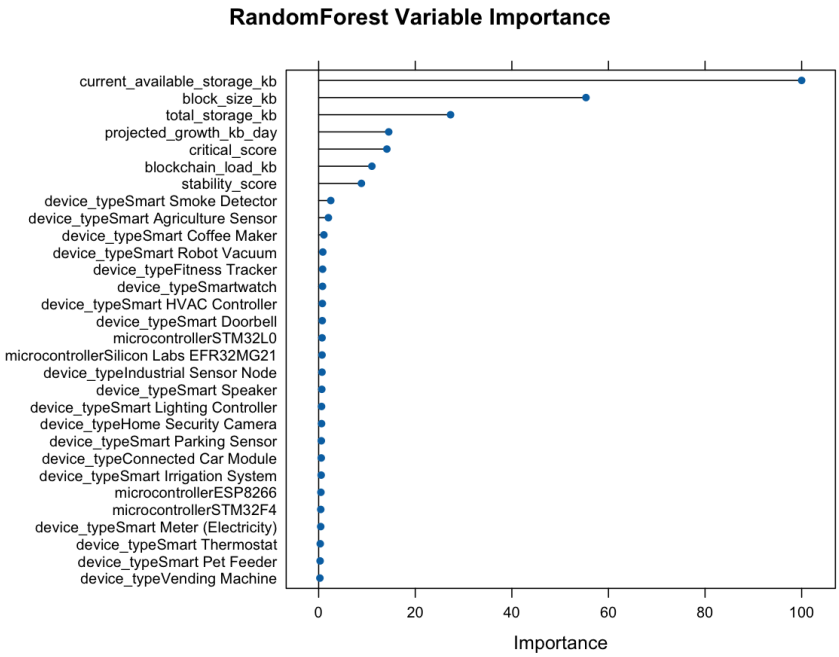
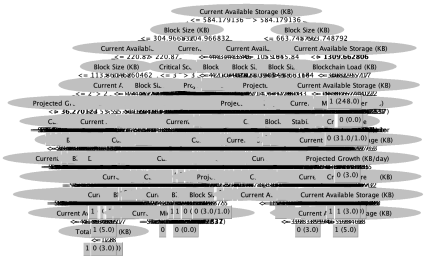


Figure 9: Random Forest Variable Importance (R)

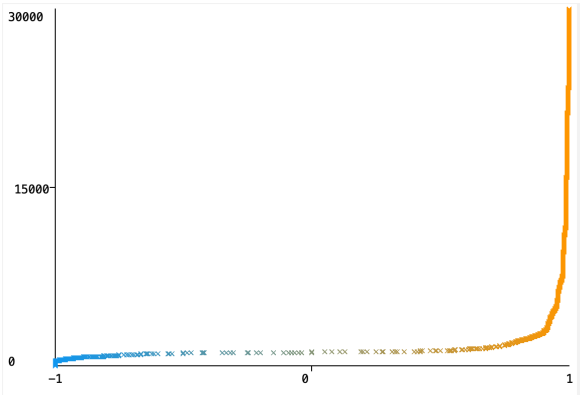
Variable Importance (Random Forest — R). The top predictors identified by the Random Forest are:

1. **Current Available Storage** (importance: 100.0) — the single most critical factor.
2. **Block Size** (55.3) — larger blocks are harder to accommodate.
3. **Total Storage** (27.3) — overall device capacity matters.
4. **Projected Growth** (14.5) — high growth leaves less room for blocks.
5. **Critical Score** (14.1) — critical devices are less likely to accept blocks.
6. **Blockchain Load** (11.0) — existing blockchain data reduces acceptance.
7. **Stability Score** (8.8) — more stable devices are preferred.

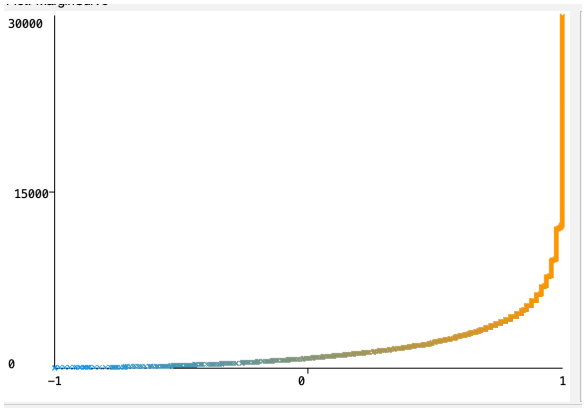
Device type and microcontroller contribute marginally (importance < 3), suggesting that the acceptance decision is primarily driven by *runtime resource state* rather than device category.



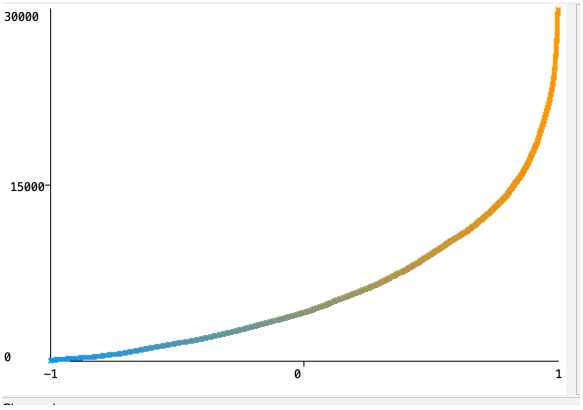
(a) J48 Decision Tree



(b) J48 Margin Curve



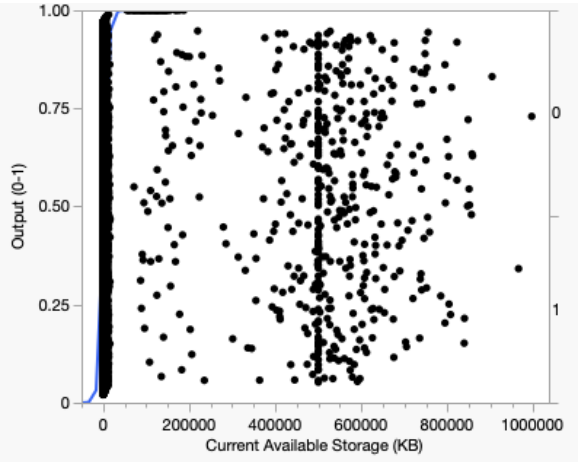
(c) Random Forest Margin Curve



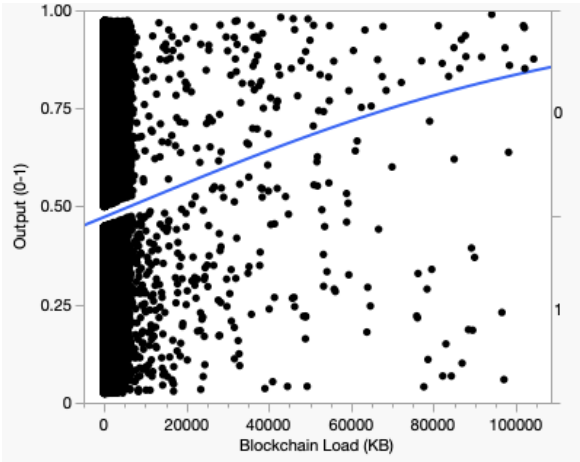
(d) Logistic Regression Margin Curve

Figure 10: Weka Model Outputs

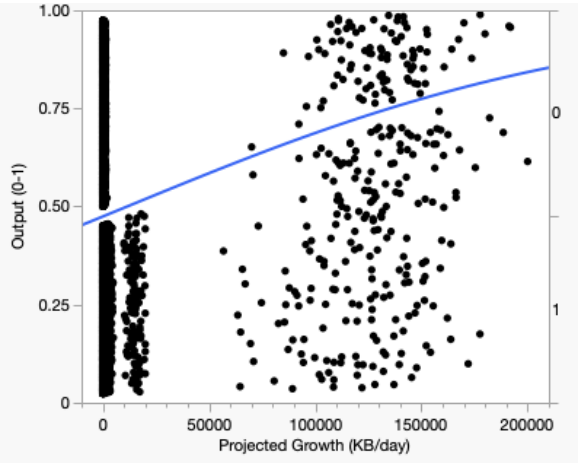
Weka Model Visualizations.



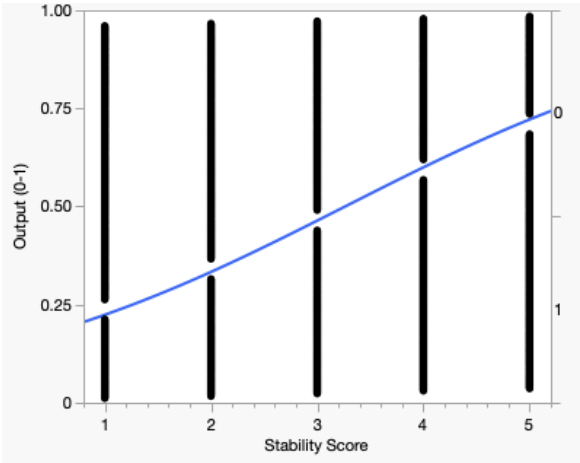
(a) Available Storage vs. Output



(b) Blockchain Load vs. Output



(c) Projected Growth vs. Output



(d) Stability vs. Output

Figure 11: JMP Y-by-X Logistic Plots

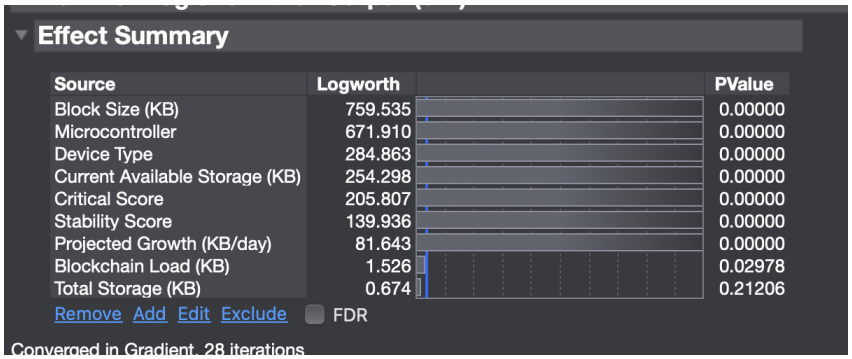


Figure 12: JMP Nominal Logistic Regression Summary

JMP Y-by-X Analysis.

4.4 Novelty of Approach

The novelty of this study lies in the **multi-tool comparative validation** approach:

1. **Cross-tool consistency:** The same dataset was analyzed independently using three different statistical tools (R, Weka, JMP), and results were compared to ensure that findings are robust and not artifacts of a single tool's implementation. The Random Forest achieved AUC ≥ 0.997 in both R and Weka, confirming tool-independent reliability.
2. **End-to-end reproducible pipeline:** The R script (`analyse.r`) provides a fully automated, reproducible pipeline—from data ingestion through EDA to model training and evaluation—which can be re-run on updated data without manual intervention.
3. **Feature importance analysis:** Rather than treating the classifier as a black box, we extracted variable importance from the Random Forest to provide actionable insights for edge blockchain system designers: the acceptance decision is dominated by *current available storage*, not device type or microcontroller architecture.
4. **Data normalization:** We identified and resolved inconsistencies in the microcontroller labels (16 raw categories containing duplicates, normalized to 11 distinct types), improving data quality before modelling.

4.5 Summary and Interpretation

1. **Random Forest is the best-performing model** across all three tools, achieving 97.5% accuracy (R), 97.3% accuracy (Weka), and AUC of 0.997 in both. It significantly outperforms Logistic Regression (75.7–86.4%) and Decision Tree (90.0–96.3%).
2. **Logistic Regression underperforms** on this dataset, likely due to non-linear relationships between features and the target, as well as the high-cardinality one-hot encoded categorical features causing numerical instability (evidenced by the extremely large coefficient estimates in the R GLM output).
3. **Decision Trees provide a good balance** of accuracy (90–96%) and interpretability. The RPART/J48 tree structures reveal that the primary splits are on available storage, block size, and critical score—consistent with domain knowledge.
4. **Current available storage is the single most important predictor** (RF importance = 100), followed by block size (55.3) and total storage (27.3). This makes intuitive sense: a node can only accept a block if it has sufficient free space relative to the block size.
5. **Cross-tool validation confirms robustness:** The ranking of models (RF > DT > LR) is consistent across R, Weka, and JMP, and the absolute performance numbers are closely aligned, demonstrating that the findings are not tool-specific.
6. The **correlation between Critical Score and Stability Score** ($r = -0.50$) is a notable finding: higher-criticality devices tend to have lower stability, which may reflect more demanding or variable operating conditions for critical applications.

4.6 Limitations and Future Scope

4.6.1 Limitations

- **Synthetic data:** The dataset is synthetically generated. While it captures realistic attribute distributions and relationships, real-world IoT deployments may exhibit additional complexities such as network latency, intermittent connectivity, and heterogeneous data formats.
- **Static snapshot:** The dataset represents a single time snapshot. In practice, device states (available storage, blockchain load, stability) change dynamically, requiring online or streaming classification.
- **Limited feature set:** Attributes such as network bandwidth, power consumption, geographic location, and security posture are not included but may influence block acceptance in real systems.
- **GLM instability:** The logistic regression model exhibited numerical instability (extremely large coefficients), suggesting that regularization (e.g., LASSO/Ridge via `glmnet`) or feature engineering could improve linear model performance.
- **Class balance:** The dataset is approximately balanced ($\sim$52% class 0, $\sim$48% class 1), which is favorable for classification but may not reflect real-world class distributions.

4.6.2 Future Scope

- **Real-world validation:** Deploy the Random Forest model on actual IoT testbeds to validate performance with real device data.
- **Temporal modelling:** Extend the analysis to time-series classification, predicting acceptance over future time windows based on historical trends.
- **Deep learning:** Explore neural network architectures (e.g., gradient-boosted trees via XGBoost, or lightweight neural networks) for potential further accuracy gains.
- **Federated learning:** Investigate privacy-preserving federated learning approaches where each IoT node trains a local model and shares only model updates, avoiding raw data transmission.
- **Multi-class extension:** Extend the binary classification to a multi-class or regression problem predicting the *number of blocks* a node can accept or the *optimal block size* for each node.
- **Regularized linear models:** Apply LASSO or Elastic Net regularization to the logistic regression to address the coefficient instability observed in the GLM.