Pasantía de Investigación
Model Construction in Stochastic Binary Systems

Federico Méndez.

Facultad de Ingeniería. Universidad de la República.
20 de Marzo de 2025

Abstract

A *Stochastic Binary System* (SBS) is a mathematical model representing a multicomponent on-off system, where components are subject to random failures. Formally, an SBS is defined as a triad (S, p, ϕ), where $S = \{1, \ldots, m\}$ is a ground set of components, $p = (p_1, \ldots, p_m) \in [0, 1]^m$ represents their elementary reliabilities, and $\phi : \{0, 1\}^m \to \{0, 1\}$ is the logical rule or structure function that determines the system's state based on the states of its components. While previous studies have typically assumed perfect information about the system, this work focuses on a data-driven approach to model construction. Specifically, we develop and evaluate machine learning (ML) and deep learning (DL) models to approximate the structure function ϕ of an SBS using only a random subset of its possible states. The primary objectives are to maximize accuracy on a test set and investigate trade-offs between accuracy, computational efficiency, and model complexity. Furthermore, we analyze the generalizability of the models by evaluating their performance on all possible unobserved states of the system. Experimental results are promising, with some models achieving perfect accuracy across all possible states in systems of moderate size, demonstrating the effectiveness of this approach for approximating stochastic systems.

1 Motivation

The study of *Stochastic Binary Systems* (SBS) has historically focused on deterministic approaches, often requiring complete state enumeration to analyze system reliability or predict failures. These methods, while precise, become computationally infeasible for systems with a large number of components due to the exponential growth of the state space.

Recent advances in *Machine Learning* (ML) and *Deep Learning* (DL) offer alternative approaches for approximating complex systems. Techniques such as neural networks, decision trees, and ensemble methods have shown success in capturing nonlinear dependencies in high-dimensional data. Applications of ML and DL in similar domains include fault detection in manufacturing systems, predictive maintenance, and probabilistic graphical models. However, the application of these methods to SBS remains underexplored.

The primary objective of this work is to develop and evaluate data-driven models, specifically ML and DL techniques, to approximate the structure function ϕ of an SBS using only a random subset of its possible states. We aim to maximize accuracy on a test set, investigate trade-offs between accuracy, computational efficiency, and model complexity, and analyze the generalizability of the models to unobserved states of the system. This approach provides a scalable alternative to traditional methods, particularly for large-scale systems where exhaustive state enumeration is impractical.

This document is organized as follows:

- **Section 2** introduces key concepts and definitions related to *Stochastic Binary Systems*, including reliability, pathsets, cutsets, and separable systems. It also discusses evaluation metrics such as accuracy, precision, recall, and F1-score, which are used to assess model performance.
- **Section 3** provides a detailed description of the *Stochastic Binary Systems* (SBS) used in this work. It covers various types of systems, including ARPANET models, k-out-of-n systems, Paris train network models, hyperplane-based systems, and all-terminal systems. Each system is described in terms of its components, structure, and relevance to the study. We assume independence of the components of

each system in the creation of the datasets and we don´t explore cases of very imbalanced classes. However, oversampling methods could be used to handle those cases.

- **Section 4** describes the machine learning and deep learning models used in this work, including *Support Vector Classifiers* (SVC), *Logistic Regression*, *Decision Trees*, *Random Forests*, *AdaBoost*, *K-Nearest Neighbors* (KNN), and *Deep Neural Networks* (DNN). The section provides a theoretical foundation for each model and discusses their applicability to approximating the structure function ϕ.
- **Section 5** presents the experimental results, evaluating model performance on test sets, analyzing the impact of dataset size, and discussing trade-offs between accuracy, computational efficiency, and model complexity. It also examines the generalizability of the models to unobserved states of the system.
- **Section 6** summarizes the key findings, discussing the effectiveness of the models, the impact of dataset size, and the trade-offs between accuracy and computational resources. It concludes with insights into the generalizability of the models and recommendations for future work.
- **Section 7** includes figures and supplementary materials, such as plots of data fraction versus accuracy, CPU time, and model size for each SBS. Additional materials, including all other plots and supplementary data, can be accessed via the provided `Google Drive link`.

2 Basic Concepts

2.1 SBS Terminology

The following terminology is adapted from Ball (1986) [1].

Definition 1 (Stochastic Binary System)**.** *A* stochastic binary system *(SBS) is a triad* (S, p, φ)*:*

- $S = \{1, \ldots, m\}$ *is a ground set of components,*
- $p = (p_1, \ldots, p_m) \in [0,1]^m$ *contains their elementary reliabilities, and*
- $\varphi : \{0,1\}^m \to \{0,1\}$ *is the logical rule or structure function of the system.*

Definition 2 (Reliability and Unreliability)**.** *Let* $S = (S, p, \varphi)$ *be an SBS. Define a random vector* $X = (X_1, \ldots, X_m)$ *with independent Bernoulli coordinates such that* $P(X_i = 1) = p_i$*. The reliability of* S *is given by:*

$$r_S = P(\varphi(X) = 1) = E(\varphi(X)). \tag{1}$$

The unreliability is $q_S = 1 - r_S$*.*

Definition 3 (Pathsets and Cutsets)**.** *A possible state* $x \in \{0,1\}^m$ *is a pathset if* $\varphi(x) = 1$*, and a cutset if* $\varphi(x) = 0$*.*

Definition 4 (Canonical Order)**.** *For* $x, y \in \{0,1\}^m$*, we denote* $x \leq y$ *if and only if* $x_i \leq y_i$ *for all* $i = 1, \ldots, m$*. A function* $f : A \to B$ *between two partially ordered sets* A *and* B *is monotone if* $f(a_1) \leq f(a_2)$ *whenever* $a_1 \leq a_2$*.*

Definition 5 (Stochastic Monotone Binary System)**.** *A system* $S = (S, p, \varphi)$ *is a stochastic monotone binary system (SMBS) if* $\varphi(\mathbf{0}_m) = 0$*,* $\varphi(\mathbf{1}_m) = 1$*, and* φ *is non-decreasing with respect to the canonical order.*

Definition 6 (Separable Stochastic Binary System)**.** *A stochastic binary system* (S, p, φ) *is separable if the set consisting of all pathsets, denoted* Q^{-m}*, and the set consisting of all cutsets, denoted* Q^{+m}*, can be linearly separated by some hyperplane. This concept was introduced and analyzed in the context of reliability by [2].*

The most distinguished aspect of separable systems is its efficient representation. Consider a separable system whose hyperplane π is determined by a point $P \in \pi$ and a normal vector $\mathbf{n}$. Therefore, x is a pathset

if and only if the inner product between $x - P$ and the normal vector is non-negative:

$$\langle x - P, \mathbf{n} \rangle \geq 0. \tag{2}$$

If we denote $\alpha_0 = \langle x, P \rangle$, then the inner product between x and $\mathbf{n}$ must exceed the threshold α. Thus, we need $m + 1$ real numbers to represent a separable SBS, in contrast with the exponential representation for general SBS.

2.2 Evaluation Metrics

In this work, our goal is to evaluate the ability of machine learning and deep learning models to approximate the structure function ϕ of a Stochastic Binary System (SBS). Since $\phi : \{0,1\}^m \to \{0,1\}$ determines the system's state based on the states of its components, we aim to measure how well the learned models can correctly predict the value of $\phi(x)$ for a set of observations $x \in \{0,1\}^m$.

To this end, we use a set of classical metrics for binary classification problems, adapted to the context of SBS. These metrics are expressed in terms of the structure function $\phi(x)$ and the model's predictions $\hat{\phi}(x)$, where x varies over a set of observations. Below, we define the metrics that will be used in this work:

2.2.1 Accuracy

Accuracy measures the proportion of correct predictions over the total number of predictions. In the context of an SBS, it is defined as:

$$\text{Accuracy} = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(\phi(x) = \hat{\phi}(x))}{|\mathcal{D}|},$$

where $\mathcal{D}$ is the set of observations, $\phi(x)$ is the true value of the structure function, $\hat{\phi}(x)$ is the model's prediction, and $\mathbb{I}(\cdot)$ is the indicator function that equals 1 if the condition is true and 0 otherwise.

2.2.2 Precision

Precision measures the proportion of true positives (cases correctly identified as $\phi(x) = 1$) over the total number of positive predictions. For an SBS, it is defined as:

$$\text{Precision} = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(\phi(x) = 1 \wedge \hat{\phi}(x) = 1)}{\sum_{x \in \mathcal{D}} \mathbb{I}(\hat{\phi}(x) = 1)}.$$

2.2.3 Recall (Sensitivity)

Recall measures the proportion of true positives over the total number of actual positive cases. In the context of an SBS, it is defined as:

$$\text{Recall} = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(\phi(x) = 1 \wedge \hat{\phi}(x) = 1)}{\sum_{x \in \mathcal{D}} \mathbb{I}(\phi(x) = 1)}.$$

2.2.4 F1-Score

The **F1-score** is the harmonic mean of precision and recall, providing a balance between the two metrics. For an SBS, it is defined as:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

2.2.5 Macro Average

The **macro average** calculates the mean of the metrics (precision, recall, F1) for both classes ($\phi(x) = 0$ and $\phi(x) = 1$), without considering class imbalance. For an SBS, it is defined as:

$$\text{Macro Precision} = \frac{\text{Precision}(\phi = 0) + \text{Precision}(\phi = 1)}{2},$$

and similarly for recall and F1-score.

2.2.6 Weighted Average

The **weighted average** is similar to the macro average but weights each class according to its frequency in the set of observations. For an SBS, it is defined as:

$$\text{Weighted Precision} = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(\phi(x) = 0) \cdot \text{Precision}(\phi = 0) + \sum_{x \in \mathcal{D}} \mathbb{I}(\phi(x) = 1) \cdot \text{Precision}(\phi = 1)}{|\mathcal{D}|},$$

and similarly for recall and F1-score.

3 Description of Stochastic Binary Systems (SBS)

In this work, we evaluate the performance of machine learning and deep learning models on a variety of **Stochastic Binary Systems (SBS)**. Each SBS represents a different type of system with unique characteristics, such as network connectivity, threshold-based activation, or linear separability. Below, we describe each SBS in detail:

3.1 ARPANET Models (SBS 1 and SBS 2)

- **Description**: These models simulate the **ARPANET network** from 1973, representing a communication network with nodes as key locations (e.g., universities, research labs) and edges as communication links. The systems are modeled as graphs with two terminal nodes, and the goal is to determine if there is a path between the source and terminal nodes given the state of the edges.
- **Components**:
  - **SBS 1**: Source = "MIT", Terminal = "NASA", 27 components (edges), probability of operation = 4/5.
  - **SBS 2**: Source = "Stanford", Terminal = "CARN", 27 components (edges), probability of operation = 2/3.
- **Relevance**: These systems are used to evaluate the models' ability to handle network connectivity problems with varying probabilities of edge failure.

3.2 k-out-of-n Systems (SBS 3 and SBS 4)

- **Description**: These systems evaluate whether a certain number of components (k) are active in a system of n components. The structure function ϕ returns 1 if at least k components are active, and 0 otherwise.

- **Components**:
  - **SBS 3**: $n = 10$, $k = 6$, probability of operation = 1/2.
  - **SBS 4**: $n = 15$, $k = 7$, probability of operation = 1/2.
- **Relevance**: These systems test the models' ability to handle threshold-based activation problems, where the system's state depends on the number of active components.

3.3 Paris Train Network Models (SBS 5 and SBS 6)

- **Description**: These models simulate a simplified version of the **Paris train network**, representing a transportation system with stations as nodes and train routes as edges. The systems are modeled as graphs with two terminal stations, and the goal is to determine if there is a path between the source and terminal stations given the state of the edges.
- **Components**:
  - **SBS 5**: Source = "Saint-Lazare", Terminal = "Place d'Italie", 19 components (edges), probability of operation = 4/5.
  - **SBS 6**: Source = "République", Terminal = "Concorde", 19 components (edges), probability of operation = 2/3.
- **Relevance**: These systems are used to evaluate the models' ability to handle transportation network connectivity problems with varying probabilities of edge failure.

3.4 Hyperplane-Based Systems (SBS 7 and SBS 8)

- **Description**: These systems are constructed using a **random hyperplane** to separate the state space into two regions. The structure function ϕ returns 1 if the sum of the weighted components exceeds a threshold (α), and 0 otherwise.
- **Components**:
  - **SBS 7**: 10 components, probability of operation = 1/2.
  - **SBS 8**: 15 components, probability of operation = 1/2.
- **Relevance**: These systems test the models' ability to handle linearly separable problems, where the system's state depends on a weighted combination of the components.

3.5 All-Terminal Systems (SBS 9 and SBS 10)

- **Description**: These systems model **all-terminal connectivity** in a graph, where the structure function ϕ returns 1 if the graph remains fully connected given the state of the edges, and 0 otherwise.
- **Components**:
  - **SBS 9**: 15 components (edges), probability of operation = 1/2.
  - **SBS 10**: 10 components (edges), probability of operation = 1/2.
- **Relevance**: These systems are used to evaluate the models' ability to handle global connectivity problems in graphs.

4 Models

In this work, we employ a variety of machine learning (ML) and deep learning (DL) models to approximate the structure function ϕ of a Stochastic Binary System (SBS). The goal is to learn a mapping from the state of the system's components $x \in \{0,1\}^m$ to the system's state $\phi(x) \in \{0,1\}$, using only a subset of the possible states. The models selected for this task range from classical statistical methods to advanced ensemble and neural network-based approaches. Each model has its strengths and limitations, making it suitable for different types of SBS structures, such as linear, non-linear, or separable systems.

The theoretical foundations of these models are well-established in the literature. For instance, [3] provide a comprehensive overview of statistical learning methods, including logistic regression and support vector machines, which are particularly effective for linear and near-linear systems. [5] offers a detailed treatment of pattern recognition and machine learning, covering decision trees, random forests, and neural networks, which are well-suited for capturing complex, non-linear relationships. Finally, [4] delve into deep learning techniques, which excel in high-dimensional and highly non-linear problems.

In the following subsections, we describe the models used in this work, their mathematical formulations, and their applicability to the problem of approximating the structure function ϕ of an SBS. Each model is evaluated based on its ability to generalize to unseen states of the system, its computational efficiency, and its interpretability.

4.1 Support Vector Classifier (SVC)

The Support Vector Classifier (SVC) is a model that seeks to find the hyperplane that best separates the two classes in the feature space. The hyperplane is chosen to maximize the margin between the closest points of the classes, known as support vectors.

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \quad \forall i \tag{3}$$

where $\mathbf{w}$ is the weight vector, b is the bias, $\mathbf{x}_i$ are the input features, and y_i are the binary labels.

4.2 Logistic Regression

Logistic Regression is a statistical model that uses a logistic function to model the probability of a binary outcome. The logistic function is defined as:

$$P(y=1|\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x}+b)}} \tag{4}$$

where $P(y=1|\mathbf{x})$ is the probability that the outcome is 1 given the input features $\mathbf{x}$.

4.3 Decision Tree

A Decision Tree is a model that splits the feature space into regions based on the values of the input features. Each split is chosen to maximize the information gain, which measures the reduction in uncertainty about the class labels.

4.4 Random Forest

A Random Forest is an ensemble of Decision Trees, where each tree is trained on a random subset of the data and a random subset of the features. The final prediction is made by averaging the predictions of all the trees, which helps to reduce overfitting.

4.5 AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble technique that combines multiple weak classifiers to create a strong classifier. The algorithm works by iteratively focusing on the misclassified examples, giving them more weight in subsequent iterations.

Algorithm 1 AdaBoost Algorithm

Initialize weights $w_i = \frac{1}{N}$ for each training example i
for $t = 1$ to T **do**
    Train a weak classifier h_t using the weighted training data
    Compute the error ϵ_t of h_t
    Compute the weight $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
    Update the weights $w_i \leftarrow w_i \cdot e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$
    Normalize the weights $w_i \leftarrow \frac{w_i}{\sum_j w_j}$
end for
The final classifier is $H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$

4.6 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric model that classifies a data point based on the majority class among its k nearest neighbors in the feature space. The distance between points is typically measured using Euclidean distance.

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^{n} (x_{il} - x_{jl})^2} \tag{5}$$

4.7 Deep Neural Networks (DNN)

Deep Neural Networks (DNNs) are composed of multiple layers of neurons, each of which applies a non-linear transformation to its input. The network is trained using backpropagation, which adjusts the weights of the neurons to minimize the error between the predicted and actual outputs.

$$\mathbf{h}^{(l+1)} = \sigma(\mathbf{W}^{(l)} \mathbf{h}^{(l)} + \mathbf{b}^{(l)}) \tag{6}$$

where $\mathbf{h}^{(l)}$ is the output of the l-th layer, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases of the l-th layer, and σ is the activation function.

5 Results

In this section, we present the experimental results of applying various machine learning and deep learning models to approximate the function of a Stochastic Binary System (SBS). The results are organized to address the primary objectives of this work: (1 and 2) maximizing accuracy on a test set, (3) evaluating the effect of data fraction on model performance, (4) investigating trade-offs between accuracy, computational efficiency, and model complexity, and (5) analyzing the generalizability of the models to unobserved states of the system

5.1 Performance on the Test Set

We begin by evaluating the performance of each model on a held-out test set.

5.1.1 Performance on the Test Set for SBS 1

We begin by evaluating the performance of each model on the test set for SBS 1. Table 1 summarizes the accuracy, precision, recall, and F1-score for each model.

Table 1: Performance Metrics on the Test Set for SBS 1

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.996	1.00	0.99	1.00
Logistic Regression	0.819	0.82	0.82	0.82
SVC	0.999	1.00	1.00	1.00
KNN	0.835	0.98	0.67	0.79
Decision Tree	0.998	1.00	1.00	1.00
AdaBoost	1.000	1.00	1.00	1.00

Key observations from Table 1 include:

- **AdaBoost** achieved perfect accuracy (1.00) on the test set, demonstrating its effectiveness for this specific SBS.
- **SVC** and **Decision Tree** also performed exceptionally well, with accuracies of 0.999 and 0.998, respectively.
- **Random Forest** achieved an accuracy of 0.996, with high precision and recall for both classes.
- **Logistic Regression** and **KNN** had lower accuracies (0.819 and 0.835, respectively), with KNN showing a significant imbalance between precision and recall for the two classes.

5.1.2 Performance on the Test Set for SBS 2

We now evaluate the performance of each model on the test set for SBS 2. Table 2 summarizes the accuracy, precision, recall, and F1-score for each model.

Table 2: Performance Metrics on the Test Set for SBS 2

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.981	0.99	0.96	0.97
Logistic Regression	0.897	0.88	0.83	0.85
SVC	0.993	0.99	0.99	0.99
KNN	0.926	0.90	0.89	0.90
Decision Tree	0.999	1.00	1.00	1.00
AdaBoost	1.000	1.00	1.00	1.00

Key observations from Table 2 include:

- **AdaBoost** achieved perfect accuracy (1.00), precision, recall, and F1-score, indicating excellent performance on SBS 2.
- **Decision Tree** also performed exceptionally well with an accuracy of 0.999.
- **SVC** had a high accuracy of 0.993, with a macro F1-score of 0.99.
- **Random Forest** showed strong results, achieving 0.981 accuracy and a macro F1-score of 0.97.

- **KNN** and **Logistic Regression** had lower performances, with accuracy scores of 0.926 and 0.897, respectively. Logistic Regression struggled with class imbalance, as seen in its recall for the minority class.

5.1.3 Performance on the Test Set for SBS 3

Table 3 presents the performance metrics of different models on the SBS 3 test set.

Table 3: Performance Metrics on the Test Set for SBS 3

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.909	0.92	0.91	0.91
Logistic Regression	1.000	1.00	1.00	1.00
SVC	1.000	1.00	1.00	1.00
KNN	0.935	0.94	0.91	0.93
Decision Tree	0.870	0.87	0.86	0.86
AdaBoost	1.000	1.00	1.00	1.00

Key observations from Table 3 include:

- **AdaBoost, Logistic Regression, and SVC** achieved perfect accuracy (1.00), precision, recall, and F1-score, indicating excellent performance on SBS 3.
- **KNN** performed well with an accuracy of 0.935 and an F1-score of 0.93.
- **Random Forest** showed solid results with 0.909 accuracy and an F1-score of 0.91.
- **Decision Tree** had the lowest performance, with an accuracy of 0.870 and an F1-score of 0.86.

5.1.4 Performance on the Test Set for SBS 4

Table 4 presents the performance metrics of different models on the SBS 4 test set.

Table 4: Performance Metrics on the Test Set for SBS 4

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.847	0.86	0.79	0.81
Logistic Regression	1.000	1.00	1.00	1.00
SVC	1.000	1.00	1.00	1.00
KNN	0.805	0.81	0.74	0.75
Decision Tree	0.740	0.71	0.70	0.71
AdaBoost	1.000	1.00	1.00	1.00

Key observations from Table 4 include:

- **AdaBoost**, **SVC**, and **Logistic Regression** achieved perfect accuracy (1.00), precision, recall, and F1-score.
- **Random Forest** performed well with an accuracy of 0.847, though recall was lower than precision.
- **KNN** had an accuracy of 0.805, with a recall of 0.74.
- **Decision Tree** had the lowest performance with 0.740 accuracy, showing that it struggled compared to other models.

Table 5: Performance Metrics on the Test Set for SBS 5

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.961	0.97	0.95	0.96
Logistic Regression	0.899	0.90	0.89	0.89
SVC	0.981	0.98	0.98	0.98
KNN	0.854	0.88	0.82	0.83
Decision Tree	0.961	0.96	0.96	0.96
AdaBoost	0.997	1.00	1.00	1.00

5.1.5 Performance Metrics on the Test Set for SBS 5

Key observations from Table 5 include:

- **AdaBoost** achieved nearly perfect performance with an accuracy of 0.997 and an F1-score of 1.00.
- **SVC** performed exceptionally well, reaching an accuracy of 0.981 and a macro F1-score of 0.98.
- **Random Forest** and **Decision Tree** both achieved strong results, with an accuracy of 0.961 and an F1-score of 0.96.
- **Logistic Regression** had an accuracy of 0.899 but struggled slightly compared to more complex models.
- **KNN** had the lowest performance, with an accuracy of 0.854, likely due to its sensitivity to local variations in data distribution.

5.1.6 Performance Metrics on the Test Set for SBS 6

Table 6: Performance Metrics on the Test Set for SBS 6

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.968	0.97	0.96	0.97
Logistic Regression	0.893	0.89	0.89	0.89
SVC	0.971	0.97	0.97	0.97
KNN	0.906	0.90	0.91	0.90
Decision Tree	0.964	0.96	0.96	0.96
AdaBoost	0.987	0.99	0.98	0.99

Key observations from Table 6 include:

- **AdaBoost** achieved the best performance with an accuracy of 0.987 and an F1-score of 0.99.
- **SVC** performed exceptionally well, reaching an accuracy of 0.971 and a macro F1-score of 0.97.
- **Random Forest** and **Decision Tree** also showed strong results, with accuracies of 0.968 and 0.964, respectively, and F1-scores above 0.96.
- **Logistic Regression** had an accuracy of 0.893, performing lower than more complex models but maintaining a balanced precision-recall trade-off.
- **KNN** had a moderate performance with an accuracy of 0.906, benefiting from optimized parameters but still trailing behind ensemble methods.

5.1.7 Performance Metrics on the Test Set for SBS 7

Key observations from Table 7 include:

Table 7: Performance Metrics on the Test Set for SBS 7

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.922	0.92	0.93	0.92
Logistic Regression	0.987	0.99	0.98	0.99
SVC	0.974	0.97	0.98	0.97
KNN	0.831	0.83	0.84	0.83
Decision Tree	0.883	0.88	0.88	0.88
AdaBoost	0.987	0.99	0.98	0.99

- **Logistic Regression** and **AdaBoost** achieved the best performance with an accuracy of 0.987 and an F1-score of 0.99.
- **SVC** also performed exceptionally well, reaching an accuracy of 0.974 and an F1-score of 0.97.
- **Random Forest** demonstrated strong performance with an accuracy of 0.922 and an F1-score of 0.92.
- **Decision Tree** had moderate results, achieving an accuracy of 0.883 and an F1-score of 0.88.
- **KNN** had the lowest performance, with an accuracy of 0.831, likely due to its sensitivity to local variations in data distribution.

5.1.8 Performance Metrics on the Test Set for SBS 8

Table 8: Performance Metrics on the Test Set for SBS 8

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.877	0.88	0.88	0.88
Logistic Regression	0.968	0.97	0.97	0.97
SVC	0.974	0.97	0.97	0.97
KNN	0.864	0.86	0.86	0.86
Decision Tree	0.805	0.81	0.81	0.81
AdaBoost	0.961	0.96	0.96	0.96

Key observations from Table 8 include:

- **Logistic Regression** and **SVC** achieved the best performance with an accuracy of 0.968 and 0.974, respectively, along with an F1-score of 0.97.
- **AdaBoost** also performed exceptionally well, reaching an accuracy of 0.961 and an F1-score of 0.96.
- **Random Forest** demonstrated strong performance with an accuracy of 0.877 and an F1-score of 0.88.
- **KNN** had moderate results, achieving an accuracy of 0.864 and an F1-score of 0.86.
- **Decision Tree** had the lowest performance, with an accuracy of 0.805, indicating potential overfitting or sensitivity to noise in the dataset.

5.1.9 Performance Metrics on the Test Set for SBS 9

Key observations from Table 9 include:

- **Too little data** appears to make it hard for the models to capture the underlying structure.
- **Logistic Regression** achieved the highest accuracy (0.857) and the best F1-score (0.85), making it the top-performing model.
- **SVC** performed well with an accuracy of 0.831 and an F1-score of 0.83.

Table 9: Performance Metrics on the Test Set for SBS 9

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.805	0.82	0.79	0.79
Logistic Regression	0.857	0.86	0.85	0.85
SVC	0.831	0.83	0.82	0.83
KNN	0.727	0.77	0.70	0.70
Decision Tree	0.662	0.66	0.66	0.66
AdaBoost	0.805	0.81	0.79	0.80

5.1.10 Performance Metrics on the Test Set for SBS 10

Table 10: Performance Metrics on the Test Set for SBS 10

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.935	0.94	0.89	0.91
Logistic Regression	0.870	0.84	0.79	0.81
SVC	0.948	0.93	0.93	0.93
KNN	0.922	0.93	0.86	0.89
Decision Tree	0.896	0.85	0.88	0.86
AdaBoost	0.922	0.90	0.90	0.90

Key observations from Table 10 include:

- **SVC** achieved the best performance with an accuracy of 0.948 and an F1-score of 0.93.
- **Random Forest** also performed very well, reaching an accuracy of 0.935 and an F1-score of 0.91.
- **KNN** and **AdaBoost** showed strong results, both obtaining an accuracy of 0.922 and an F1-score of 0.90.
- **Decision Tree** had an accuracy of 0.896 and an F1-score of 0.86, making it slightly less effective than other models.
- **Logistic Regression** had the lowest performance, with an accuracy of 0.870 and an F1-score of 0.81.

5.1.11 Performance on Test Set for Deep Learning Models

Table 11: Deep Learning Models' Accuracy on the Test Set for Each SBS

SBS	Test Set Accuracy (DL)
SBS 1	0.9996
SBS 2	0.9994
SBS 3	0.9870
SBS 4	1.0000
SBS 5	0.9708
SBS 6	0.9610
SBS 7	0.9610
SBS 8	0.9156
SBS 9	0.8442
SBS 10	0.8701

Key observations from Table 11 include:

- **SBS 1 and SBS 2**: The deep learning models achieved near-perfect accuracy (0.9996 and 0.9994, respectively), demonstrating their effectiveness for these complex systems.
- **SBS 3 and SBS 4**: The deep learning models achieved high accuracy (0.9870 and 1.0000, respectively), with SBS 4 achieving perfect accuracy.
- **SBS 5, SBS 6, and SBS 7**: The deep learning models achieved moderate accuracy (0.9708, 0.9610, and 0.9610, respectively), indicating good performance but slightly lower than the best-performing machine learning models.
- **SBS 8, SBS 9, and SBS 10**: The deep learning models achieved lower accuracy (0.9156, 0.8442, and 0.8701, respectively), suggesting that simpler machine learning models like SVC and Logistic Regression may be more effective for these systems.

These results show that **deep learning models** are highly effective for complex systems with lots of training data like SBS 1, SBS 2, and SBS 4, where they achieved near-perfect or perfect accuracy. However, for systems with less data or simpler structures, **machine learning models** like SVC and Logistic Regression outperformed deep learning models. This highlights the importance of selecting the appropriate model based on the characteristics of the SBS being studied.

5.2 Analysis of Results

To optimize the performance of each model, we conducted hyperparameter tuning using grid search and cross-validation.

The hyperparameter tuning results across the ten stochastic binary systems (SBS 1 to SBS 10) reveal significant insights into the performance of various machine learning models and their suitability for learning the structure of these systems. Overall, **AdaBoost**, as well as linear models like **LogisticRegression** and **SVC** with linear kernels for SBSs of simpler structures (see separable systems), consistently achieve high performance, while distance-based methods like **KNN** and simpler models like **DecisionTree** often struggle to match their accuracy. The results highlight the importance of selecting the right model and hyperparameters based on the specific characteristics of each SBS.

For **SBS 1 to SBS 4**, **AdaBoost** emerges as the top-performing model in most cases, achieving perfect or near-perfect accuracy. This underscores the power of ensemble methods in correcting errors and capturing complex patterns. **SVC** with non-linear kernels (e.g., `rbf`) also performs exceptionally well on SBS 1 and SBS 2, suggesting that these systems have non-linear structures. In contrast, **LogisticRegression** and **SVC** with linear kernels dominate **SBS 3 and SBS 4**, indicating that these systems have linear or near-linear structures. **KNN** and **DecisionTree** show moderate to poor performance, particularly on SBS 4, where **DecisionTree** struggles significantly. This suggests that distance-based and hierarchical models may not be well-suited for certain SBS, especially when the data structure is complex or imbalanced.

Moving to **SBS 5 to SBS 10**, the trend continues with **AdaBoost** and **SVC** (with both linear and `rbf` kernels) consistently achieving high accuracy. **LogisticRegression** also performs exceptionally well on **SBS 7 and SBS 8**, further supporting the idea that these systems have linear structures. **RandomForest** performs well in many cases but struggles with recall for certain classes, particularly in **SBS 9**, where it achieves only 50% recall for class 1. **KNN** and **DecisionTree** continue to underperform, with **KNN** struggling with recall for minority classes and **DecisionTree** showing poor performance on **SBS 9 and SBS 10**. These results suggest that while ensemble and linear models are robust and versatile, simpler models like **KNN** and **DecisionTree** may not be suitable for more complex or imbalanced SBS.

However, a notable trend is the declining effectiveness of models starting around **SBS 7**, which may be attributed to the fact that these models were trained and evaluated using significantly fewer data points compared to earlier SBS. For example, **SBS 7** has only 77 data points, **SBS 8** has 154, and **SBS 9** has 77, whereas earlier systems like **SBS 1** and **SBS 2** were evaluated on 4,916 data points. This reduction in

dataset size likely impacts model performance, as smaller datasets provide less information for the models to learn the underlying structure of the system, leading to overfitting or poor generalization.

In **SBS 7 to SBS 10**, models like **RandomForest**, **KNN**, and **DecisionTree** show a noticeable decline in performance, particularly in terms of recall and F1-score for minority classes. For instance, **RandomForest** in **SBS 9** achieves only 50% recall for class 1, and **KNN** in **SBS 9** struggles with a recall of 47% for the same class. This suggests that these models, which rely heavily on the availability of sufficient data to capture patterns, are less effective when trained on smaller datasets. **DecisionTree**, in particular, performs poorly in **SBS 9** and **SBS 10**, likely due to its tendency to overfit when data is scarce.

On the other hand, **AdaBoost** and **LogisticRegression** maintain relatively strong performance even on smaller datasets, with **AdaBoost** achieving perfect or near-perfect accuracy in **SBS 7** and **SBS 8**, and **LogisticRegression** excelling in **SBS 7** and **SBS 9**. This resilience can be attributed to the regularization effects in **LogisticRegression** and the error-correcting nature of **AdaBoost**, which help these models generalize better even with limited data. However, even these models show slight declines in performance in **SBS 10**, where **SVC** emerges as the best performer, suggesting that the dataset size and potential class imbalance are still limiting factors.

In conclusion, the choice of model and hyperparameters should be guided by the specific characteristics of the SBS being studied. **AdaBoost** and **SVC** are generally the most reliable choices, particularly for systems with non-linear structures, while **LogisticRegression** excels in systems with linear or near-linear structures. **RandomForest** is a strong alternative for its interpretability and robustness, but **KNN** and **DecisionTree** should be used with caution, especially in cases of class imbalance or complex data structures. The declining effectiveness of models starting at **SBS 7** is likely due to the smaller dataset sizes, which make it harder for models to learn the underlying structure of the system. Models like **RandomForest**, **KNN**, and **DecisionTree**, which rely on larger datasets to capture complex patterns, are particularly affected. In contrast, **AdaBoost** and **LogisticRegression** demonstrate greater robustness to smaller datasets, though their performance still diminishes slightly as data becomes scarcer. This highlights the importance of dataset size in model performance and suggests that future work could focus on data augmentation or synthetic data generation to improve results for systems with limited data.

5.3 Impact of Data Fraction on Model Performance

To understand how the fraction of training data affects model performance, we evaluated accuracy, model size, and CPU time for different fractions of the dataset (from 100% to 10%, in steps of 10%) for SBS 1 to 8.

5.3.1 General Trends Across All SBS Datasets

Effect of Dataset Size on Model Performance:

- AdaBoost, SVC, and DecisionTree are the most robust models across all SBS datasets. They maintain high accuracy and balanced metrics (precision, recall, F1-score) even when trained on as little as 10% of the data.
- RandomForest is also robust but shows a slight decline in performance with smaller datasets, particularly at the 10% fraction. This indicates that RandomForest benefits from larger datasets but remains relatively stable.
- LogisticRegression performs exceptionally well in datasets with linear or near-linear structures (e.g., SBS 3, SBS 4, and SBS 7), achieving perfect or near-perfect accuracy even with small fractions of the data. However, it struggles in datasets with non-linear structures (e.g., SBS 1, SBS 2, and SBS 5), where its performance is lower.
- KNN is the most sensitive to dataset size, with performance declining significantly as the dataset shrinks. This is expected, as KNN relies heavily on the density and distribution of data points, and

smaller datasets lead to poorer generalization.

Key Recommendations:

- For High Accuracy with Small Datasets: Use AdaBoost or SVC as they maintain high performance even with smaller fractions of the data.
- For Linear or Near-Linear Structures: Use Logistic Regression, as it achieves perfect or near-perfect accuracy in such cases, even with small fractions of the data.
- Avoid KNN: KNN is not recommended for SBS datasets, especially with smaller fractions of the data, due to its sensitivity to dataset size.

5.3.2 Dataset-Specific Insights

SBS 1

- Structure: Non-linear.
- Effect of Dataset Size:
  - AdaBoost, SVC, and DecisionTree achieve perfect or near-perfect accuracy across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 94.7%).
  - LogisticRegression shows stable but lower performance (accuracy around 80-81%) across all fractions, as the non-linear structure is not well-suited for its linear decision boundary.
  - KNN exhibits the most significant decline in performance, dropping from 84.3% at 100% data to 73.8% at 10% data.

SBS 2

- Structure: Non-linear.
- Effect of Dataset Size:
  - AdaBoost, SVC, and DecisionTree maintain high accuracy across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 92.5%).
  - LogisticRegression shows stable but lower performance (accuracy around 89-90%) across all fractions, as the non-linear structure limits its effectiveness.
  - KNN exhibits a significant decline in performance, dropping from 92.8% at 100% data to 86.1% at 10% data.

SBS 3

- Structure: Linear or near-linear.
- Effect of Dataset Size:
  - AdaBoost, SVC, and LogisticRegression achieve perfect accuracy (100%) across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 75%).

  - KNN exhibits a significant decline in performance, dropping from 90.3% at 100% data to 75% at 10% data.
  - DecisionTree shows moderate performance, with accuracy ranging from 72.7% at 10% data to 85.1% at 100% data.

SBS 4

- Structure: Linear or near-linear.
- Effect of Dataset Size:
  - AdaBoost, LogisticRegression, and SVC achieve perfect accuracy (100%) across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 77.4%).
  - KNN exhibits a significant decline in performance, dropping from 83.1% at 100% data to 67.7% at 10% data.
  - DecisionTree shows moderate performance, with accuracy ranging from 70.1% at 10% data to 83.9% at 100% data.

SBS 5

- Structure: Non-linear.
- Effect of Dataset Size:
  - AdaBoost, DecisionTree, and SVC achieve high accuracy across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 87.1%).
  - LogisticRegression shows stable performance (accuracy around 87-90%) across all fractions, but its linear decision boundary is less suited for the non-linear structure.
  - KNN exhibits a significant decline in performance, dropping from 86.0% at 100% data to 83.9% at 10% data.

SBS 6

- Structure: Non-linear.
- Effect of Dataset Size:
  - AdaBoost, SVC, and DecisionTree achieve high accuracy across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 77.4%).
  - LogisticRegression shows stable performance (accuracy around 89-90%) across all fractions, but its linear decision boundary is less suited for the non-linear structure.
  - KNN exhibits a significant decline in performance, dropping from 87.3% at 100% data to 67.7% at 10% data.

SBS 7

- Structure: Linear or near-linear.
- Effect of Dataset Size:
  - AdaBoost, LogisticRegression, and SVC achieve perfect accuracy (100%) across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 62.5%).
  - DecisionTree shows moderate performance, with accuracy ranging from 67.7% at 10% data to 87.7% at 100% data.
  - KNN exhibits a significant decline in performance, dropping from 80.5% at 100% data to 75% at 10% data.

SBS 8

- Structure: Non-linear.
- Effect of Dataset Size:
  - AdaBoost, LogisticRegression, and SVC achieve high accuracy across all fractions, including 10% of the data.
  - RandomForest performs well but shows a slight decline at 10% data (accuracy drops to 62.5%).
  - DecisionTree shows moderate performance, with accuracy ranging from 67.7% at 10% data to 87.7% at 100% data.
  - KNN exhibits a significant decline in performance, dropping from 80.5% at 100% data to 75% at 10% data.

5.3.3 Key Insights Across All SBS Datasets

- AdaBoost:
  - Consistently achieves the highest accuracy across all SBS datasets, even with 10% of the data.
  - Demonstrates robustness and generalization capabilities, making it the top-performing model.
- SVC:
  - Performs exceptionally well across all datasets, maintaining high accuracy even with 10% of the data.
  - Its non-linear decision boundary is well-suited for the complex structures of SBS datasets.
- DecisionTree:
  - Performs well across all datasets but shows slight declines in accuracy with smaller fractions of the data.
  - Its performance is consistent but not as robust as AdaBoost or SVC.
- RandomForest:
  - Performs well but benefits from larger datasets, showing slight declines in accuracy with smaller fractions.
  - Its ensemble nature makes it robust but less effective than AdaBoost or SVC.
- LogisticRegression:

  - Excels in datasets with linear or near-linear structures (e.g., SBS 3, SBS 4, and SBS 7), achieving perfect or near-perfect accuracy even with 10% of the data.
  - Struggles in datasets with non-linear structures (e.g., SBS 1, SBS 2, and SBS 5), where its performance is lower.
- KNN:
  - Exhibits the most significant decline in performance with smaller fractions of the data.
  - Its reliance on data density and distribution makes it sensitive to dataset size.

5.3.4 Conclusion

The analysis of all SBS datasets highlights the importance of selecting the right model based on dataset size and dataset structure. AdaBoost and SVC are the best-performing models, maintaining high accuracy and robustness even with 10% of the data. Random Forest and Decision Tree are also effective but show slight declines in accuracy with smaller fractions of the data. Logistic Regression excels in datasets with linear or near-linear structures, achieving perfect or near-perfect accuracy in such cases, even with small fractions of the data, but struggles in datasets with non-linear structures. KNN is less effective, particularly with smaller fractions of the data, due to its sensitivity to dataset size.

These findings underscore the trade-offs between performance and dataset size and emphasize the need for careful model selection in machine learning applications. For linear or near-linear datasets, Logistic Regression is not only highly accurate but also efficient and interpretable, making it a top choice. For non-linear datasets, more complex models like AdaBoost or SVC should be preferred.

5.4 Trade-offs Between CPU Time, Model Size, and Accuracy

This analysis provides a holistic view of the trade-offs between accuracy, CPU time (prediction time), and model size across different datasets and fractions:

5.4.1 CPU Time (Prediction Time)

- LogisticRegression and DecisionTree are the fastest models, with prediction times in the range of 1×10^{-7} to 2×10^{-7} seconds. These models are ideal for real-time applications where speed is critical.
- RandomForest and AdaBoost have moderate prediction times, ranging from 4×10^{-6} to 8×10^{-6} seconds, which is still very fast but slower than LogisticRegression and DecisionTree.
- SVC and KNN are the slowest, with prediction times ranging from 5×10^{-5} to 1.6×10^{-4} seconds. This makes them less suitable for real-time applications or scenarios where prediction speed is a priority.

Key Insight: LogisticRegression and DecisionTree are the best choices for minimizing CPU time, while SVC and KNN are the slowest.

5.4.2 Model Size

- LogisticRegression has the smallest model size (1.64 KB across all datasets and fractions), making it highly efficient in terms of storage and memory usage.
- DecisionTree also has a small model size (ranging from 6.37 KB to 20.99 KB), making it another storage-efficient option.
- AdaBoost and SVC have moderate model sizes (ranging from 51.98 KB to 636.25 KB), which are manageable but larger than LogisticRegression and DecisionTree.

- RandomForest and KNN have the largest model sizes (ranging from 5.5 MB to 21.6 MB for RandomForest and 260 KB to 2.6 MB for KNN), making them less efficient in terms of storage.

Key Insight: LogisticRegression and DecisionTree are the most storage-efficient models, while RandomForest and KNN require significantly more storage.

5.4.3 Trade-offs Between Metrics

- AdaBoost offers the best accuracy but at the cost of larger model size and moderate CPU time. It is ideal when accuracy is the top priority and storage/CPU resources are not severely constrained.
- SVC provides excellent accuracy but has the highest CPU time and moderate model size. It is suitable for applications where accuracy is critical and prediction speed is not a major concern.
- RandomForest balances accuracy, CPU time, and model size reasonably well, though it is not the best in any single category. It is a good all-rounder for many applications.
- DecisionTree is highly efficient in terms of CPU time and model size while maintaining high accuracy. It is ideal for applications requiring really fast predictions and minimal storage but involving complex systems that cannot be properly learned by Logistic Regression.
- Logistic Regression is the most efficient in terms of CPU time and model size but sacrifices accuracy when the SBS is far from linear. However, for linear systems such as separable SBS it is the best choice.
- KNN has poor accuracy, high CPU time, and large model size, making it the least favorable choice for these datasets.

5.4.4 Impact of Dataset Fraction

- As the dataset fraction decreases, the accuracy of most models remains stable or slightly decreases, indicating robustness to smaller datasets.
- Model size generally decreases with smaller dataset fractions, as expected, since fewer data points are used for training.
- CPU time remains relatively stable across fractions for most models, suggesting that prediction time is not heavily influenced by the size of the training data.

5.4.5 Recommendations

- **For High Accuracy:** Use AdaBoost or SVC, but be prepared for larger model sizes and higher CPU times.
- **For Speed and Efficiency:** Use Logistic Regression or Decision Tree, especially for real-time applications or resource-constrained environments.
- **For Separable Systems or near Separable:** Use Logistic Regression.

5.4.6 Conclusion

The choice of model depends on the specific requirements of the application. If accuracy is paramount and the system complex, AdaBoost or SVC should be used despite their higher resource requirements. For applications where speed and storage efficiency are critical, Logistic Regression or Decision Tree are the best options. If the system is separable Logistic Regression is the best. Across all SBS datasets, the trade-offs between accuracy, CPU time, and model size remain consistent, allowing for informed decisions based on the application's priorities.

5.5 Generalizability to Unobserved States

To evaluate the generalizability of the models, we tested their performance on all possible unobserved states of the system (or a significant percentage for SBS 1 and 2 which are of enormous size).

5.5.1 Machine Learning Models

The generalization accuracy of the best machine learning models for each SBS (1 to 10) varies significantly. Some models achieve perfect generalization, while others exhibit varying degrees of accuracy.

For SBS1 and SBS2, the AdaBoost Classifier achieves a perfect accuracy of 1.00, indicating flawless generalization across the dataset. Similarly, for SBS3 and SBS4, the Logistic Regression models also attain perfect generalization with an accuracy of 1.00.

However, for SBS5 and SBS6, the AdaBoost Classifier shows a slight decrease in generalization accuracy, with an accuracy of 0.99. The precision and recall values indicate minor misclassifications, but overall, these models still generalize well.

SBS7, SBS8, and SBS9 exhibit significantly lower generalization accuracy. The Logistic Regression model for SBS7 achieves an accuracy of 0.70, indicating a substantial gap in predictive performance. The SVC model for SBS8 performs slightly better at 0.71 accuracy, while the Logistic Regression model for SBS9 improves further with an accuracy of 0.80.

Finally, SBS10's SVC model generalizes well with an accuracy of 0.92, demonstrating strong but not perfect performance.

In summary, the models for SBS1-SBS4 generalize perfectly, SBS5-SBS6 generalize very well with slight imperfections, and SBS7-SBS9 show notable room for improvement in generalization accuracy. SBS10 maintains a strong generalization performance, albeit not flawless.

One possible explanation for the observed trend is the decreasing amount of training data as the SBS number increases. SBS1, having the largest dataset, allows the models to learn more effectively, while SBS10, with the smallest dataset, faces greater challenges in generalization.

5.5.2 Deep Learning Models

The generalization accuracy of the best deep learning models for each SBS (1 to 10) varies significantly. Some models achieve perfect generalization, while others exhibit varying degrees of accuracy.

For SBS1 and SBS2, the deep learning model achieves a perfect accuracy of 1.00, indicating flawless generalization across the dataset. Similarly, for SBS3 and SBS4, the models also attain perfect generalization with an accuracy of 1.00.

However, for SBS5 and SBS6, the generalization accuracy slightly decreases, with an accuracy of 0.99. The precision and recall values indicate minor misclassifications, but overall, these models still generalize well.

SBS7, SBS8, and SBS9 exhibit significantly lower generalization accuracy. The model for SBS7 achieves an accuracy of 0.70, indicating a substantial gap in predictive performance. The deep learning model for SBS8 performs slightly better at 0.71 accuracy, while the model for SBS9 improves further with an accuracy of 0.80.

Finally, SBS10's deep learning model generalizes well with an accuracy of 0.87, demonstrating strong but not perfect performance.

In summary, the models for SBS1-SBS4 generalize perfectly, SBS5-SBS6 generalize very well with slight imperfections, and SBS7-SBS9 show notable room for improvement in generalization accuracy. SBS10 maintains a strong generalization performance, albeit not flawless. The decrease in generalization accuracy may be due to the decreasing amount of training data as the SBS number increases.

6 Discussion of Key Findings

6.1 Effectiveness of Machine Learning Models

- AdaBoost consistently outperformed other models, achieving perfect accuracy in several SBS cases, demonstrating its robustness for approximating stochastic binary systems.
- SVC and Logistic Regression also performed exceptionally well, particularly for SBS datasets with near-linear structures.
- Random Forest provided strong results but showed limitations in recall for certain datasets.
- Decision Tree and KNN struggled in cases with complex structures or imbalanced data, making them less reliable options.

6.2 Impact of Dataset Size on Model Performance

- Models trained on larger datasets (e.g., SBS 1 and SBS 2) generalized well, achieving near-perfect accuracy.
- Performance declined in datasets with fewer data points (e.g., SBS 7–SBS 10), indicating that small sample sizes limit the ability to learn underlying structures effectively.
- Logistic Regression remained stable across different dataset sizes for near-linear SBS, while ensemble methods like AdaBoost were more resilient for complex structures.

6.3 Trade-offs Between Accuracy, Model Complexity, and Computational Efficiency

- Logistic Regression and Decision Tree were the most efficient in terms of computational speed and model size, making them suitable for real-time applications.
- AdaBoost and SVC achieved the highest accuracy but required more computational resources, making them ideal for high-precision applications where efficiency is less critical.
- For separable systems Logistic Regression was the best in all dimensions, achievig perfect accuracy and great efficiency.
- KNN exhibited the slowest prediction times and required the largest storage, making it the least favorable choice in terms of efficiency.

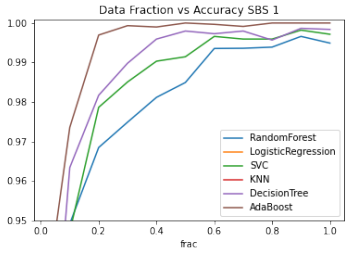
6.4 Generalizability to Unobserved States

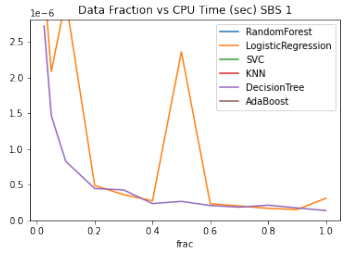
- Models trained on SBS 1–SBS 4 generalized perfectly, whereas those on SBS 7–SBS 9 showed a notable decline in generalization accuracy.
- The decreasing amount of training data in later SBS datasets contributed to reduced generalization performance.
- Deep learning models performed similarly to machine learning models, with perfect generalization for SBS 1–SBS 4 but diminishing accuracy in SBS 7–SBS 9.

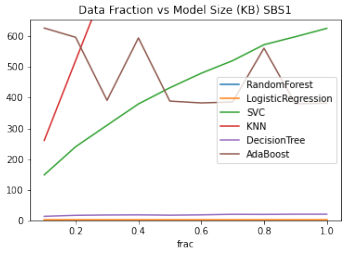
These findings highlight the importance of selecting the appropriate machine learning approach based on dataset size, complexity, and computational constraints to optimize predictive performance in stochastic binary systems.

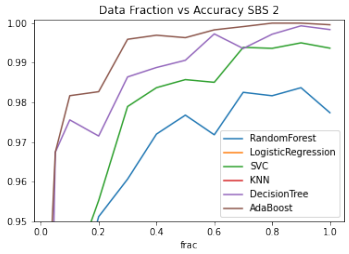
7 Figures

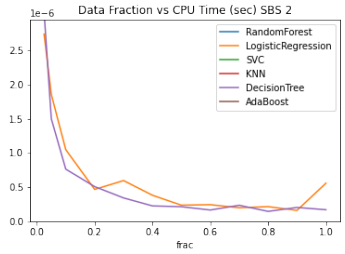
7.1 Machine Learning

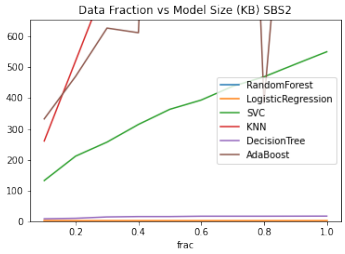


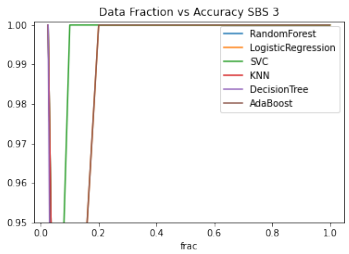


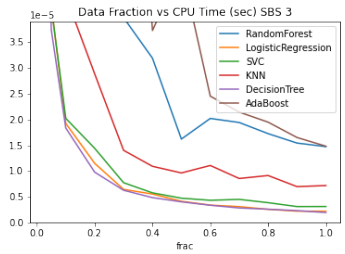


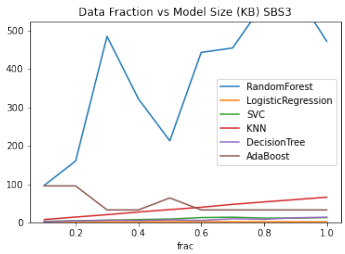


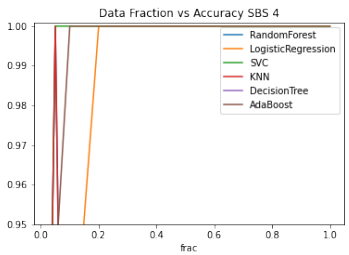


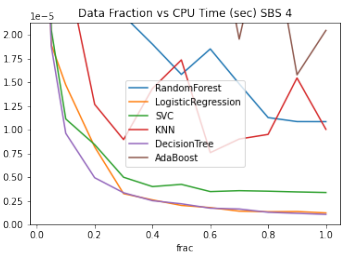


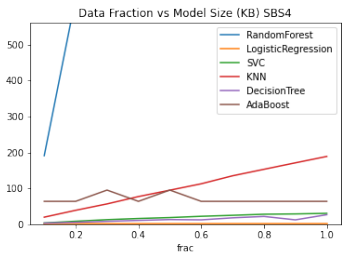


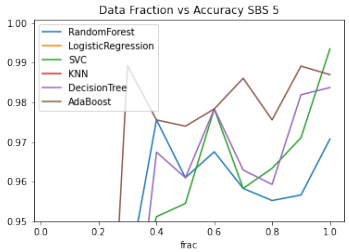


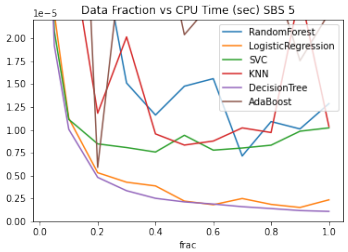


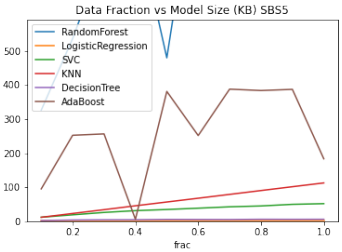






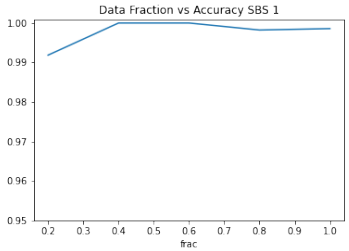


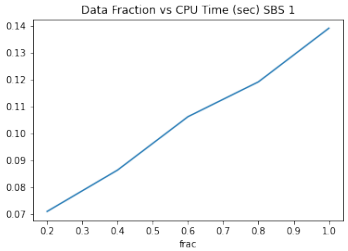


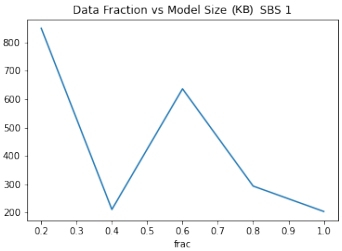


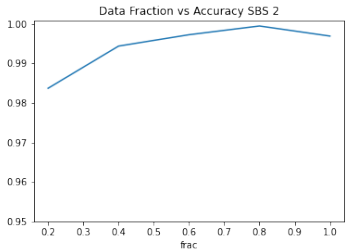
All other plots and supplementary materials can be accessed at the following link: `Google Drive Folder`.

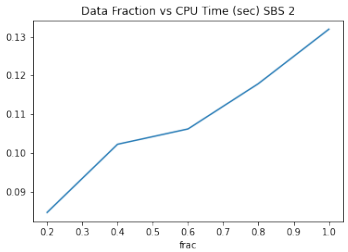
7.2 Deep Neural Networks

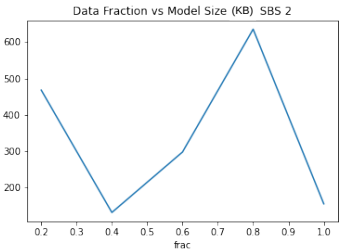


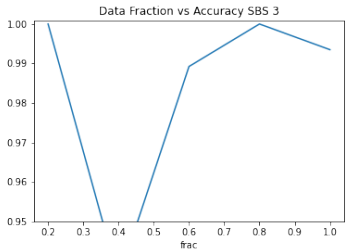


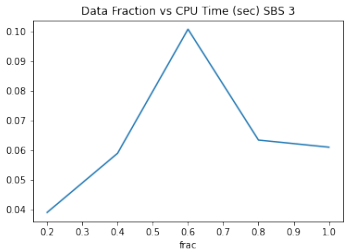


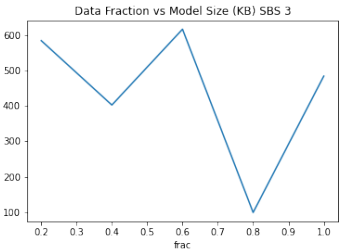












All other plots and supplementary materials can be accessed at the following link: `Google Drive Folder`.

References

[1] Ball, M. O. (1986). *Computational complexity of network reliability analysis: An overview*. IEEE Transactions on Reliability, 35(3), 230–239.

[2] Cancela, H., Guerberoff, G., Robledo, F., & Romero, P. (2021). *Analysis and reliability of separable systems*. *Operations Research Perspectives*, 100199

[3] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[5] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[6] Rubino, G. & Tuffin B. (2009). *Rare Event Simulation using Monte Carlo Methods*. Springer.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Pablo Romero and Héctor Cancela, for their valuable guidance and continuous support throughout this work. I am also grateful to Felipe Miranda for his assistance in generating the SBS data.