2020

# A PROBABILISTIC MACHINE LEARNING FRAMEWORK FOR CLOUD RESOURCE SELECTION ON THE CLOUD

Syeduzzaman Khan
*University of the Pacific*

# A PROBABILISTIC MACHINE LEARNING FRAMEWORK FOR CLOUD RESOURCE SELECTION ON THE CLOUD

By

Syeduzzaman Khan

A Thesis Submitted to the

Graduate School

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

School of Engineering and Computer Science
Computer Engineering

University of the Pacific
Stockton, California

2020

# A PROBABILISTIC MACHINE LEARNING FRAMEWORK FOR CLOUD RESOURCE SELECTION ON THE CLOUD

By

Syeduzzaman Khan

APPROVED BY:

Thesis Advisor: Vivek Krishnamani Pallipuram, Ph.D.

Committee Member: Jinzhu Gao, Ph.D.

Committee Member: David Mueller, Ph.D.

Department Chair: Elizabeth Basha, Ph.D.

# DEDICATION

This thesis is dedication to my son ‘Ahanf Khan’, who have brought a lot of joy in our life. I also dedicate this thesis to my parents and my wife.

# ACKNOWLEDGMENTS

My gratitude goes to Dr. Vivek K. Pallipuram for his hours of patience and guidance while I struggled through the master thesis work. As my mentor, he has taught more than I could ever thank him.

I would like to thank my committee members Dr. Jinzhu Gao and Dr. David Mueller for their amazing support. I would like to thank my family for their unconditional support.

# A PROBABILISTIC MACHINE LEARNING FRAMEWORK FOR CLOUD RESOURCE SELECTION ON THE CLOUD

## Abstract

By Syeduzzaman Khan

University of the Pacific
2020

The execution of the scientific applications on the Cloud comes with great flexibility, scalability, cost-effectiveness, and substantial computing power. Market-leading Cloud service providers such as Amazon Web service (AWS), Azure, Google Cloud Platform (GCP) offer various general purposes, memory-intensive, and compute-intensive Cloud instances for the execution of scientific applications. The scientific community, especially small research institutions and undergraduate universities, face many hurdles while conducting high-performance computing research in the absence of large dedicated clusters. The Cloud provides a lucrative alternative to dedicated clusters, however a wide range of Cloud computing choices makes the instance selection for the end-users. This thesis aims to simplify Cloud instance selection for end-users by proposing a probabilistic machine learning framework to allow to users select a suitable Cloud instance for their scientific applications.

This research builds on the previously proposed A2Cloud-RF framework that recommends high-performing Cloud instances by profiling the application and the selected Cloud instances. The framework produces a set of objective scores called the A2Cloud scores, which denote the compatibility level between the application and the selected Cloud instances. When used alone, the A2Cloud scores become increasingly unwieldy with an increasing number of tested Cloud instances. Additionally, the framework only examines the raw application

performance and does not consider the execution cost to guide resource selection. To improve the usability of the framework and assist with economical instance selection, this research adds two Naive Bayes (NB) classifiers that consider both the application's performance and execution cost. These NB classifiers include: 1) NB with a Random Forest Classifier (RFC) and 2) a standalone NB module.

Naive Bayes with a Random Forest Classifier (RFC) augments the A2Cloud-RF framework's final instance ratings with the execution cost metric. In the training phase, the classifier builds the frequency and probability tables. The classifier recommends a Cloud instance based on the highest posterior probability for the selected application.

The standalone NB classifier uses the generated A2Cloud score (an intermediate result from the A2Cloud-RF framework) and execution cost metric to construct an NB classifier. The NB classifier forms a frequency table and probability (prior and likelihood) tables. For recommending a Cloud instance for a test application, the classifier calculates the highest posterior probability for all of the Cloud instances. The classifier recommends a Cloud instance with the highest posterior probability.

This study performs the execution of eight real-world applications on 20 Cloud instances from AWS, Azure, GCP, and Linode. We train the NB classifiers using 80% of this dataset and employ the remaining 20% for testing. The testing yields more than 90% recommendation accuracy for the chosen applications and Cloud instances. Because of the imbalanced nature of the dataset and multi-class nature of classification, we consider the confusion matrix (true positive, false positive, true negative, and false negative) and F1 score with above 0.9 scores to describe the model performance.

The final goal of this research is to make Cloud computing an accessible resource for conducting high-performance scientific executions by enabling users to select an effective Cloud instance from across multiple providers.

TABLE OF CONTENTS

# LIST OF TABLES

Table

# LIST OF FIGURES

Figure

# CHAPTER 1: INTRODUCTION

High-Performance Computing (HPC) is now widely prevalence in several scientific research areas including computer science, quantum chemistry, physics, image processing, and among others [1]. HPC helps the researchers to reduce the application run time by executing the application on the HPC clusters. These clusters require on-premise power and regular maintenance, leading to aggregate costs. Cloud computing offers cost-effective, scalable, and sustainable Cloud resources for high-performance application execution, obviating the need for on-site HPC systems [2], [3].

Leading Cloud vendors such as Amazon Web Service (AWS) [4], Microsoft Azure [5], Google Cloud Platform [6], IBM Cloud [7], Oracle Cloud [8], Alibaba Cloud [9], and Linode [10] offer a wide range of Cloud Computing instances to the scientific community. Most of the Cloud service providers offer three broad instance categories: general purpose, compute-intensive, and memory-intensive instances with varying memory configurations and price per hour. The abundant Cloud instance configurations and pricing options overwhelm the scientist, making the selection difficult. To alleviate the issue, we propose a machine learning approach to guide the research community to select a high-performing, cost-effective Cloud instance for these applications.

We present a machine learning approach to recommend the Cloud instances for executing scientific applications on the Cloud. We propose two methodologies: Naive Bayes with Random Forest Classifier (NB-Next) and a standalone Naive Bayes (S-NB) built upon our previously proposed A2Cloud-RFC Framework [11].

The A2Cloud-RFC framework consists of the A2Cloud-ext model and the Random forest classifier (RFC). The A2Cloud-ext profiles the application and Cloud instance. The A2Cloud-ext uses hardware benchmarks to profile the application's performance parameters including the number of single-precision floating-point operations(SPFLOPs), number of double-precision floating-point operations (DPFLOPs), the total number of x87 instructions (x87), number of main memories reads and writes (mem), and number of disks reads (disk read), and writes (disk write). The framework also computes the Cloud instance characteristics including single-precision floating-point per second (SPFLOPS), double-precision floating-point operations per second (DPFLOPS), and the total number of x87 instructions (x87S), main memory bandwidth, disk write and read bandwidths. The A2Cloud framework generates the A2Cloud score using the application and the Cloud instance performance parameters. The A2Cloud score denotes to the level of match between the application and the Cloud instance. In addition, the framework uses vendor-specific cost models to form a cost score. The cost score represents the level of economical match between the application and the Cloud instance. The framework stores the A2Cloud and cost scores in a database for future analysis. Using the profiled data (A2Cloud score and cost score), A2Cloud-RFC creates the multiple decision trees where the nodes of the decision trees are assigned numerical rating from 1 to 4. The average ratings of decision trees are the final RFC and RFC and cost ratings. The RFC rating and cost ratings represent the match between the application and the target instance; the higher the rating, the better the match.

The NB-Next uses the RFC rating and cost rating to recommend Cloud instances. NB-Next first uses the K-Means clustering technique to produce four clusters (excellent, good, average, and bad) using the RFC rating and cost rating. This clustering is used for the NB model training.

The S-NB uses the A2Cloud score and cost score versus the RFC rating and cost ratings. The K-Means clustering technique forms four clusters (excellent, good, okay, and bad) using the A2Cloud and cost score dataset. The S-NB trains with the output of the K-Means clustering.

We use eight real-world scientific applications and 20 Cloud instances for generating the training dataset. For the model verification, we use three real-world scientific applications and 20 Cloud instances.

We also execute three real-world scientific applications on 20 Cloud instances from Amazon Web Service (AWS), Microsoft Azure, Google Cloud Platform (GCP), and Linode and collect the application runtime data. The A2Cloud-RFC calculates the runtime instance rating and cost rating. We apply K-Means clustering to create four runtime rating clusters (excellent, good, okay, and bad) information. Finally, we compare the NB-Next predictions with runtime clusters to verify the NB-Next model's performance.

Using the collected runtime data, A2Cloud-ext engine generates the runtime score and cost score. We apply K-Means algorithm to create clusters (excellent, good, okay, and bad) information. We perform the comparison between the generated runtime clusters and S-NB predicted clusters for verification.

The rest of the thesis is organized as follows. Chapter 2 presents the related work and research conducted on Cloud resource selection using machine learning algorithms. Chapter 3 explains the A2Cloud-RFC framework and the machine learning algorithms. Chapter 4 discusses our proposed Cloud resource selection methodologies: Naive Bayes NEXT to Random Forest classifier and Standalone Naive Bayes classifier. Chapter 5 provides the machine learning methodologies in action and model performance evaluation. In Chapter 6, we conclude our research work and provide insights into future work.

# CHAPTER 2: RELATED WORK

In this chapter, we discuss previously published research on the Cloud resource selection problem. We identify their shortcomings and show how our research overcomes them.

Roloff et al. [12] perform a detailed analysis of high-performance (HPC) application execution on the Cloud instance. They consider the application performance and cost-efficiency of HPC applications on the Cloud. The application performance is measured by using micro-benchmarks. The study finds that the costly and powerful instances ensure the high performance and efficiency of the HPC applications on the Cloud. The study does not include the application's data input-output (I/O) performance.

Okada et al. [13] evaluate the NASA Advanced Supercomputing (NAS) parallel benchmarks performance on the Cloud instances. The study focuses only on the Google Cloud Platform instances. In contrast, we include other Cloud services such as Amazon EC2, Microsoft Azure, and Linode for a comprehensive analysis.

Kim et al. [14] provide an end-to-end resource management system for scientific applications on public Clouds. They propose a local linear regression model to predict the job execution time. The proposed model uses the type of virtual machines and data size required for the execution. The resource management system works on top of Amazon EC2 and utilizes the Amazon EC2's instances. The study shows better cost efficiency than baseline models. In our study, we use the cost model and multiple Cloud service providers instances for Cloud resource selection.

Gong et al. [15] propose a predictive elastic resource scaling for Cloud services. The predictive model utilizes signal processing and statistical learning algorithms for predicting

Cloud resources. The model uses the RUBiS benchmark and executes it on the Google Cloud platform. The study presents a high accuracy for predicting Cloud resources. Our proposed method utilizes the Linux Perf engine and Cloud benchmarks for profiling the applications and Cloud instances. Overdetailed application and Cloud instance profiling provides significant insights into applicating behavior on the Cloud instance.

Grag et al. [16] present a framework for ranking Cloud computing services using the Analytical Hierarchical Process (AHP). The framework is based on the user's Quality of Service (QoS) requirements. The framework measures the instance quality and prioritizes Cloud services. AHP uses the measured data to rank the Cloud services.

Iosup et al. [17] analyze the Cloud computing services for scientific computing applications. They perform an empirical evaluation of four Amazon EC2 instances using trace-based performance characteristics and cost models. The study indicates that scientific application's performance characteristics enhance the efficiency of Cloud selection.

Chard et al. [18] develop a model based on application profiling and dynamic market prediction to recommend an effective Cloud service for a given application. In a similar work, Chard et al. [19] develop an automated tool for application performance profiling on Cloud different Cloud instances. The automated tool enables the dynamic provisioning of Cloud instances, automated application deployment on Cloud, and generation of profiling data. The automated tool performs application profiling on Cloud instance, which is a costly approach. Our proposed work does not require to deploy the application on the Cloud instance.

Several research articles are machine learning to guide Cloud resource selection. Bankole et al. [20] develop a Cloud resource provisioning framework using support vector machine, neural network, and linear regression. The use CPU utilization, response time and throughput

metrics for model training dataset. They test their models with web applications and found that the support vector machine performed a better Cloud instance prediction.

Guo et al. [21] develop a Cloud recommendation model using K-Means and Analytic hierarchy model. The machine learning model employs the user defined values such as CPU and memory usage. They executed the applications on selected Cloud instances for collecting runtime. The training dataset consists of CPU usage, memory usage, and runtime. Our proposed model does not require executions on Cloud instances thereby saving money for the end-users.

Liu et al. [22] propose a Cloud instance type selection algorithm based on genetic algorithm (CITSA-GA). The genetic algorithm uses the 2D encoding between genes, roulette strategy, and crossover with mutation methods. They test their method against three generic algorithms: traversal algorithm, genetic algorithm, and particle swarm optimization algorithm. The accuracy of the CITSA-GA was obtained almost 82.5%. They only consider the Amazon EC2 compute intensive instances and do not consider the instance pricing. Our study considers memory-intensive and general purposes instance from multiple Cloud service providers. We also include the instance cost model for recommending the Cloud instances.

Samreen et al. [23] implement Daleel, a machine learning based Cloud instance selection framework. The framework uses the evidence-based knowledge of the Internet as a service setup. The framework takes the customer's requirements and constrains to recommend the Cloud instance. They perform an empirical study on three different Amazon EC2 Cloud instances. They execute one application 'VARD' to collect data for polynomial regression. Thy use linear and nonlinear models for the application runtime predication. This study shows that the non-linear model outperforms the linear model. One major shortcoming of this study are the selection of fewer Cloud instances and benchmarking with only one application.

Ouyang et al. [24] propose a machine learning-based node performance analyzer. They analyze the node performance using OpenCloud trace log parallel execution data and select a series of node performance features. The proposed analyzer uses the parallel tasks execution log data for training and predicts its performance for scheduling tasks. They consider the MapReduce application for analysis of the model data. The model shows an average accuracy of over 92.86%.

Kaplunovich et al. [25] develop a recommendation system for recommending an effective Cloud instance. The machine learning model uses big data sets on assorted AWS instances for training. The ultimate goal is to save time and cost for choosing a Cloud instance.

Wamba et al. [26] develop a workload prediction model using constraint programming and neural network for dynamic Cloud resource provisioning. They also build two workload generators for extending the experimental data. The models validate using the real Cloud traces. The study shows that the constraint programming is highly amendable for trace generation. On the other hand, the neural network gives better predictions.

Sun et al. [27] propose a consumer-centered Cloud selection using the Analytical hierarchy process (AHP). The study considers the consumer's qualitative and semi-qualitative personalized preferences such as response time, throughput, availability, reliability, and cost to make decisions using AHP. Thy test the proposed model using AWS EC2 Cloud instances. Unlike the above activities, we focus on the scientific application's performance parameters and cost model of the Cloud instance for recommending the Cloud resource.

Chen et al. [28] develop a fuzzy logic-based decision-making method for Cloud service evaluation. The study uses the fuzzy analytical hierarchy process method to calculate the fuzzy weights of each criterion from interval-valued fuzzy sets. The decision-maker has the choice to

use the linguistic variables for selecting the criteria importance, performance rating, and systematic solve the decision problem.

Ashwini et al. [29] build an efficient Cloud resource selection framework for high-performance computing applications. They form a cluster of heterogenous computes instances for high-performance computing applications. They use a K-Means model and employ CPU power, bandwidth, and execution time dataset. The K-Means model and brute force method show identical results for Cloud instance selection.

The literature presents research on the execution of the application on the Cloud instances for model training. The execution of application on Cloud instances is an expensive approach. Rathnayake et al. [30] present an analytical modeling approach 'CELIA' to determine cost-time-optimal Cloud resources of elastic applications. The model uses the execution time and cost models from baseline for estimating application resource demand and Cloud resource capacity for Amazon EC2 instances. Their study does not characterize the applications.

Morais et al. [31] propose a proactive horizontal auto-scaling for instance selection. They use CPU and memory utilization, cost, Quality of service (QoS) for the application for developing a prediction model. In addition, they consider only the Amazon EC2 instances. In contrast, we include more instances from more Cloud service providers. Grandhi et al. [32] develop a Cloud performance evaluation model using a fuzzy algorithm. The performance of Cloud computing depends on a multi-attribute group. The proposed study considers the performance evaluation problem as a multi-attribute group decision making problem and implements the fuzzy multi-attribute group decision-making model for solving the problem. The research determines the effectiveness of the proposed fuzzy model. In our study, we use the Naive Bayes model for making predictions that requires small dataset and computation powers.

Sohaib et al. [33] propose a fuzzy model for e-commerce Cloud computing. The study includes the technological, organizational, and environmental factors associated with e-commerce applications hosted on Cloud services. The fuzzy model recommends the ideal solution for e-commerce site using the order of preference by similarity.

# CHAPTER 3: PRELIMINARIES

This chapter describes the theoretical aspects of the A2Cloud-RFC framework and explains three machine learning algorithms employed for instance recommendation. These include random forest classifier, K-means clustering, and the Naive Bayes classifier. Section 3.1 presents the A2Cloud-RFC framework and Section 3.2 describes the three machine learning algorithms.

## 3.1 A2Cloud-RFC Framework

The A2Cloud-RFC framework [11], [34] is an easy-to-use analytical framework that recommends effective Cloud instances for executing scientific applications on Cloud platforms. Figure 3.1 shows the A2Cloud-RFC framework. The framework inputs the scientific application and the selected Cloud instances and leverages the performance benchmarks and random forest classifier to generate the Cloud instances ratings. These ratings enable users to select the most effective Cloud instance for their application.

The A2Cloud-RFC framework comprises the A2Cloud framework and the random forest classifier as shown in Figure 3.1. The A2Cloud framework generates the A2Cloud score via application and Cloud instance benchmarking. The random forest classifier uses the A2Cloud score to form the random forest using multiple decision trees. Using the random forest, the A2Cloud-RFC framework assigns a final rating to the selected Cloud instances for the tested application.

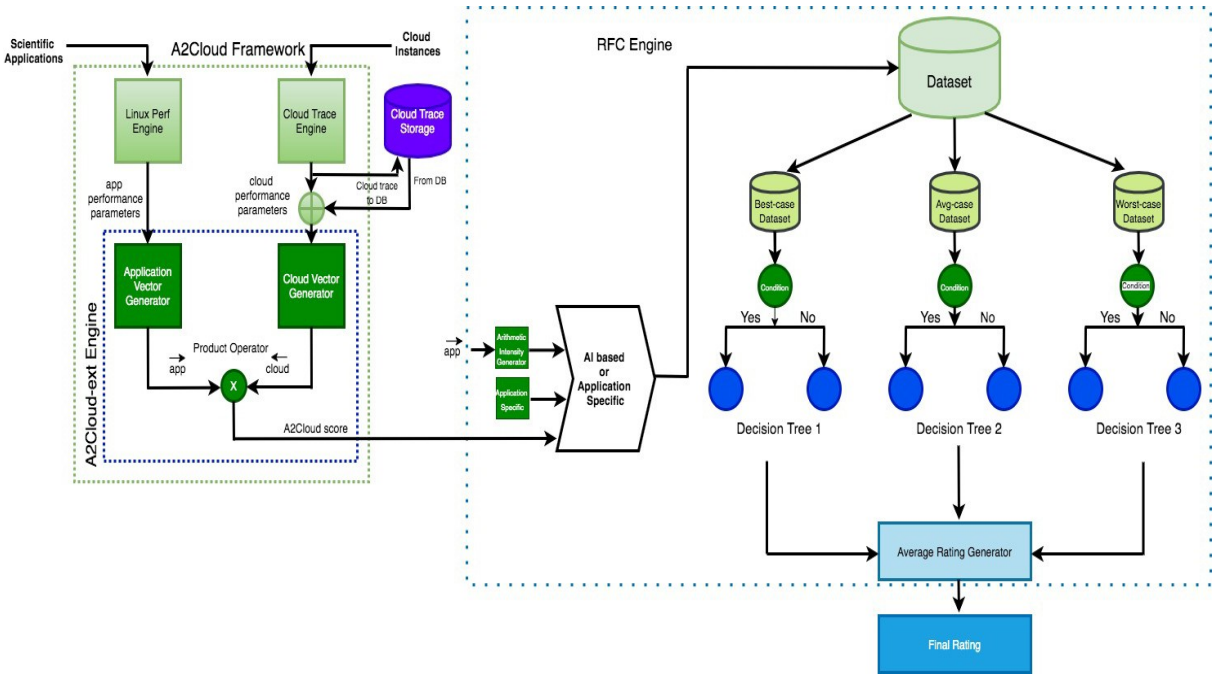



*Figure 3.1* A2Cloud-RFC framework: A2Cloud framework and random forest classifier

The A2Cloud Framework comprises the Linux Perf engine, Cloud trace engine, and A2Cloud-ext engine. The Linux Perf engine generates the application performance parameters that characterize an application. The Cloud trace engine determines the Cloud performance parameters for the Cloud instances and these parameters are complementary to application performance parameters. The A2Cloud-ext engine leverages the application vector and Cloud vector generator engines. The application vector generator creates the application vector from the application performance parameters. The Cloud vector generator constructs the Cloud vector using the Cloud performance parameters. The Matrix-vector product operator multiplies the application vector and Cloud vector to produce the A2Cloud score vector.

Section 3.1.1, 3.1.2, and 3.1.3 describe the three engines of the A2Cloud Framework: Perf Engine, Cloud Trace, and A2Cloud-ext engines, respectively.

### 3.1.1 Perf Engine

The A2Cloud framework executes the Linux Perf engine for application performance measurement (see Figure 3.1). The Linux Perf engine's statistical sampling counters are programmed to take periodical measurements of application parameters including the number of single-precision floating-point operations per second (SPFLOPs), number of double-precision floating-point operations per second (DPFLOPs), the total number of x87 instructions (x87), number of main memory reads and writes (mem), and number of disks reads ($disk_{read}$), and writes ($disk_{write}$). Table 3.1 provides the Perf engine counters and their descriptions.

Table 3.1
*A List of Perf Computation and Memory Counters*

| Counter Type | Mnemonic | A2Cloud Name | Description |
|---|---|---|---|
| Computation | fp comp ops exe.x87 x87 instructions | x87 | x87 instructions |
| | fp comp ops exe.sse packed single | SP scalar | Scalar single-precision |
| | simd fp 256.packed single | SP packed | packed SSE single-precision |
| | fp comp ops exe.sse scalar double | SP SIMD | SIMD single-precision |
| | fp comp ops exe.sse packed double | DP scalar | scalar double precision |
| | simd fp 256.packed double | DP packed | packed SSE |
| | | DP packed | SIMD double-precision |
| Main Memory | uncore imc {0-7} cas count read | uncore read {0-7} | memory read operations performed (un- core read ops) |
| | uncore imc {0-7} cas count write | uncore write {0-7 | memory write opera- tions performed (un- core write ops) |

The SFLOPs component comprises one scalar single-precision operation, four packed SSE single-precision operations, and eight SIMD single-precision operation. Equation 3.1 shows the calculation of SPFLOPs using its constituents.

$$SPFLOPs = SP_{scaler} + 4\,SP_{paced} + 8\,SP_{SIMD} \quad (3.1)$$

Similarly, the DPFLOPs combine scalar double-precision, two packed SSE double-precision and four SIMD double-precision functions (see Equation 3.2).

$$DPFLOPs = DP_{scaler} + 2\,DP_{paced} + D\,DP_{SIMD} \quad (3.2)$$

The x87 counter calculates the x87 instructions. The main memory accesses are calculated using the Perf engine uncore read and write functions. Equation 3.3 shows the memory access calculation:

$$mem = \sum_{n=0}^{7} uncore\_reads\_i + \sum_{n=0}^{7} uncore\_writes\_i \quad (3.3)$$

The disk read and write are the user-defined parameters. The PERF engine writes the performance parameters into an application trace as a JSON file. A detailed information about the counters can be found in [34], [35].

### 3.1.2 Cloud Trace Engine

The Cloud trace engine performs 1000 statistical executions of performance benchmarks on the selected Cloud instances to assess the Cloud instance's stochastic behavior. These benchmarks include LINPACK [36] and Stream [37] to calculate the floating-point precision, memory, and disk performances of the selected Cloud instances.

The LINPACK suite evaluates the single-precision floating-point per second (SPFLOPS), double-precision floating-point operations per second (DPFLOPS), and the x87 instructions per second (x87S) of a system. The STREAM benchmark suite determines the main memory

bandwidth ($mem_{\beta}$). The other Cloud performance parameters such as $disk_{read\beta}$, and $disk_{write\beta}$ are determined by using the dd micro-benchmark.

After performing the benchmarks on the Cloud instances, the engine writes the performance parameters to a Cloud trace and stores in a database as a JSON file.

### 3.1.3 A2Cloud-ext Engine

As seen in Figure 3.1, the A2Cloud-ext engine generates the final A2Cloud score using its three components: application vector generator, Cloud vector generator, and matrix-vector product operator. We describe the functionality of the A2Cloud-ext engine components.

- Application Vector Generator

The application vector generator inputs the application trace to the application vector. The application vector generator $\overrightarrow{app}$ using Equation 3.4.

$$\overrightarrow{app} = [SFLOPS\ DPFLOPS\ mem\ disk_{read}\ disk_{write}\ x87]^T \quad (3.4)$$

- Cloud matrix generator

The Cloud matrix generator creates a Cloud matrix whose columns are constituted by the Cloud vectors. A Cloud vector contains the Cloud instance performance parameters including the SFLOPS, DPFLOPS, x87S, $mem_{\beta}$, $disk_{read\beta}$, and $disk_{write\beta}$.

To construct a statistical vector for each Cloud instance, the Cloud-matrix generator fetches the JSON file from the database. The generator applies the central limit theorem [38] to the Cloud performance parameters to fit normal distribution curves. Using the normal distributions, the generator calculates the mean ($\mu$) and standard deviation ($\sigma$) of Cloud performance parameters. Equation 3.5 presents the statistical Cloud vector generated via the above process.

$$\overrightarrow{cloud} = [\frac{1}{N(\mu,\sigma^2)SPFLOPs} \ \frac{1}{N(\mu,\sigma^2)DPFLOPs} \ \frac{1}{N(\mu,\sigma^2)mem} \ \frac{1}{N(\mu,\sigma^2)disk_{read}} \ \frac{1}{N(\mu,\sigma^2)disk_{write}} \ \frac{1}{N(\mu,\sigma^2)x87}]^T \quad (3.5)$$

where $\mu$ and $\sigma$are the mean and standard deviation of the parameters.

The Cloud matrix generator arranges the Cloud vectors in the column major format to create the Cloud matrix, $cloud$. This matrix is input by the matrix-vector product operator.

- Matrix-vector product operator

This module performs a matrix-vector product (Equation 3.6) of the $\overrightarrow{app}$ and the $cloud$ to generate A2Cloud score vectors. The engine normalizes the scores on scale of 1 to 10 because the normalized score improves the performance and training stability of the model.

$$\overrightarrow{Score}_{A2Cloud} = [\overrightarrow{cloud}_1 \ \overrightarrow{cloud}_2 \quad \overrightarrow{cloud}_m]^T X \begin{bmatrix} SPFLOPs \\ DPFLOPs \\ mem \\ disk_{read} \\ disk_{write} \\ x87 \end{bmatrix} \quad (3.6)$$

Each scalar in the A2Cloud score vector represents to the level of match between the application and the corresponding Cloud instance.

### 3.2 Machine Learning Algorithms

Machine learning (ML) is a data-driven method of building an analytical model for predictive analysis or recommendation. The ML model learns from the data and makes a prediction based on the learned parameters. The algorithms are broadly three categorized into unsupervised, supervised, and reinforcement techniques [39]. The unsupervised and supervised learning require small dataset, create a less complex model, and easy to deploy whereas reinforcement learning uses large dataset, complex model, and high computing power to train the model. Our work has small dataset and therefore, we use on supervised machine learning algorithms.

### 3.2.1 K-Means Clustering

Clustering is a machine learning algorithm that searches the hidden patterns in the raw data to create clusters with similar characteristics. K-means is one of the popular algorithms that uses numerical and unsupervised method to create clusters. In our work, we use K-Means to create labeled data.

K-Means divides the data based on the Euclidean distance from the cluster origin. The algorithm steps are as follows [40], [41]:

- Identify the number of clusters (K) and randomly assign the cluster center coordinates
- Calculate Euclidean distance of each data point from the cluster centroid
- Move to the cluster centroid to the mean of its Euclidean distance of assigned datapoints
- Repeat step 2 and 3 until the centroid does not change

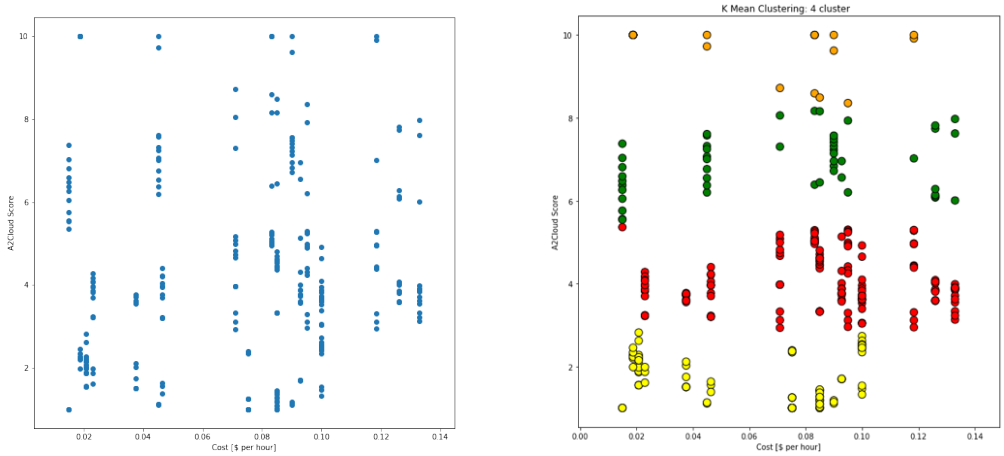
Equation 3.8 shows the formula for calculating Euclidean distance between two points is

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \qquad (3.7)$$

where d means the distance between two points, centroid and points coordinates are $(x_1, x_2)$ and $(y_1, y_2)$ respectively.

### 3.2.2 Random Forest Classifier

Random Forest classifier (RFC) [42], [43] is a supervised machine learning algorithm that uses multiple decision trees constructed from a dataset. The entropy and information gain are the basis of decision trees construction. We use Iterative Dichotomiser 3 (ID3) [44] algorithm to calculate the entropy and information gain.




*(a)* Unstructured data (*b)* Dataset with four clusters

*Figure 3.2* Example of K-Means algorithm with four clusters

A conceptual diagram of the random forest is depicted in the Figure 3.3. The decision trees are constructed using a top-down approach. The required ID3 metrics are entropy and information gain. The algorithm parameters (entropy and information gain) and ID3 algorithms are described as follows.

- Algorithm Parameters

Entropy represents the amount of uncertainty in the dataset. It is also a way of measuring impurity of the data. Based on the impurity, decision tree nodes are separated. Equation 3.8 denotes the entropy:

$$H(S) = \sum p(x) log_2 P(x) \qquad (3.8)$$

where $H(S)$ is the entropy of dataset, $S$ represents the current dataset, $P(x)$ is the proportion of the number of elements in a category.

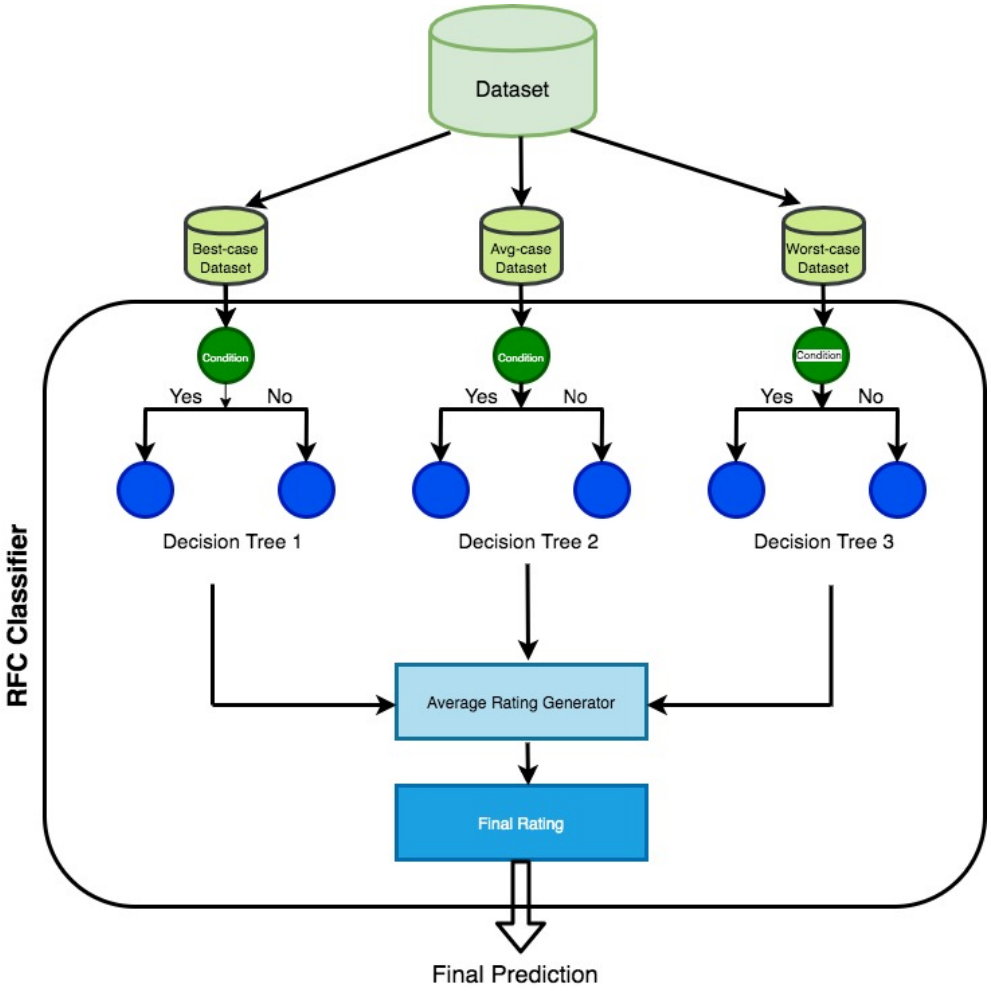



*Figure 3.3* The random forest classifier (RFC) function block diagram

The entropy value becomes zero when all samples of a node belong to the same category. In contrast, the entropy has the maximum value for the uniform class distribution. Also, it may reach the maximum value because of all classes in the node having equal probability. So, the entropy maximizes mutual information by creating an equal probability node in the decision tree [45], [46].

In the decision tree technique, we create the root node first and then pass the feature data on the leaf node. It results in the largest information gain (IG). Also, IG calculates the reduction

in entropy in the training dataset and can be used for feature selection by determining the gain of each variable in the target variable. Equation 3.9 shows the mathematical representation of IG calculation.

$$IG(S|A) = H(S) - \sum p(t)H(t) = H(S) - H(S|A) \qquad (3.9)$$

where $S$ represents the current dataset, $H(S)$ is entropy of set S, $\sum p(t)H(t)$ is total entropy of all subsets of $S$.

- ID3 algorithm

The ID3 algorithm calculates the entropy and information gain of each data attributes from the dataset. The attribute with maximum information gain is the root node of decision tree. The values contained in this specific attribute become the node's branches. The algorithm continuously splits the attributes of subsets and stops when no more splitting is possible for any attribute [47]. Those attributes information gain values become terminal nodes. ID3 algorithm generates multiple decision trees to perform the random forest classification. The final classification combines the terminal nodes of all the decision trees. The terminal nodes denote a different classification and have its own weight value. The average numerical weight of terminal nodes is the final rating for the particular item.

The RFC engine is cascaded to the A2Cloud Framework as shown in Figure 3.1. The RFC engine suggests the Cloud instances based on two different methodologies: arithmetic intensity-based (AIRF) and application-specific random forest generator (ARF).

- Application-specific Random Forest (ARF) generator

The application-specific random forest (ARF) generator uses the A2Cloud scores to make decision trees (see figure 3.1). The ARF constructs three decision trees for the best-case, avg-

case, and the worst-case instance performance. Each decision tree uses the data splitting rules enlisted by Samuel et. al. [11].

- Arithmetic-Intensity Random Forest (AIRF) Generator

The arithmetic intensity generator (AIG) is responsible for generating the arithmetic intensity (AI) value of applications. The AIG engine takes the application vector as input and calculates the arithmetic intensity value. The AIG obtains the AI value using Equation 3.10. The numerator term denotes the sum of the computation components of the application vector and the denominator term represents the memory access component of the application vector.

$$AI = \ln\left(\frac{\sum computions}{\sum memory\ access}\right) \quad (3.10)$$

After determining the AI value, the AIRF pulls the performance traces from the database. Then, it constructs the trees using the same methodology as the ARF generator to construct the decision trees. The constructed decision trees combine together to form the random forest. Each tree node is assigned with numerical ratings to generate the final Cloud instance rating.

### 3.2.3 Naive Bayes Classifier

The Naive Bayes classifier is a supervised machine learning algorithm that falls into probabilistic classifiers family. The algorithm is based on Bayes' theorem of probability. The Naive word means that the features are independent of each other.

Bayes' theorem [48] determines the conditional probability of an event based on the prior associated conditions of that event. Bayes theorem is given in Equation 3.11.

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)} \quad (3.11)$$

where p(H|D) is posterior probability or probability of hypothesis H given data D, p(D|H) is the probability of given data D when hypothesis H is true, p(H) is the hypothesis probability or prior probability, p(D) is the probability of data

#### 3.2.3.1 Probabilistic Framework of Naive Bayes Classifier

The NB framework is shown in the Figure 3.4. The classifier model maps input feature vectors $x \in X$ to output class labels $y \in 1, 2, .., C_k$ where $x = [x_1, x_2, \ldots., x_n]$ feature vector, number of classes k, and classes $C_k$.

The classifier model learns from a labeled training set of input pairs as a part of supervised learning method. The Naive Bayes probabilistic classifier [49]-[52] including Bayes theorem is shown in Equation 3.12.

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} \qquad (3.12)$$

where $p(C_k|x), p(x|C_k), p(C_k)$, and $p(x)$ are posterior, likelihood, prior, and evidence respectively.

The NB model training has the input feature vectors (i.e. A2Cloud score, cost score). The features vector has a numeric value between 1 to 10 and those are continuous. The NB model assumption is that continuous input feature vectors associated with each class are in normally distributed. For our case, we divide the data by class and calculate the mean and variance of the input feature vector in each class.

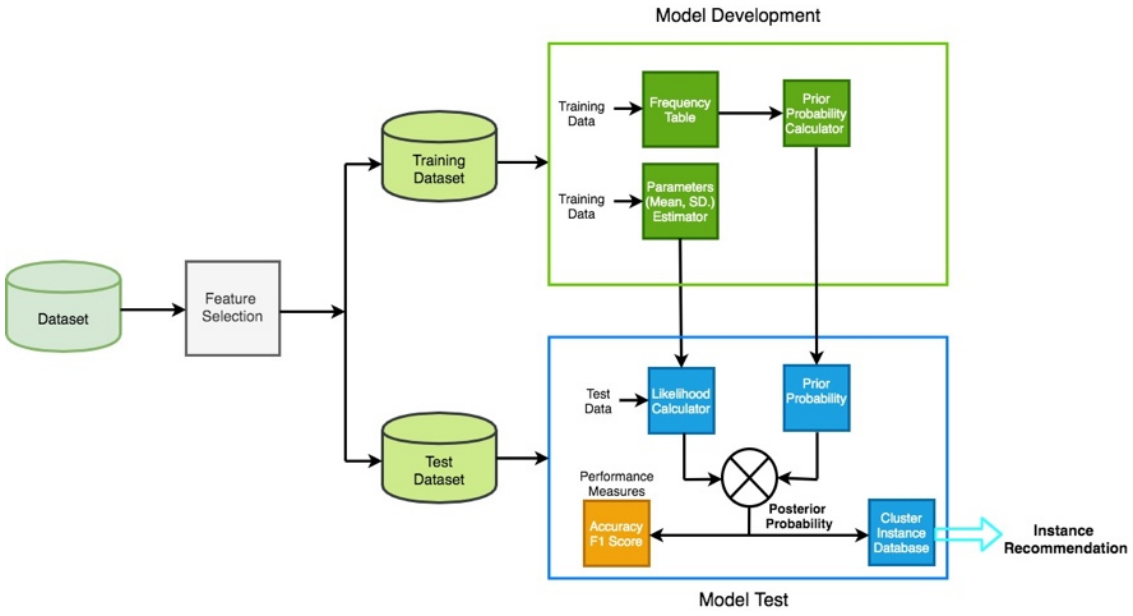



*Figure 3.4* The NB classifier working methodology

Assume mean ($\mu_k$) and standard deviation ($\sigma_k$) of the input feature vector is associated with class, $C_k$. Then, the mathematical expression of the multi-class Gaussian Naive Bayes Classifier is as follows:

$$p(x|C_k) = \frac{1}{\sqrt{2\pi\sigma_k{}^2}} exp \frac{(x-\mu_k)^2}{2\sigma_k{}^2} \qquad (3.13)$$

where feature vector $x = [x_1, x_2, \ldots., x_n]$ in $n$ dimensional space, $C_k$. is class variable.

In summary, the basic steps of Gaussian Naive Bayes Classification algorithm are described as follows [53]:

The NB classifier converts the training dataset into the frequency table and prior probability table of four classes. Based on those tables, model prepares the events probability and likelihood tables. The NB equation determines the posterior probability of each class for the new data item. The higher posterior probability of new instances determines its class.

### 3.2.4 Machine Learning Model Evaluation Metrics

We consider the confusion matrix, accuracy, and F1 score for the machine learning model evaluation.

- Confusion matrix

The confusion matrix is an NxN matrix where N represents the number of clusters. The matrix contains information about the actual cluster and model predicted cluster information. A table of confusion or the confusion matrix reports the number of false positives, false negatives, true positives, and true negatives. Those parameters express the proportion of correct classifications. True positive (TP) represents that the NB correctly predicted positive clusters are actually positive clusters. If the NB classifier predicts the clusters as positive but they are actually negative; this represents the false positive (FP). True negative (TN) expresses the accurate prediction of the negative class. False-negative (FN) is an outcome where the model incorrectly predicts the negative class.

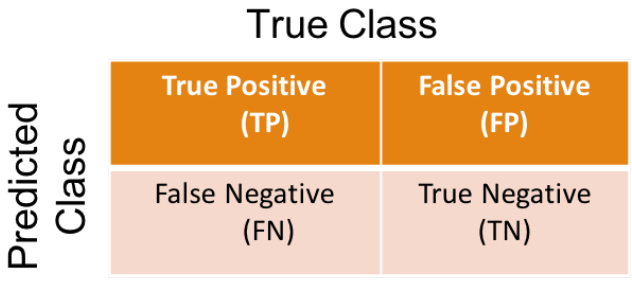



*Figure 3.5* Confusion matrix

- Accuracy

Accuracy explains the correctness of the model, showing the number of correct predictions out of the total predictions.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.14)$$

- F1 Score

F1 score is a combination of recall and precision. The maximum value of the F1 score is 1. The high F1 score represents the model performing outstanding in case of recall and precision.

$$Precision = \frac{TP}{TP+FP} \quad (3.15)$$

$$Recall = \frac{TP}{TP+FN} \quad (3.16)$$

$$F1 = \frac{2\ X\ Precsion\ X\ Recall}{Precsion + Recall} \quad (3.17)$$

## 3.3 Summary

The A2Cloud-RFC framework includes the PERF engine, Cloud trace engine, and A2Cloud-ext engine. The PERF engine calculates the application performance parameters. Cloud trace engine generates the Cloud instance performance parameters. The A2Cloud-ext engine converts the application and Cloud performance parameters to application vector and Cloud matrix. The matrix-vector product generator multiples the application vector and Cloud matrix to from the A2Cloud score.

Machine learning algorithms (K-Means, Random Forest Classifier, Naive Bayes) are used to build the Cloud instance recommender. K-Means generates the labeled data using the

original dataset. Random Forest Classifier generates the instance RFC rating. The Naive Bayes makes the final Cloud instance recommendation. The confusion matrix, accuracy, and F1-score evaluate a machine learning model's performance.

# CHAPTER 4: METHODOLOGY

This chapter outlines the synergy between the A2Cloud-RFC framework and machine learning agents for recommending the Cloud instances. We propose two different implementation methodologies: Naive Bayes NEXT to Random Forest Classifier (NB-Next) and a Stand-alone Naive Bayes classifier (S-NB). In addition, we provide an overview of the dataset generation and feature selection techniques for the NB-Next and S-NB classifiers.

## 4.1 NB-Next

Sections 4.1.1 and 4.1.2 describe the NB-Next machine approach and feature selection methodology, respectively.

### 4.1.1 NB-Next Machine Learning Approach

Figure 4.1 exhibits the workflow of the NB-Next classifier. The model pipeline comprises the A2Cloud-RFC framework with three NB classifiers: compute-intensive (CI), balanced, and memory-intensive (MI). Each one of the NB classifiers trains with a specific application class dataset.

The A2Cloud-ext framework takes the scientific application and target Cloud instance as an input. The A2Cloud-ext uses its internal counters and engines to generate the $\overrightarrow{app}$ and A2Cloud score (see Section 3.1).

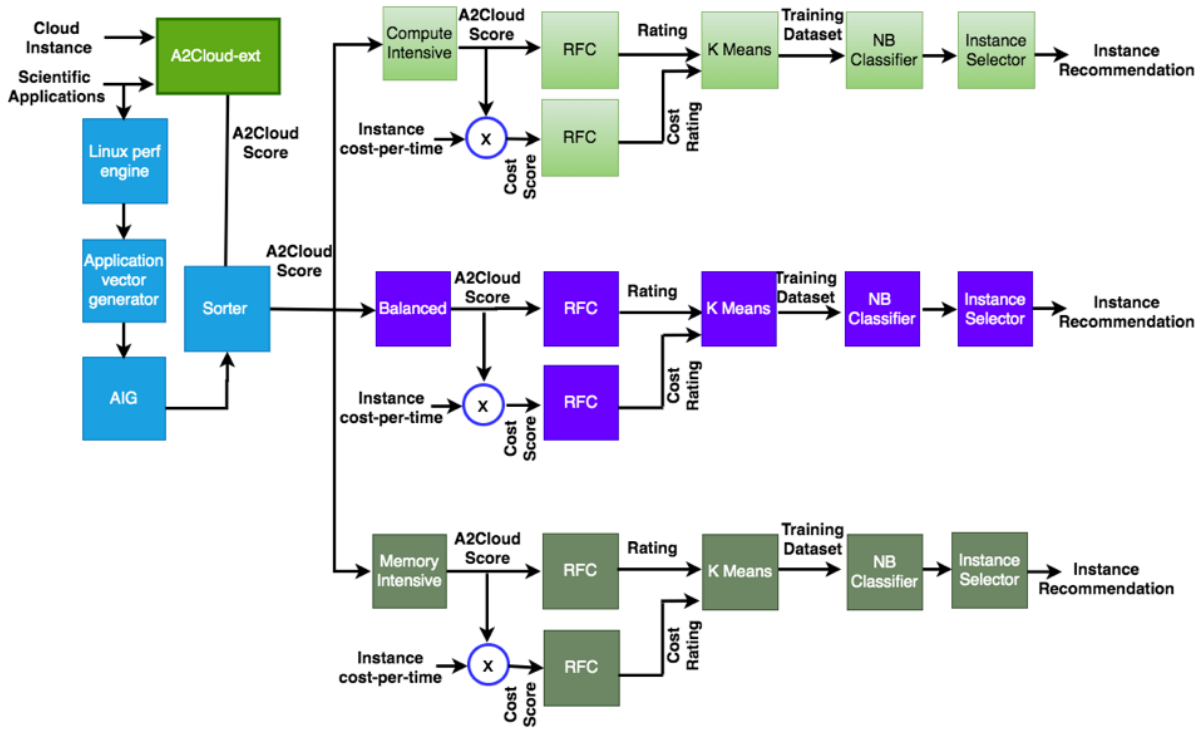



*Figure 4.1* NB-Next working methodology

Arithmetic-intensity generator (AIG) determines the arithmetic intensity (AI) of the scientific application (see Section 3.3.2). The AI value is the natural logarithm of the number of computations divided by memory access. If the arithmetic intensity (AI) value is greater than zero, then the AIG classifies the application as compute-intensive (CI) class because the application has more computations than memory accesses. A negative value of AI denotes that an application is memory-intensive (MI) (more memory accesses than computations). The balanced class has an AI value that is close to zero (approximately equal number of computations and memory accesses).

As seen in Figure 4.1, the NB-Next classifier workflow has three branches: compute-intensive, balance, and memory-intensive. The working principle of the three branches are

identical. Therefore, we describe the compute-intensive branch by using each component (RFC (A), K-Means (B), NB (C), and Instance selector (D)) in the pipeline.

#### 4.1.1.1 Random Forest Classifier (RFC)

The Random forest classifier (RFC) [11] takes the compute-intensive dataset as an input. The RFC generates three decision trees (based on best-case, average-case, and worst-case) instance performance based on the A2Cloud scores. RFC combines the three decision trees together to make a random forest where it assigns a number from 1 to 4 to each individual leaf node of the decision trees. The assigned number represents four cases: excellent (4), good (3), okay (2), and bad (1). Finally, RFC calculates the average of the leaf nodes for a given individual instance to provide an average rating.

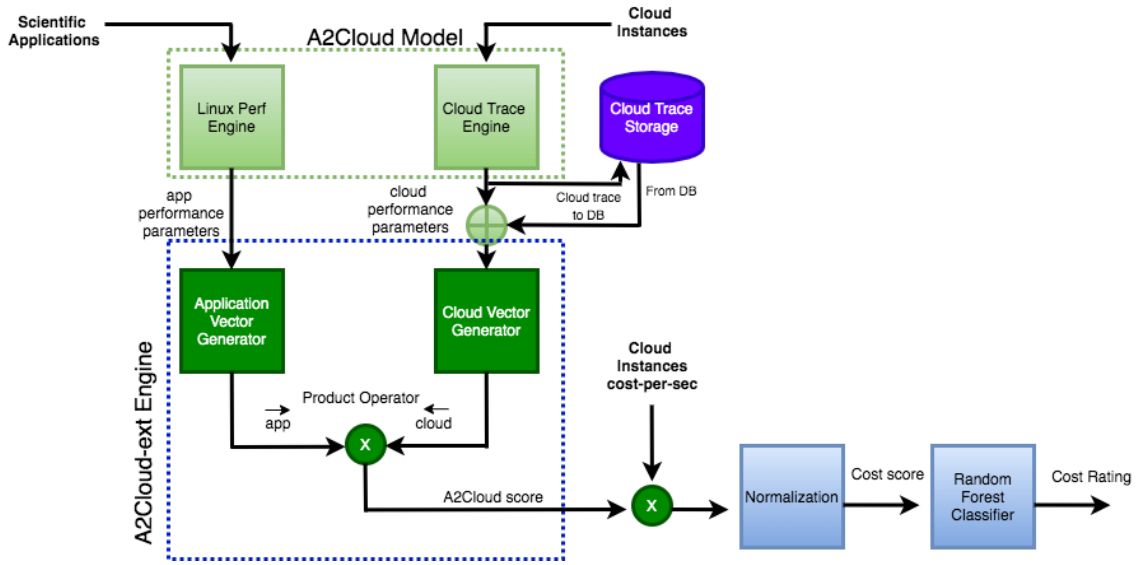
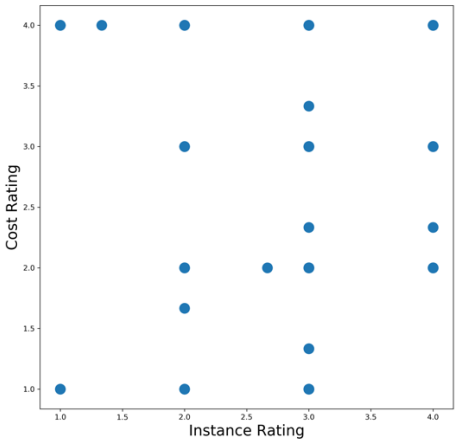
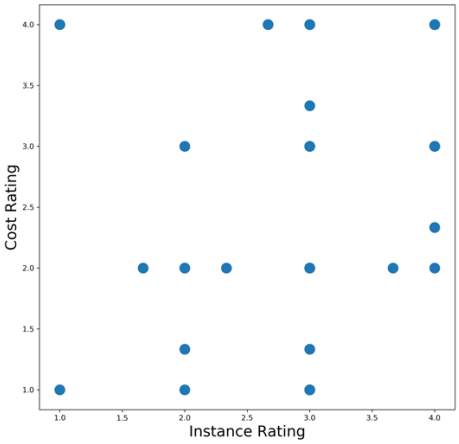


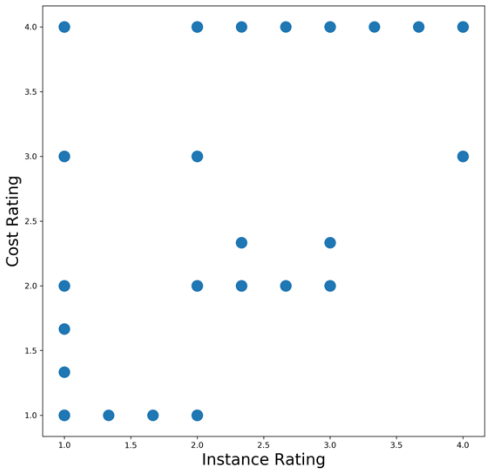

*Figure 4.2* Cost Rating generator working principle




*(a)* Compute-intensive dataset




*(b)* Balanced dataset




*(c)* Memory-intensive dataset

*Figure 4.3* Cloud instance rating and cost rating using A2Cloud-RFC framework

In addition to the A2Cloud scores, we also consider the Cloud instance pricing to construct the cost rating score as shown in Figure 4.2. The cost-per-second and A2Cloud scores are multiplied to form the cost score, which is then fed to the RFC to compute the cost random forest using the same approach as the A2Cloud random forest.

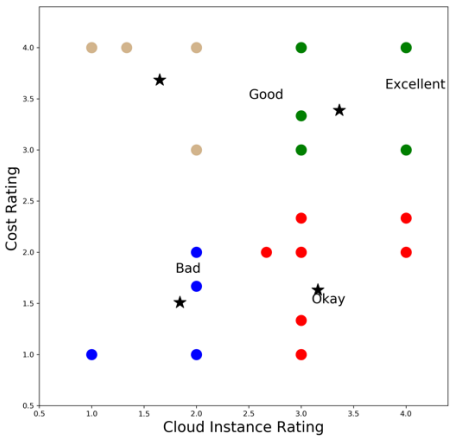
Figure 4.3 shows the NB-Next classifier's training dataset for three application classes: CI, balanced, and MI. Figure 4.3 a shows compute-intensive application class dataset. Figures 4.3 b and 4.3 c exhibit the balanced and memory-intensive datasets. We apply the K-Means clustering on the dataset to label the data.

#### 4.1.1.2 K-Means Clustering

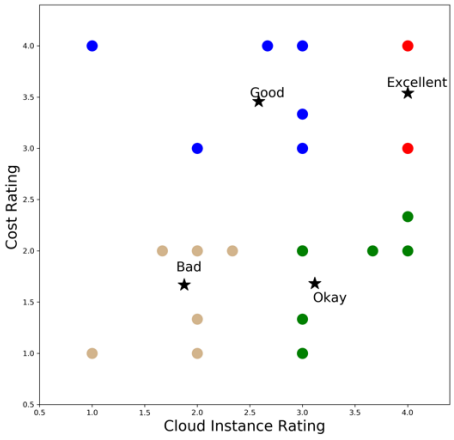
K-Means algorithm transforms the unlabeled dataset into labeled dataset. K-Means creates the four clusters: excellent (4), good (3), okay (2), and bad (1) from the training dataset. Figure 4.4 displays the K-Means clustered data with four clusters highlighted in different colors. The top right cluster in red represents the excellent case, the bottom right cluster in green denotes the okay case, the top left in blue represents the good case, and the bottom left in orange represents the bad case. The instance and cost ratings construct the input features for training.

#### 4.1.1.3 NB Classifier

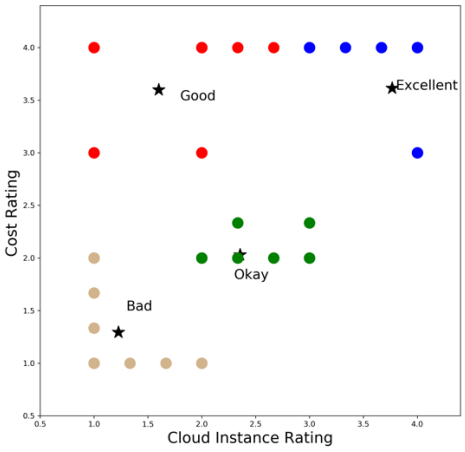
NB classifier has two phases: training and testing. In the training phase, the NB classifier uses the result of K-Means clustering to train the model. Figure 4.5 shows the working principle of the NB classifier. The NB model converts the training dataset into a frequency table for four classes: excellent (E), good (G), okay (O), and bad (B). Based on the frequency table, the NB model calculates the prior probability of four classes. In addition, the NB learns the respective mean ($\mu$) and standard deviation ($\sigma$) of input features (RFC rating and cost rating) for the four classes.




*(a)* Compute-intensive labeled dataset




*(b)* Balanced labeled dataset




*(c)* Memory-intensive labeled dataset

*Figure 4.4* Cloud instance rating and cost rating labeled dataset

The testing dataset has instance rating and cost rating for model testing. During the testing phase, the NB determines the likelihood probability using the Gaussian NB equation 3.13.

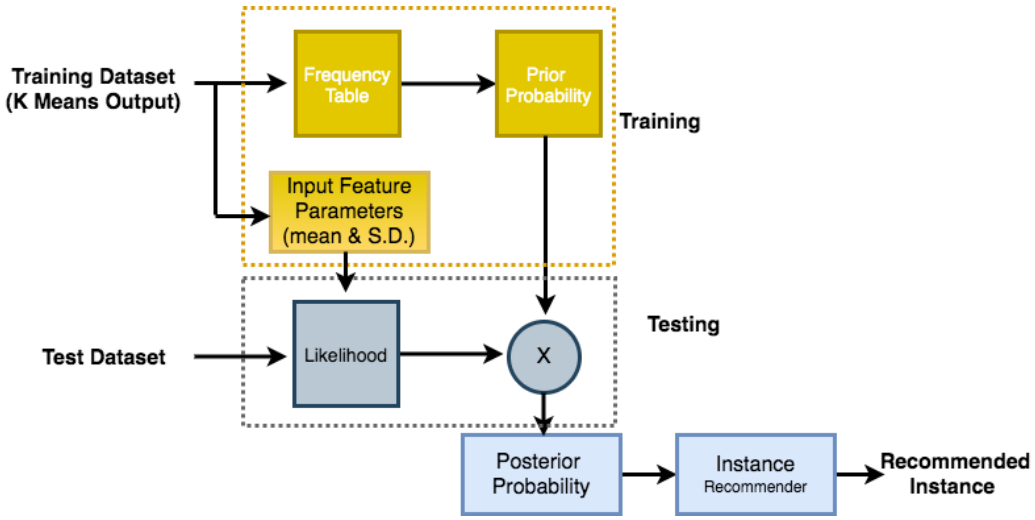



*Figure 4.5* NB training and testing phase methodology

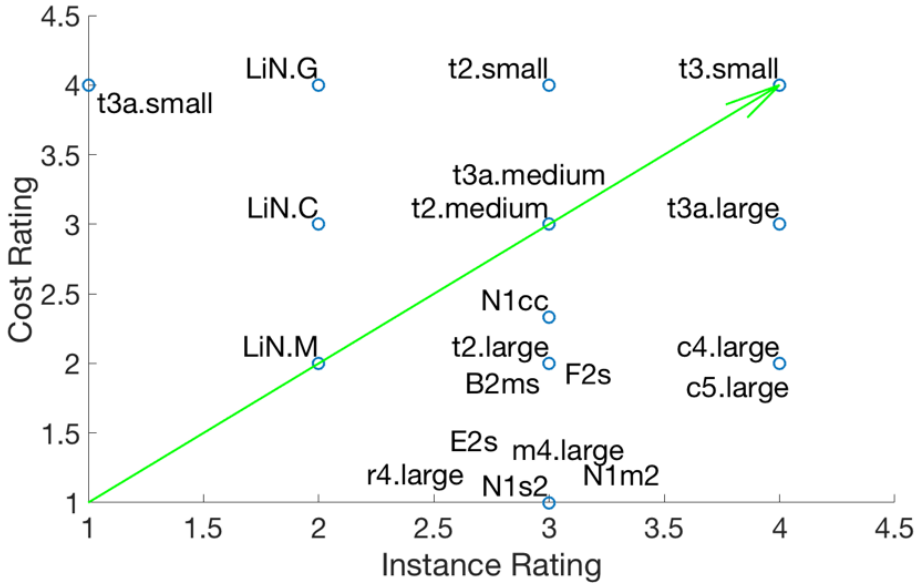



*Figure 4.6* Instance selector recommends instance with the highest Euclidean distance

Finally, the NB classifier uses the Naive Bayes equation (3.12) to determine the posterior probability of the testing dataset. The highest posterior probability of a class represents the outcome of the prediction.

For the testing phase, NB classifies the Cloud instances for the target scientific application and Cloud instances as excellent (E), good (G), Okay (O), or bad (B). The determined classification information is sent to the instance selector, which follows next.

#### 4.1.1.4 Instance Selector

The instance selector selects as optimal instance from the excellent (E) class. The instance selector uses the Euclidean distance method to recommend the final instances. The instance selector determines the Euclidean distance of all instances (RFC rating and Cost rating) in excellent class from the tuple (1,1). We choose the tuple (1,1) as origin because the minimum RFC rating and cost rating is (1,1). The instance selector recommends an instance with the highest Euclidean distance because the ideal tuple for the excellent class is (4,4). NB-Next uses RFC rating and cost rating that vary from 1 (least) to 4 (excellent). Figure 4.6 represents an example of the Cloud instance selector. The instance selector recommends the t3.small instance because it hast the highest Euclidean distance value 4.24 from the tuple (1,1).

The balanced and memory-intensive application classes follow the same methodology as discussed above.

## 4.2 Stand-alone Naive Bayes (S-NB) Methodology

The Stand-alone Naive Bayes (S-NB) classifier recommends the Cloud instances using the NB classifier alone. The model is referred as Stand-alone because it does not employ the Random Forest Classifier [34]. Section 4.2.1 describes the methodology of S-NB.

### 4.2.1 Stand-alone Machine Learning Approach

Figure 4.7 shows the working methodology of the S-NB classifier that uses the A2Cloud score directly from the A2Cloud framework.

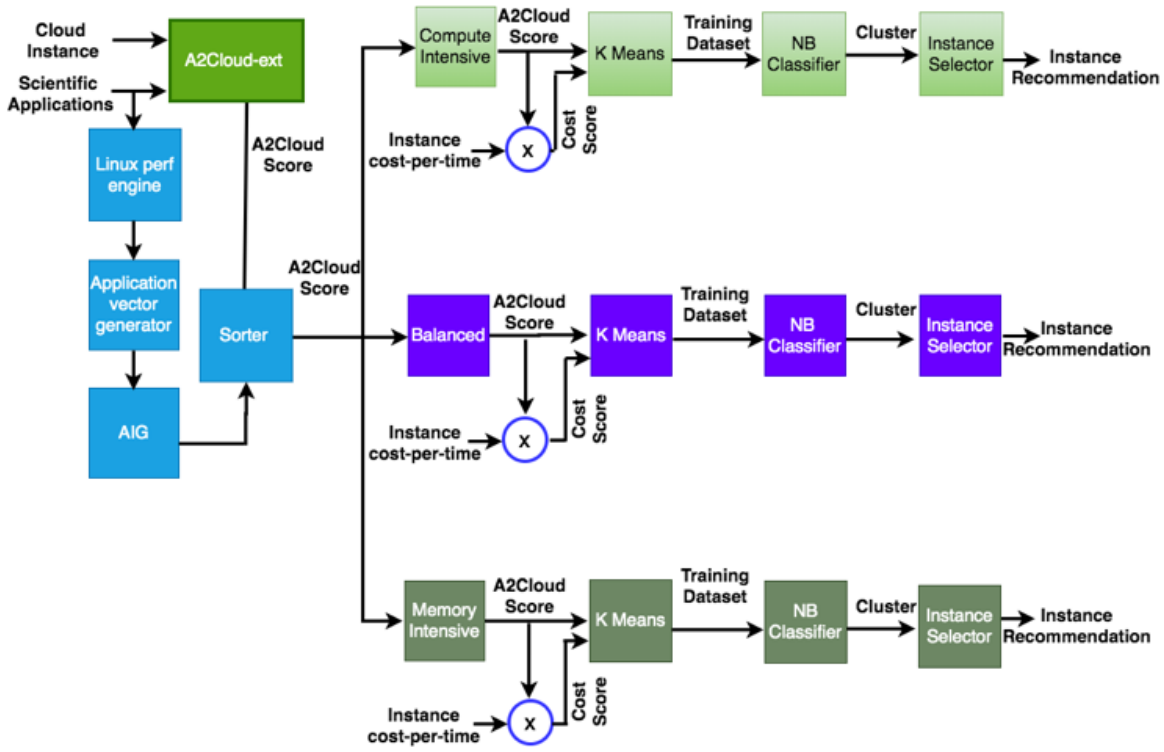



*Figure 4.7* S-NB classifier working methodology

As shown in Figure 4.7, the A2Cloud-ext framework generates the A2Cloud scores using the scientific applications and target Cloud instance (see Section 3.1). The arithmetic intensity generator (AIG) calculates the arithmetic intensity of the scientific application. The scientific application can be compute-intensive, balanced or memory-intensive based on the arithmetic intensity value. The AIG categorizes the A2Cloud scores into four application classes. Therefore, there are three A2Cloud scores datasets available for training one NB classifier (CI, MI, and balanced). In addition to the A2Cloud score, we add the cost metric of Cloud instances.

Figure 4.8 shows the cost score generation principle. The instance cost-per-second and A2Cloud score multiplies to produce the cost score where the lower value indicates a better fit.

The working methodology of S-NB has three branches: compute-intensive, memory-intensive, and balanced. All of the three branches follow the same working principle. Therefore, we explain the compute-intensive branch.

The input features (A2Cloud score, cost score) are used to train the S-NB classifier. Figure 4.9 displays the S-NB method's training dataset for three application classes: CI (4.9 a), balanced (4.9 b), and MI (4.9 c). We apply K-Means on the dataset to generate the labeled data.

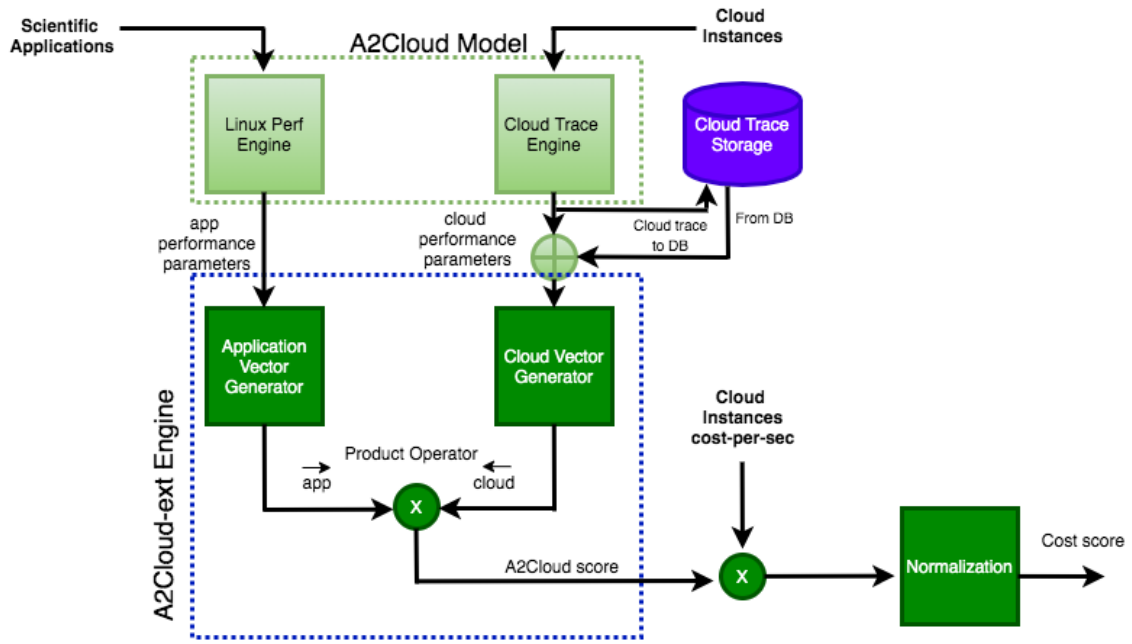


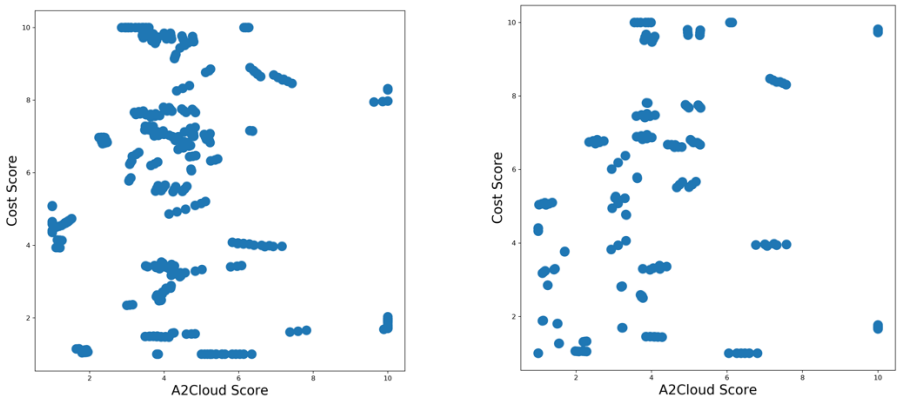

*Figure 4.8* Cost score generator working principle

#### 4.2.1.1 K-Means Clustering

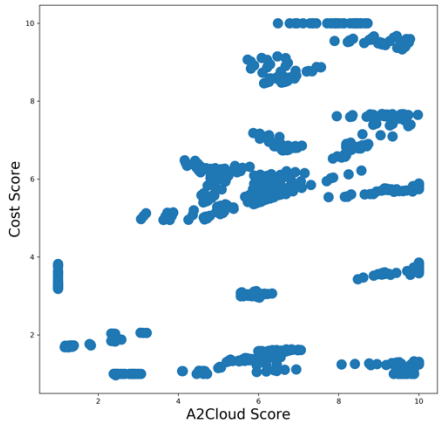
We apply the K-Means algorithm to generate the labeled data of four clusters: Excellent (E), Good (G), Okay (O), and Bad (B). Figure 4.10 displays the K-Mean clustered data with four clusters for the three application classes. Each different color represents a separate cluster.

Figure 4.10a shows the compute-intensive class labeled data. The lowest A2Cloud score and cost score values form the excellent class because low A2Cloud and cost scores are desirable (unlike RFC and cost ratings). On the other hand, the highest A2Cloud score and cost score belongs to bad class. Figure 4.11b represents the balanced class labeled data. The data-points in the left-bottom of the plot are the excellent class. On the other hand, the data-points close to the right-top of the graph constitute the bad case. Figure 4.10c exhibits the memory-intensive class labeled data. The excellent class contains the A2Cloud score and cost score with having lowest value. The right-top of the figure represents the bad class.

The output of the K-Means is the training dataset for the S-NB classifier. The dataset has input features: A2Cloud score and cost score. Also, the dataset contains the clusters number obtained from the K-Means clustering.

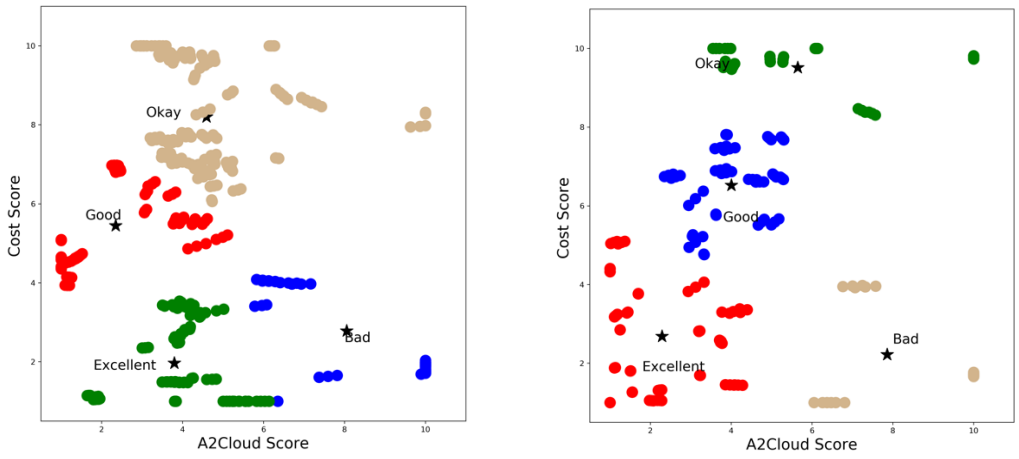



*(a)* Compute-intensive dataset *(b)* Balanced dataset

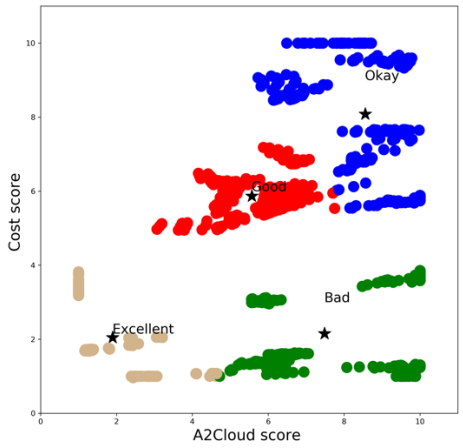



*(c)* Memory-intensive dataset

*Figure 4.9* A2Cloud score and cost score using A2Cloud and cost generator




*(a)* Compute-intensive labeled dataset *(b)* Balanced labeled dataset




*(c)* Memory-intensive labeled dataset

*Figure 4.10* Cluster generation using K-Means

#### 4.2.1.2 NB Classifier

Figure 4.11 represents the overall working principle of the S-NB classifier. The NB classifier trains using the K-Means results. During the NB model training, the NB classifier

transforms the dataset into a frequency table and prior probability. The model also computes the input features' mean and standard deviation. During the NB model testing, the NB model determines the posterior probability for each cluster by using the Naive Bayes equation. The highest posterior probability of a class represents the outcome of the prediction. For example, for a given test data=[A2Cloud score, Cost score]= [1.2, 1.2], the NB calculates the posterior probability for excellent, good, okay, and bad classes as follows: 0.6, 0.1, 0.2, and 0.1. The excellent class has the highest posterior probability. Therefore, the NB recommends the excellent class as output for the test data and passes the information to instance selector. Using this method, the classifier classifies all (A2Cloud score, cost score) into the four application cl assess.

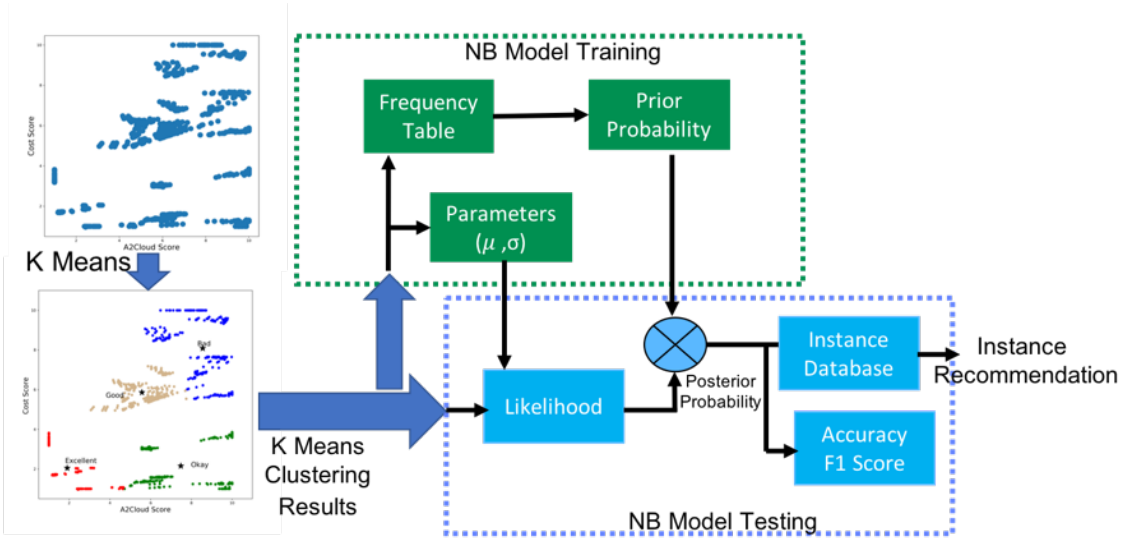



*Figure 4.11* S-NB training and testing phase methodology

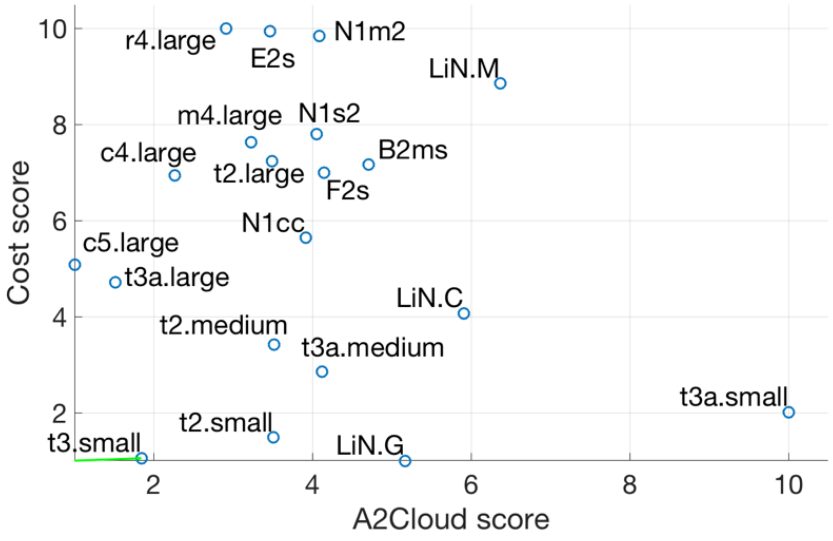



*Figure 4.12* Instance selector recommends instance with the least Euclidean distance

#### 4.2.1.3 Instance Selector

The instance selector selects the optimal instance from the excellent (E) class. The instance selector uses the Euclidean distance method to recommend the final instances. The instance selector determines the Euclidean distance of all instances (A2Cloud score and Cost score) in excellent class from the ideal tuple (1,1). We select the minimum value of the A2Cloud score and cost score (1,1) as ideal value. The instance selector recommends with the least Euclidean distance. Figure 4.12 represents an example of the S-NB Cloud instance selector. The instance selector recommends the t3.small instance because it hast the least Euclidean distance value 1.15 from the tuple (1,1).

The balanced and memory-intensive application follow the same methodology as compute-intensive to recommend the Cloud instances.

### 4.3 Summary

NB-Next datasets contain the RFC rating and cost rating. K-Means generates the four clusters (excellent, good, okay, and bad) from the training dataset. The NB classifier uses the clustered data for training and makes predictions using the verification dataset. The NB model calculates the posterior probability of four clusters (excellent, good, okay, and bad) for the test data. The highest posterior probability is the NB predicted class. The NB models transfer the excellent clusters information to the instance selector. The instance selector recommends the instance with has the largest Euclidean distance from the base tuple (1,1).

S-NB uses the A2Cloud score and cost score for training and verification studies. We apply K-Means clustering on the training dataset to create four clusters (excellent, good, okay, and bad). The NB calculates the posterior probability of the verification dataset. Then, the instance cluster is determined by its highest posterior probability. The S-NB passes the excellent cluster information to the instance selector. The instance selector calculated the Euclidean distance for each instance in the excellent cluster. Finally, the instance selector recommends the instance with the least Euclidean distance from the base tuple (1,1).

# CHAPTER 5: EXPERIMENTATION AND VERIFICATION

This chapter explains the experimentation and verification results for the Naive Bayes NEXT to Random Forest Classifier (NB-Next) and Standalone Naive Bayes classifier (S-NB). The chapter also outlines the NB-Next and S-NB in action.

## 5.1 Cloud Instances

We select a total of 20 Cloud instances from different Cloud service providers including AWS (Amazon Web Service) EC2 (Elastic Compute Cloud) [4], Microsoft Azure [5], Google Cloud Platform [6], and Linode [10]. Our tested Cloud instances, include the general-purpose, computation-optimized, and memory-optimized instances,which differ on the number of virtual CPUs, memory (GB), and cost-per-hour. Table 5.1 presents a list of tested Cloud instances together with their distinctive characteristics.

Section 5.2 presents an introduction to the real-world applications used for training and verification studies.

## 5.2 Real-world Applications Executed on Cloud Instances

We use several real-world scientific applications for dataset generation and verification. Our selected applications cover a wide range of scientific fields including: computer science, quantum chemistry, computer vision, hydrodynamics, and neural networks. Additionally, our study considers an application from each application category (compute-intensive, memory-intensive, and balanced) to verify the classifiers. Sections 5.2.1, 5.2.2, 5.2.3 describe the selected scientific applications used for training and verification.

Table 5.1
*Cloud Instances Categories: General-purpose, Compute and Memory Optimized*

| Type | Instance | vCPUs | Memory (GB) | Disk | Provider | Price (per hour) |
|---|---|---|---|---|---|---|
| General-Purpose | t2.large | 2 | 8 | Network SSD | AWS EC2 | $0.0928 |
| | t3a.large | 2 | 8 | Network SSD | AWS EC2 | $0.0753 |
| | t3.small | 2 | 2 | Network SSD | AWS EC2 | $0.0208 |
| | t3a.small | 2 | 2 | Network SSD | AWS EC2 | $0.0188 |
| | t3a.medium | 2 | 4 | Network SSD | AWS EC2 | $0.0376 |
| | m4.large | 2 | 8 | Network SSD | AWS EC2 | $0.1000 |
| | t2.small | 1 | 2 | Network SSD | AWS EC2 | $0.0230 |
| | t2.medium | 2 | 4 | Network SSD | AWS EC2 | $0.0464 |
| | B2ms | 2 | 8 | Network SSD | Azure VMs | $0.0912 |
| | N1s2 | 2 | 7.5 | Network SSD | GCP | $0.0200 |
| | Linode.G | 2 | 7.5 | Network SSD | Linode | $0.0150 |
| Compute-Optimized | c4.large | 2 | 8 | Network SSD | AWS EC2 | $0.1000 |
| | c5.large | 2 | 4 | Network SSD | AWS EC2 | $0.0850 |
| | F2s | 2 | 4 | Network SSD | Azure VMs | $0.0110 |
| | N1cc | 2 | 4 | Network SSD | GCP | $0.0150 |
| | Linode.C | 2 | 7.5 | Network SSD | Linode | $0.0450 |
| Memory-Optimized | r4.large | 2 | 15.25 | Network SSD | AWS EC2 | $0.1330 |
| | E2s | 2 | 16 | Network SSD | Azure VMs | $0.0782 |
| | N1m2 | 2 | 13 | Network SSD | GCP | $0.0250 |
| | Linode.M | 2 | 7.5 | Network SSD | Linode | $0.0900 |

### 5.2.1 LULESH

The Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH) solves the Sedov blast problem of hydrodynamics and presents solutions using numerical methods [54]. LULESH application has three problem sizes: 30, 50, and 70. The Arithmetic-Intensity generator (AIG) calculates the LULESH's arithmetic intensity as 0.23, 0.45, and 2.69 for LULESH problem sizes 30, 50, and 70, respectively. The AI value greater than zero indicates that LULESH's performs computations than memory accesses. Therefore, LULESH is

moderately compute-intensive application. We use LULESH 50 and 70 to train the machine learning model. We select LULESH 30 to verify the NB-Next and S-NB models.

### 5.2.2 Data Migration Scheduler

The data migration (DM) scheduler is a large simulation application that simulates the scheduling steps in large data centers [55], [56]. DM with flatten and color, DM greedy, Edge ranking and DM with space constraints are the different versions of the data migration application. The Data Migration with space constraints has the arithmetic intensity -5.7052. DM with space constraints has the higher number of memory access over the number of computations, which classifies this application as highly memory-intensive. We select the DM with space constraints to perform verification study and use other DM's to generate training dataset.

### 5.2.3 QODE

The University of the Pacific's chemistry department developed an Electron structure theory simulation application, QODE to simulate the electronic structure problem using the excitonically re-normalized coupled-cluster theory [57], [58]. The arithmetic intensity of QODE is -0.78, meaning that it falls within the balanced category class. We use QODE to verify the balanced class NB-Next and S-NB models.

### 5.2.4 Spiking Neural Networks

The Spiking Neural Networks (SNN) is a large scale neural network simulation models that mimic the human brain mechanism to use for character recolonization [59]. We use the Hodgin-Huxley (HH) model (compute-intensive application) [60], Wilson model (Balanced

application) [61], and the Izhikevich model (memory-intensive application) [62] for generating training dataset.

### 5.2.5 Rotoscope

The best features digital Rotoscope is a computer vision application that generates the artistic videos by adding animation to video sequences [63]. The rotoscope requires more memory access operation than computations (memory-intensive application). We use Rotoscope to generate the training dataset.

We choose real-world applications: LULESH, three SNN simulations, digital rotoscope, and three data-migration schedulers for generating training dataset.

Sections 5.3 and 5.4 describe the NB-Next in action, NB-Next model performance evaluation, S-NB in action, and S-NB model performance evaluation procedure.

## 5.3 NB-Next

NB-Next uses Cloud instance rating and cost rating for the NB-Next model training, testing, and verification. Sections 5.3.1 and 5.3.2 show the NB-Next in action and model performance evaluation, respectively.

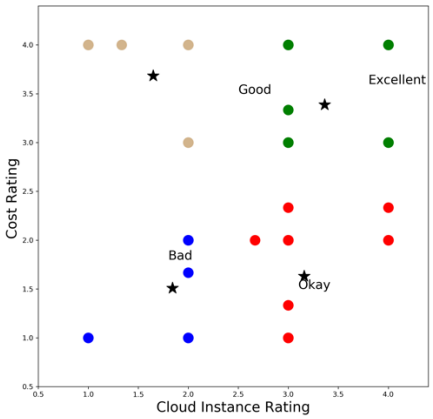
### 5.3.1 NB-Next in Action

NB-Next pulls the Cloud instance rating and cost rating datasets from the database, and generates the labels using K-means algorithm. We pick the compute-intensive dataset to explain how the NB classifier learns hypothesis parameters from the dataset (see Figure 5.1). NB converts the data into a frequency table and calculates the prior probability for the four clusters. Table 5.2 lists the calculated frequencies and the cluster prior probabilities which is the

frequency divided by the total number of data-points in the dataset. As seen in the table, the frequency column represents the number of data-points within a given cluster.

| | Rating | cost_rating | Cloud Instances | cluster |
|---|---|---|---|---|
| 0 | 3.000000 | 2.0 | t2.large | 3 |
| 1 | 3.000000 | 2.0 | B2ms | 3 |
| 2 | 3.000000 | 1.0 | N1s2 | 3 |
| 3 | 2.000000 | 4.0 | Lin.G | 2 |
| 4 | 4.000000 | 2.0 | c4.large | 3 |
| ... | ... | ... | ... | ... |
| 155 | 3.000000 | 4.0 | t2.small | 4 |
| 156 | 4.000000 | 3.0 | t3a.large | 4 |
| 157 | 3.000000 | 4.0 | t3a.medium | 4 |
| 158 | 1.333333 | 4.0 | t3a.small | 2 |
| 159 | 4.000000 | 4.0 | t3.small | 4 |

160 rows × 4 columns




*(a)* Compute-intensive training dataset *(b)* Dataset with four clusters

*Figure 5.1* Compute-intensive NB-Next training dataset

The NB classifier generates the training set parameters (mean and standard deviation) for the RFC rating and cost rating. Table 5.3 shows the training parameters of the NB classifier. The mean and standard deviation help the NB model to get insight into the clustered data. Using Tables 5.2 to 5.4, the NB model performs the prediction.

We verify the NB model with a test case (t3.small instance for LULESH 30 application) with RFC rating of 4.0 and cost rating of 4.0. The NB model begins with calculating the likelihood of the test data by using Equation 3.13. Therefore, there are two variables for the input data so that it calculates two sets of likelihood probabilities per cluster. Also, the model has already learned the prior probability of the cluster. Using the above-mentioned parameters, the NB classifier determines the posterior probability. Table 5.4 lists the likelihood, prior

probability, and posterior probability values for the test case. Table 5.4 shows that the excellent cluster has the highest posterior probability. Therefore, t3.small instance belongs to excellent cluster.

Table 5.2
*Frequency and Prior Probability of Compute-intensive Training*

| Cluster | 1 or bad | 2 or average | 3 or good | 4 or excellent | Total |
|---|---|---|---|---|---|
| Frequency | 19 | 19 | 55 | 67 | 160 |
| Prior Probability | 0.12 | 0.12 | 0.34 | 0.42 | 1.0 |

Table 5.3
*Mean and Standard Deviation of Compute-intensive Training Dataset*

| Cluster | Mean ( $\mu_{rating}$) | S.D ($\sigma_{Rating}$) | Mean ( $\mu_{cost\_rating}$) | S.D ( $\mu_{cost\_rating}$) |
|---|---|---|---|---|
| 1 or bad | 1.84 | 0.37 | 1.50 | 0.50 |
| 2 or average | 1.64 | 0.47 | 3.68 | 0.43 |
| 3 or good | 3.15 | 0.37 | 1.63 | 0.49 |
| 4 or excellent | 3.36 | 0.48 | 3.38 | 0.39 |

Table 5.4
*Testing Using LULESH 30 of T3.small Instance A2Cloud and Cost Rating 4, 4*

| | Likelihood | | | |
|---|---|---|---|---|
| Cluster | p(Rating\|Cluster) | p(Cost\|Cluster) | Prior Probability | Posterior Probability |
| 1 or bad | 4.59x10-5 | 2.97x10-6 | 0.12 | 1.63x10-11 |
| 2 or average | 2.84x10-6 | 0.67 | 0.16 | 2.29x10-7 |
| 3 or good | 0.078 | 6.77x10-6 | 0.34 | 1.77x10-7 |
| 4 or excellent | 0.34 | 0.35 | 0.42 | 0.05 |

### 5.3.2 Model Performance Evaluation

Model performance evaluation estimates the accuracy of the NB classifier using verification dataset. We select the confusion matrix, accuracy, and F1-score metrics to evaluate the classifiers.

We split the dataset into 80/20 ratio for training and testing purposes with the three real-world applications execute on the 20 Cloud instances. We then collect the runtime ratings via actual execution and the A2Cloud-RFC [11]. In addition, we calculate the cost rating by multiplying the instance-cost-per-time and runtime. K-Means generates clusters data from runtime and cost rating. We use this clustering result to evaluate the NB-Next's predictions.

Figure 5.2 shows the compute-intensive NB-Next classifier's testing and verification confusion matrix. The predicted label and true label present the predicted cluster and the actual cluster (derived from runtime analysis). Figure 5.2a exhibits the testing set confusion matrix. The NB-Next classifies all classes correctly. Therefore, the confusion matrix is diagonal. Figure

5.2b displays the verification set (LULESH 30) confusion matrix. The NB-Next model identifies 17 correct prediction out of 20. That makes the confusion matrix close to diagonal. The model performs misclassification of two good classes as average and one average class as good. It shows the conservative nature of the NB-Next model while rating instances.


*(a)* Testing set confusion matrix *(b)* LULESH 30 confusion matrix

*Figure 5.2* Compute-intensive NB-Next classifier confusion matrix

Figure 5.3 represents the memory-intensive NB-Next classifier's testing and verification confusion matrix. Figure 5.3a shows the testing set confusion matrix. The NB-Next identifies all points correctly that makes the confusion matrix diagonal. Figure 5.3b displays the verification set (Data Migration) confusion matrix. The NB-Next model makes 19 correct prediction out of 20 data points. One miss-prediction is where the NB-Next predicts a category as good but actually it is excellent. Although the NB-Next predicts a class as good but actually it is excellent, it represents the NB-Next's conservative nature.


*(a)* Testing set confusion matrix *(b)* Data migration confusion matrix

*Figure 5.3* Memory-intensive NB-Next classifier confusion matrix

Figure 5.4 represents the balanced NB-Next classifier's testing and verification confusion matrix. Figure 5.4a displays the testing set confusion matrix. The NB-Next identifies all points correctly that makes the confusion matrix strictly diagonal. Figure 5.4b displays the verification set (QODE) confusion matrix. The NB-Next model identifies 18 data point correctly out of 20 data points. Therefore, the QODE confusion matrix is almost diagonal. Although the NB-Next performs two miss-classification, it does not identify bad cluster as good or excellent. That means the NB-Next is conservative while making prediction.

Table 5.5 shows the NB-Next models performance parameters (accuracy and F1 score) for testing and verification datasets. For the testing dataset, the NB-Next exhibits high accuracy (100%) and F1 score (1.0). The model predicts all the data-points accurately and identifies all the possible positive labels. For the verification studies, the CI NB-Next model shows the accuracy and F1 score 85% and 0.84, respectively. The MI and balanced NB-Next models perform higher accuracy (>90%) and F1 score (>0.90).


*(a)* Testing set confusion matrix *(b)* QODE confusion matrix

*Figure 5.4* Balanced NB-Next classifier confusion matrix

Table 5.5
*Testing and Verification Set Accuracy and F1 Score*

| | Testing | | Verification | |
|---|---|---|---|---|
| | Accuracy (%) | F1 Score | Accuracy (%) | F1 Score |
| Compute-intensive | 100 | 1.0 | 85 | 0.84 |
| Memory-intensive | 100 | 1.0 | 95 | 0.95 |
| Balanced | 100 | 1.0 | 90 | 0.92 |

Figure 5.5 presents the runtime and cost rating of 20 Cloud instances for LULUESH 30 application. The t3.small is located the highest distance from the tuple (1,1). The instance selector recommends t3.small as best match for LULESH 30. The NB-Next also recommends t3.small instance for LULESH 30 which matches with the runtime and cost rating plot. The runtime plot suggests that t3a.large is the best for Data Migration application. The NB-Next also recommends t3a.large instance for Data Migration which verifies the NB-Next model.

Figure 5.7 shows the runtime and cost rating of 20 Cloud instances for balanced (QODE) application. The c5.large Cloud instance has the highest Euclidean distance 4.24 from the tuple (1, 1). The runtime plot suggests that c5.large is the best match for QODE application. The NB-Next also recommends c5.large instance for QODE which verifies the NB-Next model.

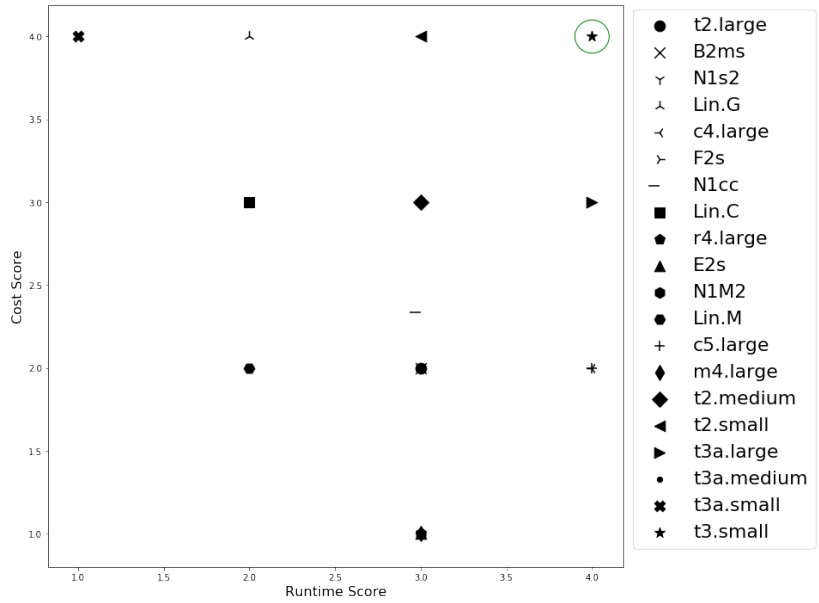



*Figure 5.5* The runtime and cost rating of 20 instances for LULESH 30

Figure 5.6 shows the runtime and cost rating of 20 Cloud instances for Data Migration application. The t3a.large Cloud instance has the Euclidean distance 3.72 from the tuple (1,1).

## 5.4 S-NB

The stand-alone Naive Bayes (S-NB) classifier working principle is discussed in the Chapter 4. In what follows, we explain the S-NB training and testing phase in details. Furthermore, the validation and instance recommendation are presented for three real-world applications.

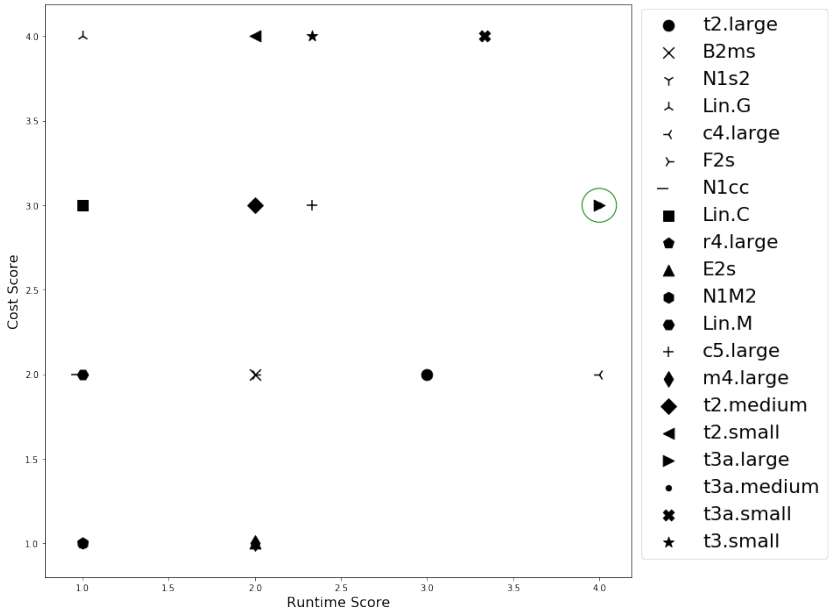



*Figure 5.6* The runtime and cost rating of 20 instances for Data Migration

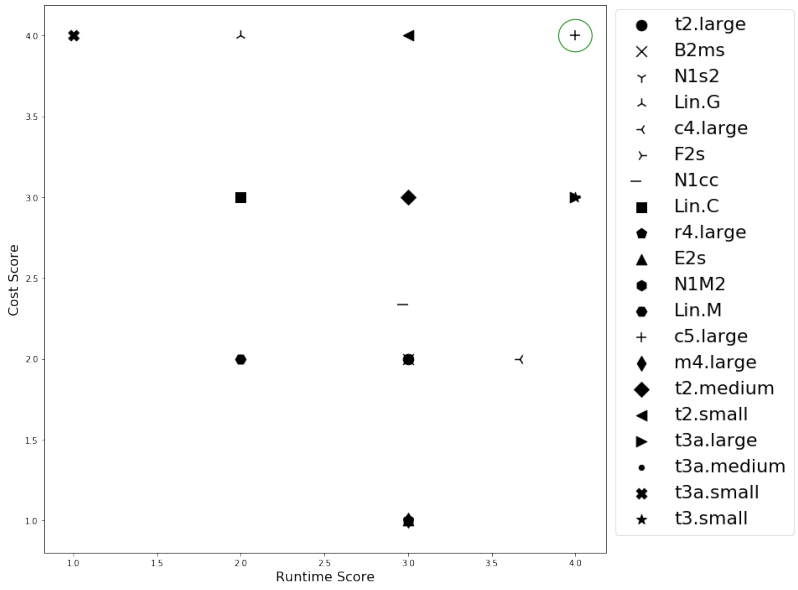


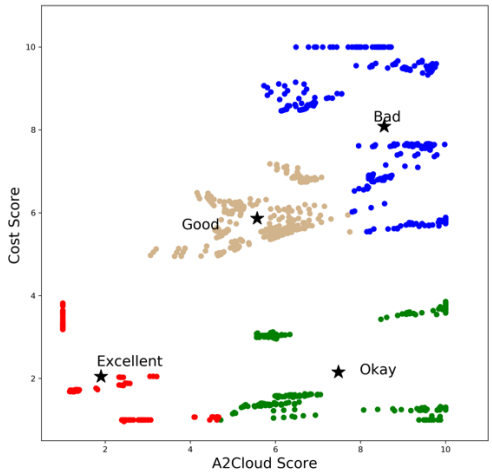

*Figure 5.7* The runtime and cost rating of 20 instances for QODE

### 5.4.1 S-NB in Action

In the S-NB approach, there are three separate NB classifiers used for instance recommendation: CI, MI, and balanced. We apply K-Means on the datasets to create clusters. Out of three application classes, we explain how the memory-intensive NB classifier training and testing phase because the other classes follow the same methodology.

| | a2cloud_score | cost_score | Instance | cluster |
|---|---|---|---|---|
| 0 | 4.728713 | 5.464126 | t2.large | 3 |
| 1 | 7.803559 | 10.000000 | r4.large | 1 |
| 2 | 6.114115 | 5.576482 | B2ms | 3 |
| 3 | 9.067434 | 7.635719 | N1s2 | 1 |
| 4 | 9.553885 | 7.660587 | N1s2 | 1 |
| ... | ... | ... | ... | ... |
| 1135 | 9.136618 | 5.724298 | N1cc | 1 |
| 1136 | 1.000000 | 3.306001 | t3a.large | 4 |
| 1137 | 6.557449 | 6.846846 | m4.large | 3 |
| 1138 | 5.958300 | 1.390324 | t3.small | 2 |
| 1139 | 9.553363 | 7.660113 | N1s2 | 1 |

1140 rows × 4 columns




*(a)* memory-intensive training dataset *(b)* Dataset with four clusters

*Figure 5.8* S-NB memory-intensive training dataset

The memory-intensive application class dataset has 1140 rows and 4 columns (see Figure 5.8). The NB classifier uses the A2Cloud score and cost columns to map its hypothesis function into the cluster value.

The NB classifier converts the dataset into the frequency distribution table for four clusters where the frequency means the number of samples per cluster. The prior probability (p[cluster]) is the frequency divided by the total number of samples. The frequency distribution

and the prior probability of the memory-intensive class dataset are shown in the Table 5.6. The NB classifier determines the model parameters such as mean and standard deviation for each class using the Gaussian distribution assumption. Table 5.7 presents the model training function parameters including mean $(\mu_{A2Cloud_{score}}, \mu_{Cost_{score}})$ and standard deviation $(\sigma_{A2Cloud_{score}}, \sigma_{Cost_{score}})$. Using Tables 5.6, 5.7, and 5.8, the S-NB model performs the prediction.

Table 5.6
*Frequency and Prior Probability of Memory-intensive Training*

| Cluster | 1 or bad | 2 or average | 3 or good | 4 or excellent | Total |
|---|---|---|---|---|---|
| Frequency | 345 | 339 | 286 | 170 | 1140 |
| Prior Probability | 0.30 | 0.29 | 0.26 | 0.15 | 1.0 |

We test the NB model with a test case (t3a.medium instance for Data Migration application) with A2Cloud score of 1.16 and cost rating cost score 1.68. The NB model calculates the likelihood of the test data by using the equation 3.13. Table 5.8 lists the likelihood, prior probability, and posterior probability values for the A2Cloud score=1.16 and cost score= 1.68. The excellent cluster has the highest posterior probability (1.68x10-2). So, the S-NB identifies the t3a.medium as excellent instance for Data Migration. S-NB passes the information to instance selector to make final decision using Euclidean distance.

Table 5.7
*Mean and Standard Deviation of Memory-intensive Training Dataset*

| Cluster | $Mean$ $(\mu_{A2Cloudscore})$ | S.D $(\sigma_{A2Cloudscore})$ | $Mean$ $(\mu_{Cost\_score})$ | S.D $(\sigma_{cost\_score})$ |
|---|---|---|---|---|
| 1 or bad | 8.55 | 1.11 | 8.80 | 1.48 |
| 2 or average | 7.45 | 1.88 | 2.14 | 1.01 |
| 3 or good | 5.56 | 0.87 | 5.86 | 0.57 |
| 4 or excellent | 1.88 | 1.06 | 2.05 | 0.99 |

Table 5.8
*Testing Using T3a.medium Instance A2Cloud and Cost Scores 1.16 and 1.68*

| | Likelihood | | | |
|---|---|---|---|---|
| Cluster | P (A2Cloud score \| Cluster) | P (Cost score \| Cluster) | Prior Probability | Posterior Probability |
| 1 or bad | $8.53x10^{-11}$ | $2.34x10^{-5}$ | 0.30 | $5.99x10^{-16}$ |
| 2 or average | $7.5x10^{-4}$ | 0.35 | 0.29 | $7.84x10^{-5}$ |
| 3 or good | $1.27x10^{-6}$ | $1.47x10^{-12}$ | 0.26 | $4.89x10^{-19}$ |
| 4 or excellent | 0.29 | 0.37 | 0.15 | $1.68x10^{-2}$ |

### 5.4.2 Model Performance Evaluation

For the training and testing purposes, the dataset is split into a 80/20 ratio. The verification dataset is derived from real-world applications: LULESH 30, Data Migration, and QODE.

Figure 5.9 shows the compute-intensive S-NB model performance visualization using the confusion matrix. The testing set confusion matrix represents that S-NB classifies single data-point as average class instead of good class; aside from this the S-NB performs well on testing dataset. Figure 5.9b represents the LULESH 30 verification set confusion matrix. We observe that the S-NB has miss-classified two Cloud instances out of 20. S-NB predicts a good instance and an average class instance as excellent and bad class, respectively. Although S-NB makes two false predictions, but those pre- dictions are no more than the category apart. This characteristic of S-NB shows the conservative nature. Overall, the S-NB model accuracy and F1 score for LULESH 30 are 90% and 0.90 enlists in the Table 5.9.

Figure 5.10 exhibits the memory-intensive S-NB model confusion matrix. For the testing set (Figure 5.10a), the S-NB has the almost diagonal confusion matrix that represents the S-NB per- forms correct predictions on testing set. Figure 5.10b shows the Data Migration application verification confusion matrix. The S-NB performs excellent because the confusion matrix is diagonal or almost diagonal. The model predicated one instance as good instead of average class.


*(a)* Compute-intensive testing set confusion matrix *(b)* Verification (LULESH 30) confusion matrix

*Figure 5.9* The S-NB compute-intensive application class classifier confusion matrix


*(a)* Memory-intensive testing set confusion matrix *(b)* Data Migration verification confusion matrix

*Figure 5.10* The S-NB memory-intensive application class classifier confusion matrix

Figure 5.11 exhibits the balanced application S-NB model's confusion matrix. The S-NB has the almost diagonal confusion matrix that expresses the S-NB model high accuracy on

testing dataset (see Figure 5.11a). For the QODE application verification, the S-NB exhibits excellent performance because the confusion matrix is diagonal or almost diagonal (see Fig. 5.11b). The model predicated two instances as bad instead of average class.


*(a)* Balanced class testing set confusion matrix *(b)* Verification (Qode) confusion matrix

*Figure 5.11* The S-NB QODE application class classifier confusion matrix

Table 5.9 shows the S-NB models performance parameters (accuracy and F1 score) for testing and verification datasets. The original dataset divides into training dataset (80%) and testing dataset (20%). For the testing dataset, the S-NB exhibits the high accuracy (>97%) and F1 score (>0.98). The model predicts all the data-points accurately and identifies all the possible positive labels. For the verification studies, the CI S-NB model shows the accuracy and F1 score 90% and 0.90, respectively. The MI S-NB model has the accuracy and F1 score 95% and 0.93, respectively. The balanced S-NB model shows the 90% accuracy and 0.90 F1 score.

Table 5.9
*Accuracy and F1 Score of Testing and Verification Dataset*

| | Testing | | Verification | |
|---|---|---|---|---|
| | Accuracy (%) | F1 Score | Accuracy (%) | F1 Score |
| Compute-intensive | 99.07 | 0.99 | 90 | 0.90 |
| Memory-intensive | 99.58 | 9.98 | 95 | 0.93 |
| Balanced | 97.11 | 0.98 | 90 | 0.91 |

Figure 5.12 presents the runtime and cost rating of 20 Cloud instances for LUESH 30 application. The S-NB instance selector recommends the instance that has the minimum distance from the base tuple (1,1). The base tuple is (1,1) because an instance could have minimum (1,1) runtime score and cost score. The t3.small has the least Euclidean distance (1.20) from the tuple (1,1). The instance selector recommends t3.small as best match for LULESH 30. The NB-Next also recommends t3.small instance for LULESH 30 which matches with the runtime and cost rating plot.

Figure 5.13 shows the runtime and cost rating of 20 Cloud instances for Data Migration application. The t3a.medium Cloud instance has the Euclidean distance 1.07 from the tuple (1,1). The runtime plot suggests that t3a.medium is the best for Data Migration application. The S-NB also recommends t3a.medium instance for Data Migration which verifies the NB-Next model.

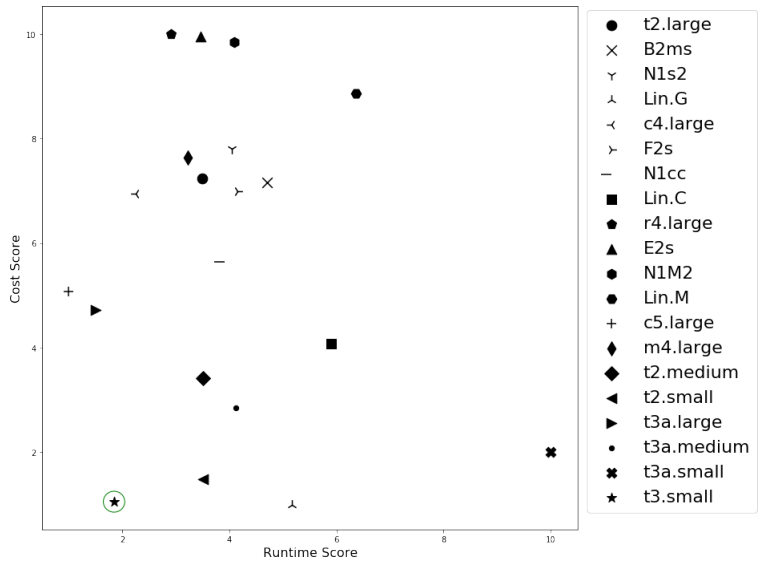



*Figure 5.12* The runtime and cost score of 20 instances for LULESH 30

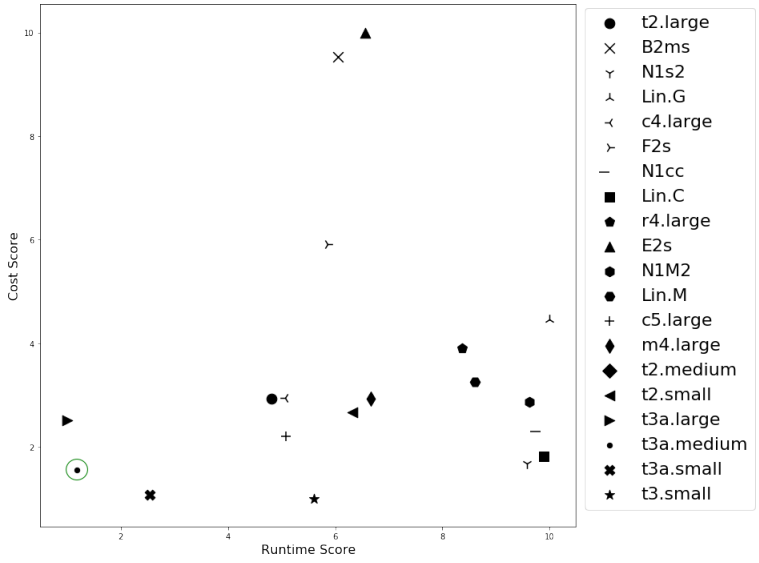



*Figure 5.13* The runtime and cost score of 20 instances for Data Migration

Figure 5.14 shows the runtime and cost rating of 20 Cloud instances for balanced (QODE) application. The t3.small Cloud instance has the highest Euclidean distance 1.11 from the tuple (1,1). The runtime plot suggests that t3.small is the best match for QODE

application. The NB-Next also recommends t3.small instance for QODE which verifies the NB-Next model.

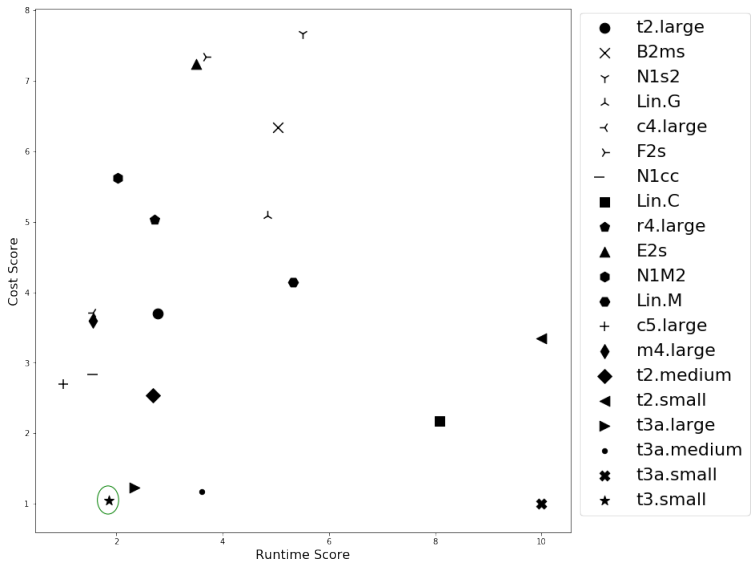



*Figure 5.14* The runtime and cost score of 20 instances for QODE

## 5.5 Summary

We use eight scientific applications and 20 Cloud instances for generating training dataset. To verify the NB-Next and S-NB models, we use LULESH 30, QODE, Data Migration with space constraints applications.

NB-Next uses the RFC rating and cost rating for model training. To verify the NB-NEXT, the CI NB-Next model shows the accuracy and F1 score 85% and 0.84, respectively. The MI and balanced NB-Next models perform higher accuracy (> 90%) and F1 score (> 0.90).

S-NB uses the A2Cloud score and cost score for training purposes. For the verification study, the CI S-NB model shows the accuracy and F1 score 90% and 0.90, respectively. The MI S-NB model has the accuracy and F1 score 95% and 0.93, respectively. The balanced S-NB

model shows the 90% accuracy and 0.90 F1 score. The S-NB methodology shows the higher accuracy over NB-Next.

# CHAPTER 6: CONCLUSION

We present the NB-Next and S-NB classifiers for the Cloud instance selection. Both of the methods simplify the A2Cloud-RFC based recommender system using a Naive Bayes classifier. The A2Cloud-RFC framework profiles the scientific applications and Cloud instances without executing the applications on Cloud instance, which saves unnecessary execution costs on the Cloud. The A2Cloud-RFC framework utilizes the application performance and cloud performance characteristics to generate scores; Those scores represent a scientific application's runtime and cost on the targeted instances, thereby producing the first level of instance recommendation. The generated results are stored in a database to build the Cloud instance recommendation system using the Naive Bayes Classifier.

The NB-Next is comprised of A2Cloud-RFC framework, K-Means, Naive Bayes classifier, and an instance selector. The K-Means takes the cloud rating and cost rating as input from the A2Cloud-RFC framework and divides the dataset into four clusters: E, G, O, and B. The Naive Bayes trains with the clustered dataset to identify the Cloud instance clusters. The instance selector selects the instance from an excellent class using our proposed Euclidean distance. The RFC rating and cost rating are the higher the better. The NB-Next trains with LULESH, Data Migration, Rotoscope, and Spiking neural networks scientific applications over 20 Cloud instances. The shows an accuracy of over 85% and F1 score over 0.84 in the verification dataset.

The S-NB comprises of A2Cloud-ext engine, K-Means, Naive Bayes classifier, and instance selector. The A2Cloud-ext engine generates the A2Cloud score and cost score. The K-Means use the A2Cloud score and cost score and forms the four clusters: E, G, O, and B. The

Naive Bayes trains with the K-Means output and predicts the instances clusters. The lower the A2Cloud score and cost score are better. The instance selector pulls the instance from the excellent cluster and recommends an instance with the least Euclidean distance. The S-NB model shows an accuracy of over 90% and F1 score over 0.90 in the verification dataset.

The NB-Next include the random forest classifier. The inclusion of random forest classifier makes the NB-Next methodology more complex. It shows the average accuracy approximately 90%. In contrast, the S-NB has the simple methodology with NB classifier. It has the accuracy approximately 92%. The HPC should select the S-NB methodology to get instance recommendation.

Our proposed methodologies (NB-Next and S-NB) provide a cost-effective guidance for scientific community particularly small private/public organizations and universities to select Cloud resources. Furthermore, the proposed machine learning approaches require small training dataset and less training time. In the future, we propose to explore other machine learning algorithms such as Support vector machine or Neural Network for solving classification problems. For the existing model, we only use two input features. Additionally, we can add other Cloud instance network components i.e. latency.

# REFERENCES

[1] P. Ruiu, O. Terzo, G. Carlino, et al. “HPC Cloud Pills: On-Demand Deployment and Execution of HPC Application in Cloud Environments”. In: 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. 2014, pp. 82–88. doi: 10. 1109 / 3PGCIC.2014.39.

[2] J. Emeras, S. Varrette, and P. Bouvry. “Amazon Elastic Compute Cloud (EC2) vs. In-House HPC Platform: A Cost Analysis”. In: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD). 2016, pp. 284–293. doi: 10.1109/CLOUD.2016.0046.

[3] G. Fox and D. Gannon. “Using Clouds for Technical Computing”. In: Cloud Computing and Big Data (2013).

[4] Amazon EC2 Instances Types. url: *https://aws.amazon.com/ec2/instance-types/.*

[5] Microsoft Azure Instances Types. url: *https://azure.microsoft.com/en- us/pricing/ details/virtual-machines/linux/#a-series.*

[6] Google Cloud Platform Instances Types. url: https://cloud.google.com/compute/vm-instance-pricing.

[7] IBM Cloud. url: *https://www.ibm.com/cloud.*

[8] Oracle Cloud. url: *https://www.oracle.com/cloud/.*

[9] Alibaba Cloud. url: *https://us.alibabacloud.com/en.*

[10] Linode Instances Types. url: *https://www.linode.com/products/high-memory/.*

[11] D. Samuel, S. Khan, C.J. Balos andZ. Abuelhaj, et al. “A2Cloud-RF: A Random Forest based Statistical Framework to guide Resource Selection for High-Performance Scientific Computing on the Cloud”. In: Conc. and Comp.: Pract. (2019).

[12] E. Rolo, M. Diener, L. P. Gaspary, et al. “HPC Application Performance and Cost Efficiency in the Cloud”. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). 2017, pp. 473–477. doi: 10.1109/PDP.2017.59.

[13] T. K. Okada, A. Goldman, and G. G. H. Cavalheiro. “Using NAS Parallel Benchmarks to evaluate HPC performance in clouds”. In: 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA). 2016, pp. 27–30. doi: 10.1109/NCA.2016.7778587.

[14] I. K. Kim, J. Steele, Y. Qi, et al. “Comprehensive Elastic Resource Management to Ensure Predictable Performance for Scientific Applications on Public IaaS Clouds”. In: 2014

IEEE/ACM 7th International Conference on Utility and Cloud Computing. 2014, pp. 355– 362. doi: 10.1109/UCC.2014.45.

[15] Zhenhuan Gong, Xiaohui Gu, and J. Wilkes. "PRESS: Predictive Elastic Resource Scaling for cloud systems". In: 2010 International Conference on Network and Service Management. 2010, pp. 9–16. doi: 10.1109/CNSM.2010.5691343.

[16] "A framework for ranking of cloud computing services". In: Future Generation Computer Systems 29.4 (2013), pp. 1012–1023. doi: https://doi.org/10.1016/j.future.2012.06.
006.

[17] A. Iosup, S. Ostermann, M. N. Yigitbasi, et al. "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing". In: IEEE Transactions on Parallel and Distributed Systems 22.6 (2011), pp. 931–945. doi: 10.1109/TPDS.2011.66.

[18] R. Chard, K. Chard, R. Wolski, et al. "Cost-Aware Cloud Profiling, Prediction, and Provisioning as a Service". In: IEEE Cloud Computing 4.4 (2017), pp. 48–59. doi: 10.1109/MCC.
2017.3791025.

[19] R. Chard, K. Chard, B. Ng, et al. "An Automated Tool Profiling Service for the Cloud". In: 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). 2016, pp. 223–232. doi: 10.1109/CCGrid.2016.57.

[20] A. A. Bankole and S. A. Ajila. "Predicting cloud resource provisioning using machine learning techniques". In: 2013 26th IEEE Canadian Conference on Electrical and Computer Engineer- ing (CCECE). 2013, pp. 1–4. doi: 10.1109/CCECE.2013.6567848.

[21] Taiyang Guo, Rami Bahsoon, Tao Chen, et al. "Cloud Instance Selection Using Parallel K-Means and AHP". In: UCC '19 Companion. Auckland, New Zealand: Association for Computing Machinery, 2019, pp. 71–76. isbn: 9781450370448. doi: 10.1145/3368235.3368845. url: https://doi.org/10.1145/3368235.3368845.

[22] W. Liu, P. Wang, Y. Meng, et al. "A Novel Algorithm for Optimizing Selection of Cloud In- stance Types in Multi-Cloud Environment". In: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). 2019, pp. 167–170. doi: 10.1109/ICPADS47876.
2019.00033.

[23] F. Samreen, Y. Elkhatib, M. Rowe, et al. "Daleel: Simplifying cloud instance selection using machine learning". In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. 2016, pp. 557–563. doi: 10.1109/NOMS.2016.7502858.

[24] X. Ouyang, C. Wang, R. Yang, et al. "ML-NA: A Machine Learning Based Node Performance Analyzer Utilizing Straggler Statistics". In: 2017 IEEE 23rd International Conference on

Parallel and Distributed Systems (ICPADS). 2017, pp. 73–80. doi: 10.1109/ICPADS.2017.00021.
[25] A. Kaplunovich and Y. Yesha. “Cloud big data decision support system for machine learning on AWS: Analytics of analytics”. In: 2017 IEEE International Conference on Big Data (Big Data). 2017, pp. 3508–3516. doi: 10.1109/BigData.2017.8258340.

[26] G. M. Wamba, Y. Li, A. Orgerie, et al. “Cloud Workload Prediction and Generation Models”. In: 2017 29th International Symposium on Computer Architecture and High-Performance Computing (SBAC-PAD). 2017, pp. 89–96. doi: 10.1109/SBAC-PAD.2017.19.

[27] M. Sun, T. Zang, X. Xu, et al. “Consumer-Centered Cloud Services Selection Using AHP”. In: 2013 International Conference on Service Sciences (ICSS). 2013, pp. 1–6. doi: 10.1109/ICSS.2013.26.

[28] C. Chen and Kuan-Hung Lin. “A decision-making method based on interval-valued fuzzy sets for cloud service evaluation”. In: 4th International Conference on New Trends in Information Science and Service Science. 2010, pp. 559–564.

[29] J. P. Ashwini, C. Divya, and H. A. Sanjay. “Efficient resource selection framework to enable cloud for HPC applications”. In: 2013 4th International Conference on Computer and Communication Technology (ICCCT). 2013, pp. 34–38. doi: 10.1109/ICCCT.2013.6749599.

[30] S. Rathnayake, D. Loghin, and Y. M. Teo. “CELIA: Cost-Time Performance of Elastic Applications on Cloud”. In: 2017 46th International Conference on Parallel Processing (ICPP). 2017, pp. 342–351. doi: 10.1109/ICPP.2017.43.

[31] F. J. A. Morais, R. Lopes, and F. Brasileiro. “Instance Type Selection in Proactive Horizontal Auto-Scaling”. In: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). 2016, pp. 102–109. doi: 10.1109/CloudCom.2016.0031.

[32] S. Grandhi and S. Wibowo. “Performance evaluation of cloud computing providers using fuzzy multiattribute group decision making model”. In: 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). 2015, pp. 130–135. doi: 10.1109/FSKD.2015.7381928.

[33] O. Sohaib and M. Naderpour. “Decision making on adoption of cloud computing in e-commerce using fuzzy TOPSIS”. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). 2017, pp. 1–6. doi: 10.1109/FUZZ-IEEE.2017.8015404.

[34] C. Balos, D. De La Vega, Z. Abuelhaj, et al. “A2Cloud: An Analytical Model for Application- to-Cloud Matching to Empower Scientific Computing”. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). 2018, pp. 548–555. doi: 10.1109/CLOUD.2018.00076.

[35] G. Ofenbeck, R. Steinmann, V. Caparros, et al. “Applying the roofline model”. In: 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). 2014, pp. 76–85.

[36] A. Heinecke, K. Vaidyanathan, M. Smelyanskiy, et al. “Design and Implementation of the Linpack Benchmark for Single and Multi-node Systems Based on Intel® Xeon Phi Coprocessor”. In: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing. 2013, pp. 126–137.

[37] John D. McCalpin. STREAM: Sustainable Memory Bandwidth in High Performance Comput- ers. Tech. rep. A continually updated technical report. http://www.cs.virginia.edu/stream/. Charlottesville, Virginia: University of Virginia, 1991-2007. url: http://www.cs.virginia.edu/stream/.

[38] M. Reitzner. “Central limit theorems for Gaussian polytopes”. In: Probab. Theory Relat. Fields (2005).

[39] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. “Machine learning: a review of classification and combining techniques”. In: Artif Intell Rev 26 (2006).

[40] K. Rajeswari, O. Acharya, M. Sharma, et al. “Improvement in k-Means Clustering Algorithm Using Data Clustering”. In: 2015 International Conference on Computing Communication Control and Automation. 2015, pp. 367–369.

[41] S. Na, L. Xumin, and G. Yong. “Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm”. In: 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. 2010, pp. 63–67.

[42] K. Nugroho, E. Noersasongko, Purwanto, et al. “Improving Random Forest Method to Detect Hate speech and Offensive Word”. In: 2019 International Conference on Information and Communications Technology (ICOIACT). 2019, pp. 514–518.

[43] M. S. Kumar, V. Soundarya, S. Kavitha, et al. “Credit Card Fraud Detection Using Random Forest Algorithm”. In: 2019 3rd International Conference on Computing and Communications Technologies (ICCCT). 2019, pp. 149–153.

[44] X. Liu, D. Wang, L. Jiang, et al. “A novel method for inducing ID3 decision trees based on variable precision rough set”. In: 2011 Seventh International Conference on Natural Computation. Vol. 1. 2011, pp. 494–497.

[45] SCIKIT-LEARN: DECISION TREE LEARNING I- ENTROPY, GINI, AND INFORMATION GAIN. Available at https:// www. bogotobogo. com/ python/ scikit-learn/scikt_machine_learning_Decision_Tree_Learning_Informatioin_Gain_IG_Impurity_Entropy_Gini_Classification_Error.php (03/25/2020).

[46] Jason Brownlee. Information Gain and Mutual Information for Machine Learning. Available at https://machinelearningmastery.com/information-gain-and-mutual-information/ (03/26/2020).

[47] Jason Brownlee. Decision Tree. Available at https://www.hackerearth.com/practice/ machine - learning / machine - learning - algorithms / ml - decision - tree / tutorial/ (03/26/2020).

[48] Kamal Nigam Andrew Mccallum. A Comparison of Event Models for Naive Bayes Text Classification. Available at https:// www. cs. cmu. edu/~knigam/ papers/ multinomial-aaaiws98.pdf (03/31/2020).

[49] Kevin P. Murphy. Na¨ıve Bayes Classifications. Available at https://www.ic.unicamp.br/ ~rocha/teaching/2011s1/mc906/aulas/naive-bayes.pdf (04/02/2020).

[50] Irina Rish. "An Empirical Study of the Na¨ıve Bayes Classifier". In: IJCAI 2001 Work Empir Methods Artif Intell 3 (Jan. 2001).

[51] Zhihua Cai Jia Wu. "A naive Bayes probability estimation model based on self-adaptive differential evolution". In: Journal of Intelligent Information Systems 42 (2013), pp. 671– 694.

[52] George H. 5. John, Pat Langley, and G. Yong. "). Estimating Continuous Distributions in Bayesian Classifiers". In: 1995 Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. 1995, pp. 338–345.

[53] L. Mandal and N. D. Jana. "A Comparative Study of Naive Bayes and k-NN Algorithm for Multi-class Drug Molecule Classification". In: 2019 IEEE 16th India Council International Conference (INDICON). 2019, pp. 1–4.

[54] Ian Karlin, Jeff Keasler, and Rob Neely. Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH). Available at https:// computing. llnl. gov/ projects/ codesign/lulesh (04/14/2020).

[55] G. Roberts, S. Chen, C. Kari, et al. "Data migration algorithms in heterogeneous storage systems: A comparative performance evaluation". In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA). 2017, pp. 1–4.

[56] S. Chen, C. Kari, and M. Coolbeth. "Data Migration in Large Scale Heterogeneous Storage Systems with Space Constraints". In: 2020 International Conference on Computing, Networking and Communications (ICNC). 2020, pp. 358–362.

[57] Yuhong Liu and Anthony Dutoi. "Excitonically renormalized coupled-cluster theory". In: Molecular Physics 117 (Sept. 2018), pp. 1–16. doi: 10.1080/00268976.2018.1523481.

[58] Anthony Dutoi and Yuhong Liu. "Systematically improvable excitonic Hamiltonians for electronic structure theory". In: Molecular Physics 117 (Sept. 2018), pp. 1–15. doi: 10.1080/ 00268976.2018.1522003.

[59] M. A. Bhuiyan, V. K. Pallipuram, M. C. Smith, et al. “Acceleration of spiking neural networks in emerging multi-core and GPU architectures”. In: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW). 2010, pp. 1–8.

[60] Huxley AF Hodgkin AL. “A quantitative description of membrane current and application to conduction and excitation in nerve”. In: J Physiol. 1952; 117:500-544 (Aug. 1952), pp. 500–544.

[61] Wilson HR. “Simplified dynamics of human and mammalian neocortical neurons”. In: Journal of Theoretical Biology (1999), pp. 375–388. doi: https://doi.org/10.1006/jtbi.1999. 1002.

[62] Izhikevich EM. “Simple model to use for cortical spiking neurons”. In: IEEE Trans Neural Network (2003), pp. 1569–1572.

[63] I. Murphy, T. Norlund, and V. K. Pallipuram. “A best-features based digital rotoscope”. In: 2017 51st Asilomar Conference on Signals, Systems, and Computers. 2017, pp. 243–247.