Chapter 4
Estimation of Sharing Dependencies in Personal Storage Clouds Using Ensemble Learning Approaches

S. Poonkuntran and J. Manessa

4.1 Dynamic Provisioning of Cloud Resources in Multi-Cloud Environment

Cloud computing is an emerging trend in all kinds of industries starting from data collection to prepare detailed analysis and conclusions. It becomes a preference for industries in different ways. They include no need to invest in infrastructure, an infinite amount of resource capacity, and no need to wait for several months to build infrastructure to start the business. The cloud provides complete virtualization where user can hire their required resources at competitive prices. They need to pay only for the resource they are using. The users can acquire their resources dynamically, and the same can elastically be customized based on the changing needs at different times [1].

Many architectures have been proposed and practiced by companies. Multi-cloud is one particular architecture where the services will be availed from different vendors for single network architecture. It is different from the hybrid cloud that provides a heterogeneous environment composed of different infrastructure environments such as the public and private cloud. There are three main reasons for choosing a multi-cloud environment.

Choice: The cloud service vendors are chosen by the companies for their needs, and they have flexibility in choosing the vendors. It helps the companies to avoid vendor lock-in.

S. Poonkuntran (✉)
School of Computer Science and Engineering, VIT Bhopal University, Bhopal, India
e-mail: poonkuntran.s@vitbhopal.ac.in

J. Manessa
Freelance Trainer, Madurai, India

R. Nagarajan et al. (eds.), *Operationalizing Multi-Cloud Environments*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-74402-1_4

Disaster Avoidance: The multi-cloud environment provides 100% availability of resources for computation and storage if any loss happens due to human errors or disaster. It helps the companies to avoid downtime.

Compliance: The multi-cloud environments help the companies to achieve their goals for governance, risk management, and compliance regulations.

The dynamic and elasticity are important features of multi-cloud service for which users no need to do anything from their side. The service provider of cloud computing needs to take care provisioning of the resources to the different users based on their changing needs over different times. This cannot be simply done and needs to have a good mechanism to exactly predict the number of resources required to execute the computation, improve utilization of resources, and minimize the cost.

The resource provisioning includes the selection of appropriate hardware resources (Processor, RAM, Storage, Networking), deployment, OS requirements, creating a runtime environment, and providing necessary software and its management to ensure the guaranteed performance of the application as per service level agreement (SLA). Every service will be offered through an SLA executed between provider and consumer of the cloud.

The SLA will provide QoS parameters which include availability, reliability, response time, throughput, security, and performance. The provisioning of resources needs to be done without affecting the SLA and QoS. The resource provisioning will have four different styles. They are static, dynamic, static/dynamic, and user self-service. In static, the resource provisioning will be fixed irrespective of workload and requirements. It is suffered from underprovisioning and overprovisioning. In underprovisioning, few jobs will never be executed, since no resources are provisioned for them. In overprovisioning, few jobs will have more resources than required, and it becomes a waste. Both situations will lead to ineffective utilization of resources. The dynamic provisioning will allocate resources based on the demands, and it is not fixed. It is decided based on different parameters which include demand, events, and popularity. Here, the allocation of resources will be varied over time based on requirements. Predicting the exact requirement will be a challenge in dynamic provisioning. Sometimes, the prediction will not be possible due to frequent requirement changes. Hence, the third style static/dynamic is used. It is a hybrid style where static provisioning will be done in cases where dynamic provisioning is not feasible. The fourth style is user self-service where the user will decide the requirements and purchase the resources [1, 2].

The dynamic resource provisioning targeted response time, minimization of the cost of the services, maximization of company profit, fault tolerance, reducing SLA violation, and efficient power consumption [3]. The prediction of the pattern of resource consumption in cloud computing is very crucial and important in the dynamic provisioning of resources. It can only serve as a base for the provisioning. It raises the demand for the prediction of workload patterns in cloud computing. The workload parameters are different from one application to another running on the cloud.

The social networking applications which require websites to be run on the cloud need higher bandwidth and storage requirements. The scientific application which requires computing infrastructure to be run on cloud needs high computing powers. The e-Commerce applications require their application to be run on cloud needs current load parameters. Similarly, the different parameters will be used to measure workload parameters based on different applications.

Many companies are presently shifting their business to the multi-cloud to avail the benefits of elasticity and pay as you go cost molds. Such an environment will have several public clouds and private clouds based on the company's business needs. Such an environment demands storage solutions in the cloud. Thereby, the storage cloud becomes popular today [4–19].

4.1.1 Storage Cloud

The storage cloud provides storage solutions to the user where the user can store their data on the cloud through the Internet and access them being anywhere. It provides anytime anywhere access to the user's data. The cloud storage provider delivers your data on demand, just in time fashion and cheaper cost. It makes users avoid buying their own storage devices. The cloud storage vendors need to have a well scalable infrastructure to provide storage services to the users on-demand and pay as you go model. They need to manage their capacity, security, and durability to provide access to your data [4–19].

The storage cloud takes many advantages, and the major three advantages are listed below.

1. No need to invest in hardware infrastructures to create our storage facilities. Users need to avail of the entire storage solution on the cloud, and they can pay as per their usage. Usually, the two parameters will be used to measure the consumption of storage resources in the cloud. They are the amount of storage purchased and how frequently the storage is accessed for storage and retrieval. This model brings different cost metrics based on the data access frequency. If a particular data is created and it is not frequently accessed, this will come to the lesser cost models.
2. The storage cloud provides on-time deployment of resources where any additional resources can be purchased in time without worries about installation and deployment. It can quickly be added to your subscription. We can find additional resources dynamically on the go, no need to fix it in advance. However, it requires well prediction techniques to forecast future needs.
3. The cloud storage provides centralized data management where all the data is stored centrally in the cloud, and it can be easily accessed, shared, and migrated online. It can also be done anywhere and anytime.

Cloud storage supports different types of formats in which the data can be stored and retrieved. It includes object storage, file storage, and block storage. The object storage stores the objects that are metadata of the data to be stored. It allows us to build our storage solution from scratch. The Amazon Simple Storage Service (S3) is an example of object storage. It can be widely used to import, backup, and archive our data.

The second type is file storage where the application needs a file system through which it will share the files between users. Usually, file storage is linked to Network Attached Storage (NAS). File storage is widely used by enterprises and users to store their information as files and storing media files. The Amazon Elastic File System (EFS) is an example of file storage.

The third type is block storage where the application will have low latency and variable size storage for all the users connected to it. For example, the ERP systems will need individual storage to work with databases. Usually, this storage is facilitated through Storage Area Networks (SAN). The Amazon Elastic Block Store (EBS) is an example of block storage.

From the storage cloud revolution, personal cloud storage has come up now. Personal cloud storage provides cloud storage solutions to individuals for storing their files, photos, and videos. As in cloud storage, it enables individuals to access their files at any time, anywhere [4–19].

4.1.2 Dynamic Provisioning of Personal Storage Cloud

The dynamic provisioning of personal storage cloud includes volume creation, ensuring the availability of files in time, preparing the necessary number of copies of the same file for parallel processing, ensuring the security of the contents, protecting ownership rights, and measuring sharing dependency. The sharing dependency is a key parameter in the storage cloud by which how files are being shared among the number of users. This will be an important parameter for predicting the workload patterns in personal storage clouds. The majority of the user who uses the personal cloud accounts is mainly for sharing their contents with their dears and nears.

The sharing dependency helps the cloud service provider to estimate the resources and provisioning them for sharing. However, it is not easy to estimate the sharing dependency of the files on cloud storage. The pattern of sharing will change over time and requires intensive machine learning applications to predict them [4–19].

It also helps the companies to check the utilization of the spaces availed from the cloud and to estimate the future trends in storage requirements.

4.2 Machine Learning in Cloud Resource Provisioning

The elasticity is the main feature of cloud computing which automatically increases and shrinks the resources for the applications based on the workload. The prediction of the workload usually is done through resource indicators such as processor usages, storage usage, and traffic parameters. It is well suited for non-complex applications where the workload patterns are having periodical changes. However, it is very difficult to fix the indicator values, changes, and obtained thresholds.

When applications become complex, the indicators will be at a low level and limited. The changes in workload will be ad hoc, and it requires techniques apart from the resource indicators. In general, all the cloud services will be offered as best of service where no reservation of resources will be done by the vendors for the applications. Based on the demand raised by the applications, the resources are being provisioned. No guarantee is given by the vendor.

At the same time, cloud vendors can also provide advanced reservations for the applications. In this case, the SLA will have guaranteed QoS parameters to provide reservation. Such reservations need to be supported by the machine learning utilities to compute the current level of resource utilization and estimate future trends [20].

Machine learning is being applied in cloud resource provisioning, especially for pro-active management and auto-scaling. Many kinds of research have witnessed the role of machine learning in auto-scaling and the proactive management of clouds. VMware's Distributed Resource Scheduler (DRS) is proposed in [21] and uses an auto-scaling. A middleware called "Haizea" is introduced in [22] and offers reservation facility through leases. The Haizea can be called anytime and anywhere. The metric-based scaling technique called "Amazon Cloud Watch" is proposed in [23] that does auto-scaling on demand. No reservation is supported by the Amazon Cloud Watch. In [24], a system is proposed that reserves the resources initially and later adjusts based on the demands of the applications. In [25], the Aneka system is proposed that collects and releases the resources based on the completion time of requests. The auto-scaling is governed by the thresholds that must meet the requirements of completion of applications. In [26], a system is proposed for auto-scaling that uses past workload statistics to estimate future needs.

In [27], statistical machine learning was employed in single tier applications on the cloud to estimate the system performance parameters. The CPU and bandwidth utilization of VMs for single-tier applications were measured in Amazon EC2 using machine learning [28]. The reinforcement learning approach was used to find the best optimal server configurations in [29]. The same reinforcement learning was used to allocate the resources automatically in [30] for single-tier applications. The work in [31] uses non-linear regression techniques to learn the patterns of the performance of web applications hosted on the cloud, and a trained model is then used to predict future needs. The queuing networks were used in the research for analyzing the behaviors of each tier in multi-tier web applications [32]. The capacity of multi-tier web applications was estimated online using machine learning in [33]. The k-means clustering algorithm was used to model the dynamic workloads of

multi-tier applications. This work uses queuing theory and service rates as base parameters for the clustering. Another work in [34] uses the learning models to do resource provisioning for homogeneous performances. In [35], machine learning is employed to automatically configuring web applications based on CPU utilization, number of requests, and network utilization. A rule-based learning model was used in [36] for identifying sudden changes in requirements.

From these works, it is concluded that workload volume is a key parameter that is being identified differently and used to estimate the future needs of the requirements. The machine learning techniques were good in predicting the workload patterns in the clouds, especially for dynamic provisioning. The machine learning models need to be designed based on the application for which the estimation is carried out [27–37].

The sharing dependency is a vital relation in identifying the workload volumes of storage clouds. This chapter takes NEC personal cloud data and models the sharing dependency using ensemble learning for the auto-provisioning of resources.

4.3 Prediction of Sharing Dependency Using Ensemble Learning

4.3.1 Ensemble Learning

Ensemble learning is a branch of machine learning where multiple model outputs are combined to yield improved performance. Bias and variance are two important factors in machine learning.

The bias is referring to the average difference between actual values and predicted values. The trained model bias is high; it will underperform and miss important details in the models. Hence, we need to have a lower bias. When models have become complex to train, the bias will reduce and it is up to some point. After that, the variance of the model will increase.

The variance is another factor that defines how the prediction is done on the same data different from each other. The high variance will overfit your data and the model will be failed. However, the bias and variance will be the tradeoff as shown in Fig. 4.1. To resolve this, ensemble learning is used.

The bias and variance will play an important role in machine learning. The low bias and low variance model will be good, and it will classify all the data into the respective classes. If the model has low bias and high variance, it will classify the data around the respective classes, not exactly on the classes, and each data will be far away from each other. If the model has high bias and low variance, it will miss the data and will not classify it into respective classes. However, all these data will be near to each other because of the low variance. If the model has high bias and high variance, it will miss all the data and will not classify into the respective classes. All these data will be far away from each other because of the high variance.

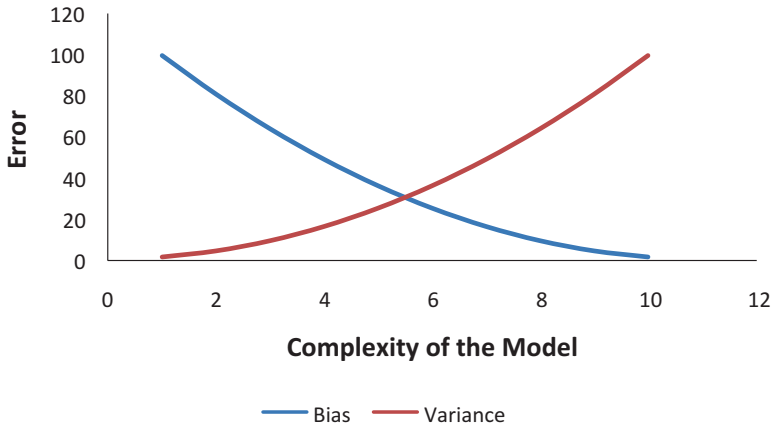

Fig. 4.1 Bias vs. variance in machine learning

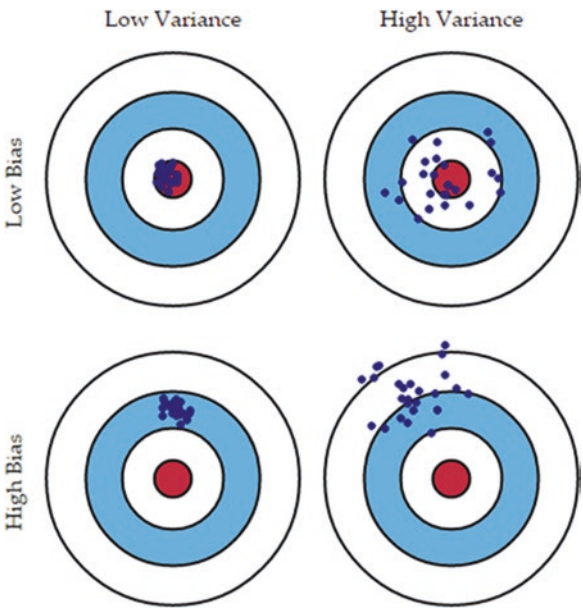

Fig. 4.2 The bulls eye diagram for bias vs. variance in machine learning. Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Hence, the main objective of any machine learning model is to reduce bias and variance to get the best model. It is shown in Fig. 4.2. The red color circle is a class to which the data points (blue colored dots) to be classified.

The ensemble learning can be applied in three different styles. They are bagging, boosting, and stacking. The bagging style will divide your input data sets into small, multiple datasets. The appropriate, different machine learning models will be chosen for every subset of the data and applied independently. The results of all the subsets are aggregated through an average to give the results of the original dataset.

The boosting style will not divide the datasets. It collects all the appropriate models and starts applying them one by one on the dataset. It adjusts the weight values of the model based on the previous results. Thereby, it reduces the bias and makes the model the best fit for the data.

The stacking is a different style where different model outputs are combined. The combined operation targets reducing bias or variance. Thereby, the best model is identified through stacking.

4.3.2 Dataset

The NEC Personal Cloud Data set is used in this chapter [38]. This dataset is an original dataset collected from the vendors with limitations. The limitation includes no location information is collected for privacy. All the data are collected as traces and it contains log information. The data are given by two trace files. The File Trace (CS_FileTraces.txt) contains details about the file, and Sharing Traces (CS_SharingTraces.txt) contains sharing details of the files. The data collected is real data from 7th March 2013 to 9th September 2015, and the total duration of the data is 2 years and 6 months. This data set contains two levels of information. They are storage and sharing interactions.

From the storage level, file traces were collected directly from the provider through SQL Server and Open Stack Swift. The file traces are log files containing file id, user id, file size in Bytes, and Account/Container ID in which the file is located. This information is used to analyze the file-related details at the storage layer. The statistics of the file traces are listed in Table 4.1.

From the sharing interaction, the sharing traces were collected. The sharing traces are log files that contain interaction among users and their sharing file information. The data is collected for 2 years and 6 months duration as mentioned above. This file contains 75,041 interaction details of file sharing. The fields include user 1, user 2, file id, folder ID, and account ID. User 1 is the owner of the file, and it is located in the folder that is linked to that user in the cloud. The user 2 is sharing the files of the user1. The statistics of the sharing interaction are listed in Table 4.2.

Table 4.1 CS_FileTraces – statistics

Total number records	76,554 file records
Total number of unique users	7721
Total number of unique accounts/containers	9215
Total number of unique files	74,723
Total number of unique file formats available in the dataset	418
Total number of unique file size present in the dataset	63,430

Table 4.2 CS_SharingTraces – statistics

Total number records	75,041 sharing interaction records
Total number of unique owners (User 1)	7015
Total number of unique users sharing the files (User 2)	8314

4.3.3 Preprocessing

The given datasets cannot be directly used for machine learning. To prepare the extracted data to be used for machine learning, preprocessing is carried out. All the details given in the file traces are categorical types except file size which is a numerical type. Similarly, the details in the sharing traces are also in categorical types. The preprocessing analyzes the dataset thoroughly. The file traces contain the details about each file whose owner details are given by the user id. The file sizes are given in bytes. The account/container id is the location id where the file is kept in the cloud. The sharing traces contain files whose owners are in user 1 and sharing users are in user 2. However, the sharing traces recorded each interaction separately. These interactions are raw and not labeled or grouped. The sharing interactions are in different styles which include a single file shared by many users, many files shared by a single user, a single file shared by a single user, and many files shared by a single user. To label the interactions, the number of files of each owner that is shared by others is calculated. The distribution of the number of files of each owner is illustrated in Fig. 4.3.

From Fig. 4.3, it is found that few owners do not have any files in their accounts that are being accessed by others. The number of files held by the owner is around 500 for the majority of the owners. Few of them are having a huge number of files. The minimum and the maximum number of files held by the owner are 0 and 9369, respectively.

Next, the file sizes are computed for each owner. The sharing traces contain only the file ID that is being shared. The size of each file is given in file traces. To calculate the file sizes, all the files shared by an owner are taken from sharing traces, whose size is referred from file traces, and the total file sizes are computed by sum. This is taken as a folder size since it contains many files. The folder sizes are computed in Bytes and converted into MegaBytes (MB). The distribution of folder size is as shown in Fig. 4.4. It is clearly shown that few owners do not have any files in their folder and it makes folder size zero. Many owner's folder size is around 5000 MB and a few of them having folder size above 5000 MB. The minimum and maximum folder sizes are 0 MB and 241830.1 MB, respectively.

Next, the frequency of folder access for each owner is computed. It is directly calculated from the sharing traces file by doing self-references of owners in the user1 field. The reason is that sharing traces contain interaction details of each file that is linked to an owner.

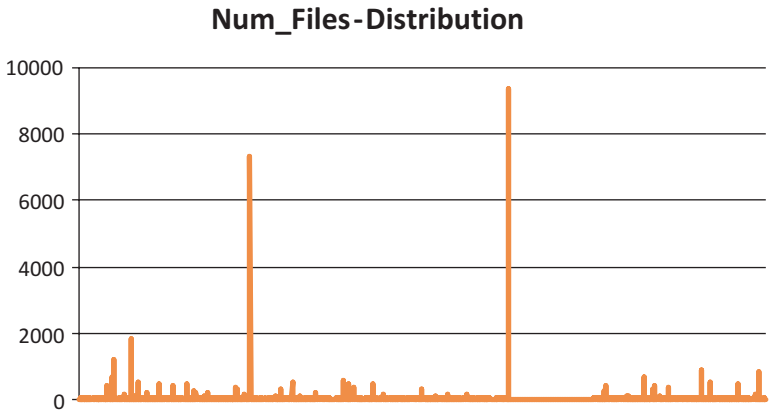

Fig. 4.3 The distribution of the number of files for each owner that is shared by others

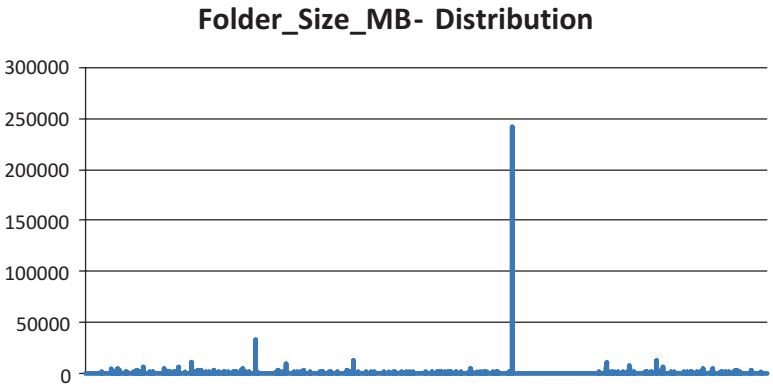

Fig. 4.4 The distribution of the folder size of each owner

The distribution of frequency access to the folder is shown in Fig. 4.5. It is clearly shown that the minimum frequency is zero. It means that the folder is not accessed by anyone. The maximum frequency is 242. It means that the folder is shared by 242 users. Thus the preprocessing extracted four different parameters from the dataset. The user ID is mainly for identification, and it is excluded from machine learning. The other three parameters number of files, folder size, and frequency access are taken for machine learning.

4.3.4 Ensemble Classifiers

The five different ensemble classifiers are used in the experiment. They are bagged tree, boosted tree, subspace discriminant, subspace kNN, and RUSboosted tree.

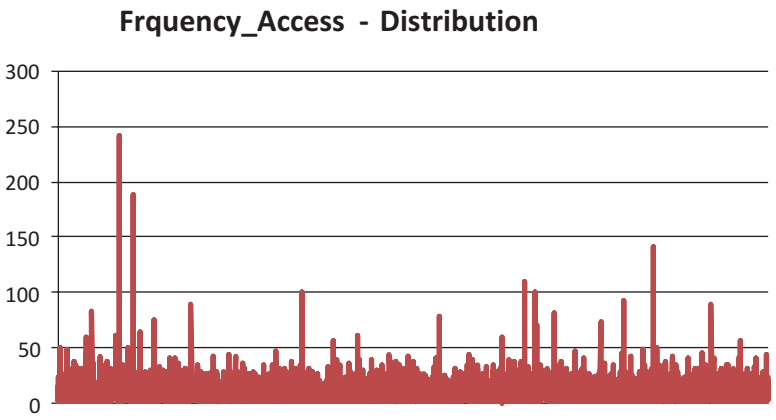

Fig. 4.5 The distribution of the frequency of access to the owner's folder

The bagged tree uses a random forest algorithm. It divides the given input into *n* number of parts using bootstrap sampling. Then, it constructs n number of classification trees and combines them into a single tree to yield the final classification [39–41].

The ADA boosting is used in the boosted tree which is one of the first boosting algorithms that combines multiple weak learner's outputs into single strong learners by using different weights. The ADA boost first divides the input into number splits and then applies a decision tree on them individually. A single split with a decision tree is called decision stumps. It analyzes all the stumps carefully and assigns higher weightage to the stumps which is difficult to classify and low weightage to the stumps which can classify well [42].

The subspace discrimination uses discriminant learners in subspace. The learners use Gaussian mixture models. The subspaces are generated using random subspace methods. The random subspace method uses three parameters. The first parameter is the actual dimension (D1) of the data (it means the number of columns in the given data) and next is the dimensions of the data (D2) to be used in the samples of each learner in the ensemble. The value of D2 is to be calculated statistically. The last parameter is the number of learners (N) used in the ensemble.

The random subspace method identifies the random subset of D2 variables from the D1 possible variables. Then, the methods iteratively train the week learners of discriminant analysis using these D2 variables, until there are N weak learners. The final prediction is carried out from the highest average of weak learners. Thus, the subspace discrimination works. Similarly, the subspace kNN works in subspace with nearest neighbors' leaners rather than discriminant analysis [42–47].

The RUSboosted tree uses ADA boosting algorithm in under-sampling space. This method is well suited for imbalanced data where few classes will have only a few members in the training data compared to other classes. The under-sampling is a key concept in this method where the number of members for each class in the training data is computed first and selects the minimum value. The chosen minimum

value will be the sample size for each class to which the undersampling is done for the classes whose members are above the minimum. Finally, the boosting procedure will be done based on ADA Boost [42–47].

4.3.5 Data Cleaning

As in Sects. 3.3.2 and 3.3.3, the NEC data set is a log file that contains two sets of information for sharing dependency in the personal clouds. From this, we have extracted three vital pieces of information, namely, the number of files, folder size in MB, and frequency of access of the files for each owner. This is the data to be used in machine learning. This data is collected for 7015 owners in the NEC personal cloud. Thereby, the data set contains 7015 records and the dimension is three. The number of files and folder size is chosen as predictors, and the frequency of access is taken as a response.

First, the outliers analysis is carried out for from the variables through the box plot. By reviewing the box plot of the number of files as shown in Fig. 4.6, it is clear that the majority of the values are below 1000 and few values are above 1000. The number of the file contains zero values for 897 records. It means that 897 owners have accounts with no files in them. The median of the number of files is six. By analyzing the box plot for the number of files, it is found that the data is not uniformly distributed.

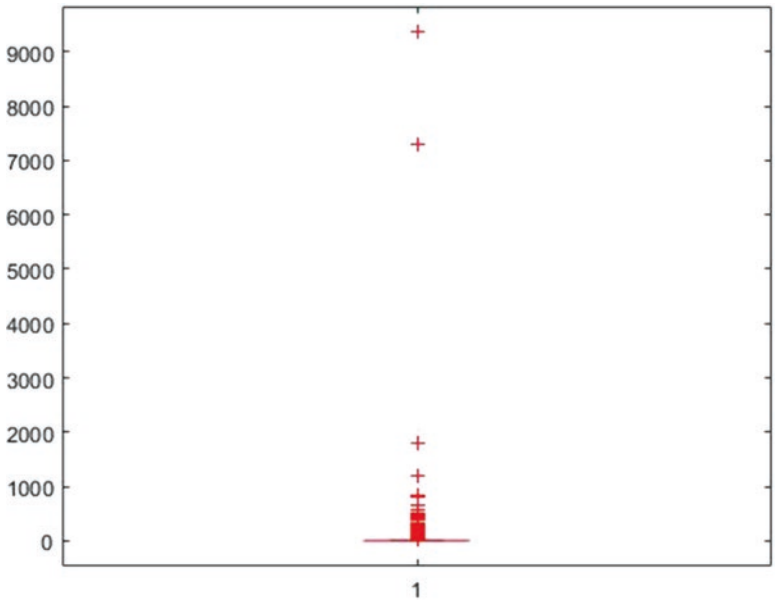

Fig. 4.6 The box plot for number of files

The box plot for the folder size is shown in Fig. 4.7. The minimum folder size is 0 MB and the maximum is 241,830 MB. The accounts which are not having any files will have a 0 MB folder size. The median value is 166 MB. The folder size is also not uniformly distributed. The response variable frequency of access is also reviewed by its box plot as shown in Fig. 4.8.

The frequency of access is having a minimum of 0 and a maximum of 242. The median is 9. and the majority of the values are within 100 and few values are above 100. It is also observed that accounts not having any files are accessed many times. The frequency of access to 897 records (which are not having any files, empty folders) is exemplified in Fig. 4.9. This information will not provide any scope in machine learning for the identification of shared dependency. The main aim of the work is to predict sharing dependency for the dynamic provisioning of cloud resources. The empty folder's access will not require any provisioning of resources. Such folders would have been accessed wrongly or the requested file is no longer available. Hence, these 897 records are filtered out from the experiment. In addition to this, the folders have few empty files whose sizes are in few kilobytes (KB). However, these files are accessed many times, and such information will be useful in learning. Hence, those records are retained. Thereby, the dataset contains 7015 records initially reduced to 6118 records after cleaning.

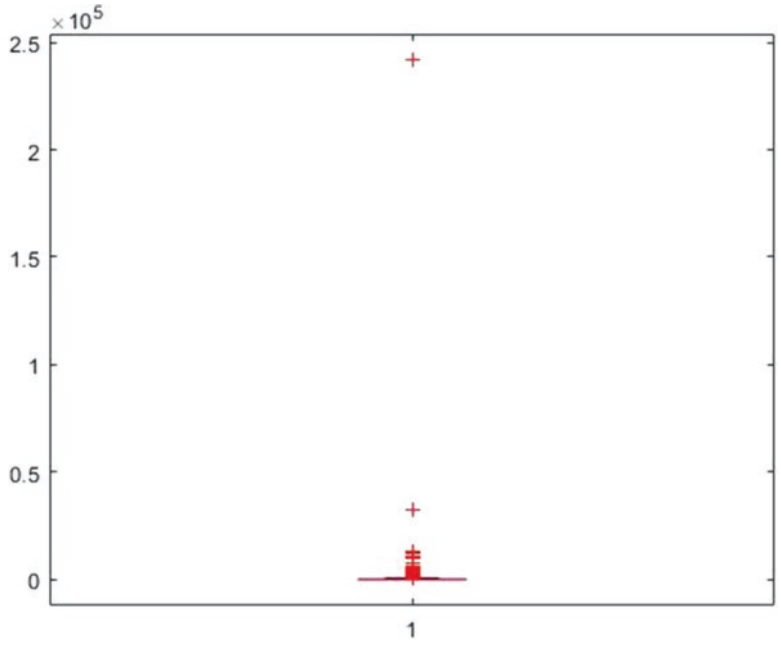

Fig. 4.7 The box plot for folder size

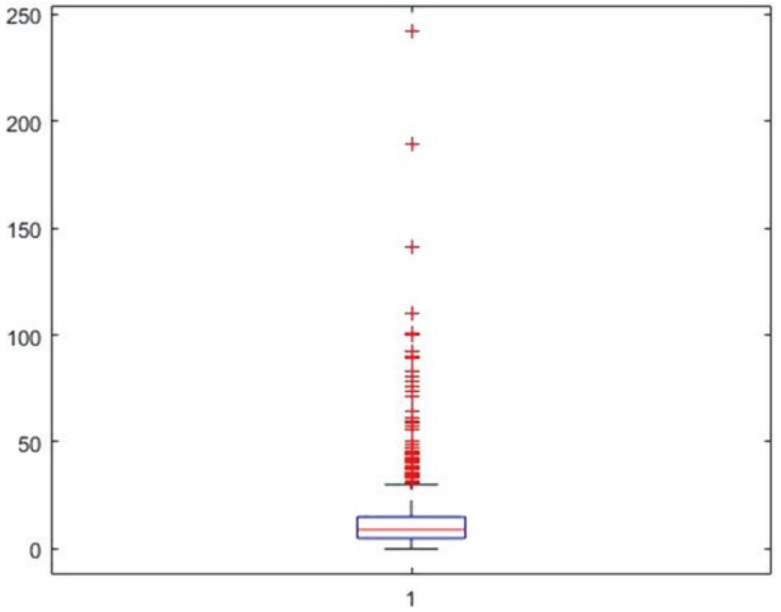

Fig. 4.8 The box plot for frequency of access

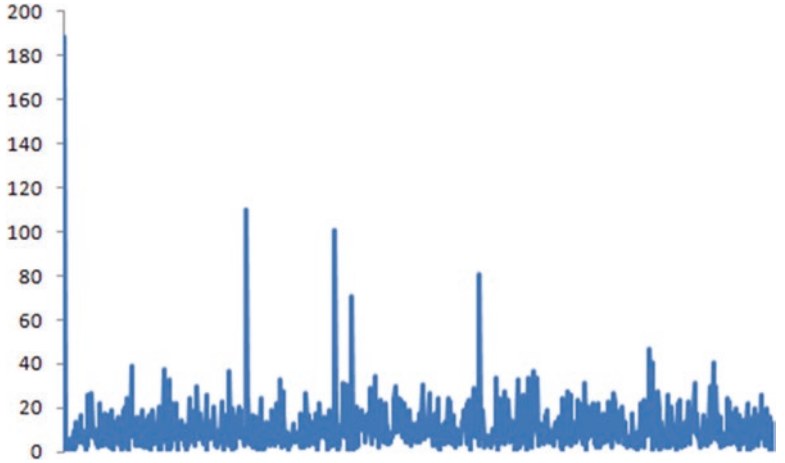

Fig. 4.9 The frequency of access to empty folders

4.3.6 Model Training and Analysis

After the cleaning, the experiment is set up with the number of files, and folder size is as predictors and frequency of access as a response. The experiment is carried out on Matlab R2016b using the classification leaners app. The response variable

contains 65 classes for which the predictors are trained in the model. The scatter plot of predictors is shown in Fig. 4.10. From the scatter plot, it is clear that the predictors are not linearly related and cannot be trained using any linear models. The relationship of predictors is non-linear and it requires non-linear models. The nature of the tasks in the experiment is to predict the frequency of access from a given number of files and folder size. The decision tree and clustering models are more suitable for such situations. Thereby, the chapter uses ensemble learning for the training.

The predictors are fed to the classification learner app with the default configuration of all the ensemble learners. The results are summarized in Table 4.3. The boosted trees yield the highest accuracy of 7.1%, and all other models are providing an accuracy below 7.1%. The ROC curves of all the models are shown in Fig. 4.10. All the models provide an AUC value of around 0.5, and around 50% of the data is properly mapped to positive and negative classes as shown in Fig. 4.11.

It is also understood that the response variable has 65 classes of the frequency of access not well mapped by the number of files and folder sizes. To fine-tune the results, we have created a new class variable based on the number of files, folder size, and frequency of access. Here, these three variables are taken as the predictors. The response variable is prepared by using quartiles of each variable. Each variable has four quartiles, and all these four quartiles of each variable are combined to quartile values of other variables. Thereby, 64 quartile combinations are created and the same is modeled as response classes for the data. Then, for every sample, the

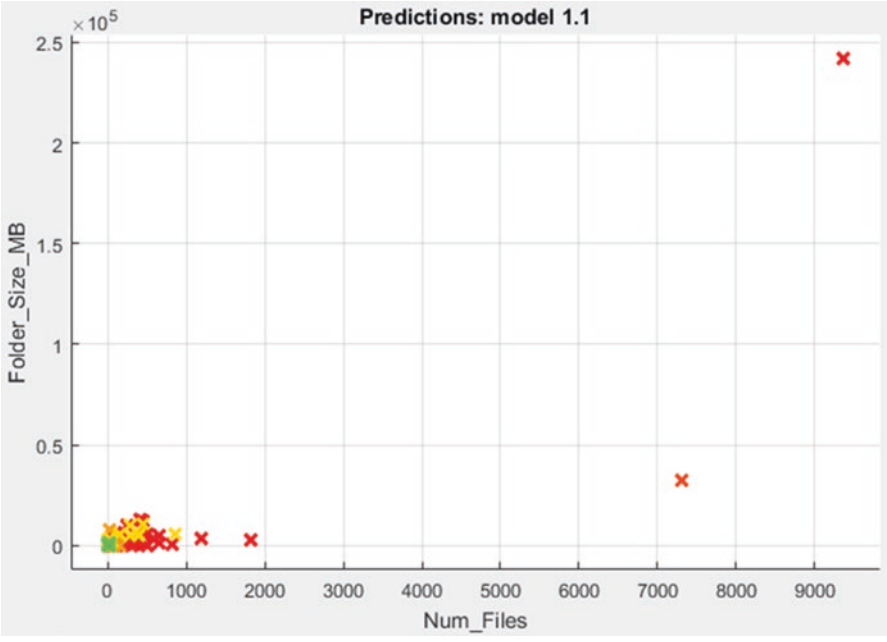

Fig. 4.10 The scatter plot between Num_Files and Folder_Size_MB

Table 4.3 Ensemble learners – results (number of predictors is two)

Ensemble learner	Accuracy (%)
Model 1.1 boosted trees	7.1
Model 1.2 bagged trees	5.1
Model 1.3 subspace discriminant	6.9
Model 1.4 subspace kNN	5.9
Model 1.5 RUSboosted trees	4.0

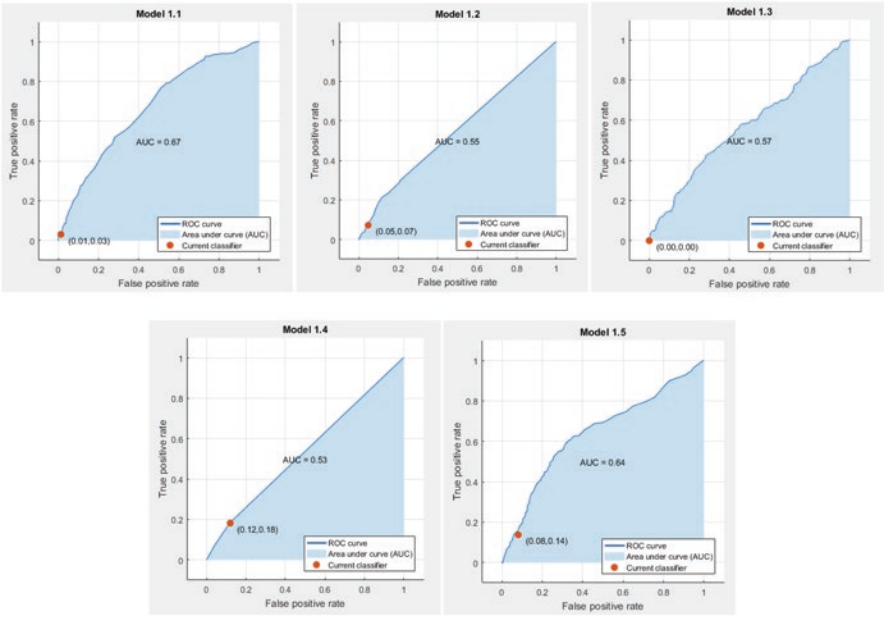

Fig. 4.11 ROC curve for the ensemble learners – number of predictors is two

corresponding quartile in which the variable is present and quartile combination of all three variables are computed.

Table 4.4 shows the quartile values of all the predictors used in the experiment. The response class is computed as follows. For example, if the sample contains values of Num_Files as 12, Folder_Size_MB as 684.58, and Frequency _Access as 18, then the quartile for Num_Files will be 4, Folder_Size_Mb will be 4, and Frequency_Access as 4. The response class for this sample will be 444. Thus, the response class for each sample is computed. Now, the model is trained using these three variables as a predictor and combined quartile values of variables as the response variable.

Table 4.5 shows the improvement in accuracy compared to the results in Table 4.4. The bagged trees provide the highest accuracy of 99.8%, and the boosted tree yields the second highest accuracy of 88.1%. All other models provide an

Table 4.4 Quartile values of predictors

Parameters	Num_Files	Folder_Size_MB	Frequency_Access
Min	1	0	1
Q1	4	91.8675	5
Q2	6	203.725	9
Q3	8	362.2425	15
Q4	9369	241830.1	242
Max	9369	241830.1	242

Table 4.5 Ensemble learners – results (number of predictors is three)

Ensemble learner	Accuracy (%)
Model 1.1 boosted trees	88.1
Model 1.2 bagged trees	99.8
Model 1.3 subspace discriminant	14.8
Model 1.4 subspace kNN	38.5
Model 1.5 RUSboosted trees	39.1

accuracy of below 50% and improved the accuracy compared to the results in Table 4.4. The second approach improved the overall accuracy of ensemble classifiers by 50% at an average. The individual model wise improvements are 81% in boosted trees, 94.70% in bagged tress, 7.90% in subspace discriminant, 32.60% in subspace kNN, and 35.10% in RUSboosted trees.

Figure 4.12 shows the performance of each model through ROC curves, and all the models can improve the AUC values compared to the previous results in Fig. 4.11. Now, the models are providing AUC values between 0.9 and 1.

4.3.7 Comparative Results

We have experimented with two different methods. First, we have taken a dataset with two predictors (Num_Files and Folder_Size_MB) and Frequency_Access as Response. The training results gave a maximum accuracy of 7.1% and an AUC of 0.67. This is achieved by the boosted tree. Second, we have taken all three variables as predictors (Num_Files, Folder_Size_MB, and Frequency_Access) and the newly computed quartile combination as a response. The training results for this method yield a maximum accuracy of 99.8% and an AUC of 1. Comparatively, the second method was found best for the NEC personal cloud dataset. This method can predict the demands of the quartile combinations using Num_Files and Folder_Size_MB and Frequency_Access. Method 2 improved the overall accuracy of all the models by 50% at an average and AUC by 45% at an average. The bagged tree model was found as the best model for the NEC cloud dataset.

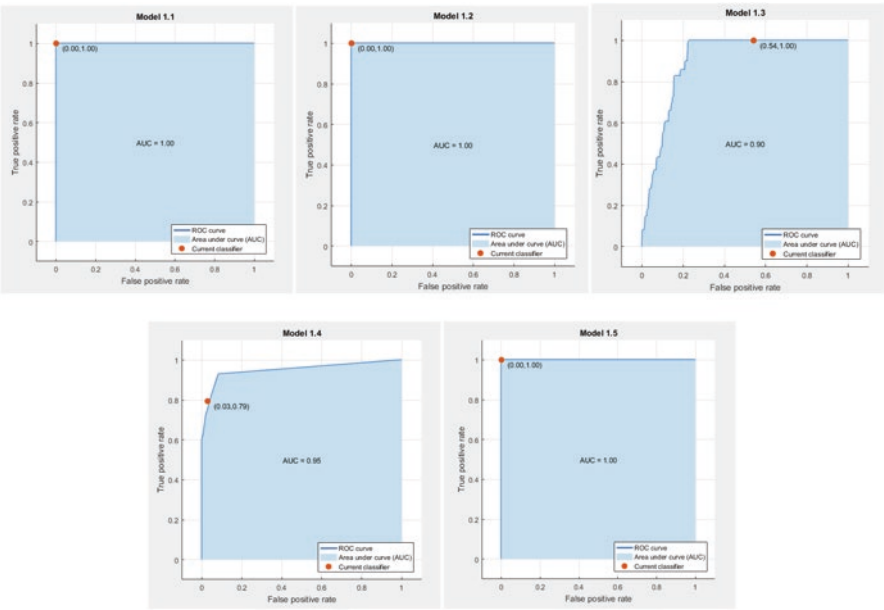

Fig. 4.12 ROC curve for the ensemble learners – number of predictors is three

4.4 Conclusions

This chapter presents ensemble classifier analysis to the NEC personal cloud dataset that contains two files File Traces and Sharing Traces for 2 years and 6 months from 7th March 2013 to 9th September 2015. From the dataset, we have extracted three 4 parameters, Owner ID, Number of Files, Folder Size in MB, and Frequency of Access. The owner ID is omitted, and the other three parameters are taken for learning. The ensemble learning contains five models boosted trees, bagged trees, subspace discriminant, subspace kNN, and RUSboosted trees. The training is done by two methods. The first method attempted to map the relationship between the number of files and folder size to the frequency of access. The second method attempted to find the relationship between all these three parameters to newly computed quartile combinations. The experimental results show that method 2 is best for the NEC data set and the Bagged Tree model can excel in the prediction that provides an accuracy of 99.8% and AUC of 1. Method 2 improves the accuracy of all the models by 50% and AUC by 45% compared to method 1. Thus, the bagged tree model with three predictors outperforms all other models of ensemble classifiers, and it is found as the best model for NCE personal cloud dataset in predicting sharing dependency. It is concluded that the bagged tree model is the best in estimating the sharing dependency of personal clouds in multi-cloud environments.

References

1. Vecchiola, C., Calheiros, R. N., Karunamoorthya, D., & Buyyaa, R. (2012). Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka. *Elsevier Journal of Future Generation Computer Systems, 28*, 58–65. https://doi.org/10.1016/j.future.2011.05.008
2. Islam, S., Keunga, J., Lee, K., & Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Elsevier Journal of Future Generation Computer Systems, 28*, 155–162. https://doi.org/10.1016/j.future.2011.05.027
3. Guruprasad, B. (2014). Resource provisioning techniques in cloud computing environment: A survey. *International Journal of Research in Computer and Communication Technology, 3*, 395–401.
4. Jeyarani, R., Nagaveni, N., & Vasanth Ramc, R. (2012). Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. *Elsevier Journal of Future Generation Computer Systems, 28*, 811–821. https://doi.org/10.1016/j.future.2011.06.002
5. Kousiouris, G., Menychtasa, A., Kyriazis, D., Gogouvitis, S., & Varvarigou, T. (2014). Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in cloud platforms. *Elsevier Journal of Future Generation Computer Systems, 32*, 27–40. https://doi.org/10.1016/j.future.2012.05.009
6. Li, C., & Li, L. Y. (2012). Optimal resource provisioning for cloud computing environment. *SpringerLink Journal of Supercomputing, 62*, 989–1022. https://doi.org/10.1007/s11227-012-0775-9
7. Javadi, B., Thulasiraman, P., & Buyya, R. (2013). Enhancing performance of failure-prone clusters by adaptive provisioning of cloud resources. *SpringerLink Journal of Supercomputing, 63*, 467–489. https://doi.org/10.1007/s11227-012-0826-2
8. Halder, K, Bellur, U, & Kulkarni, P (2012). Risk aware provisioning and resource aggregation based consolidation of virtual machines. IEEE fifth international conference on cloud computing (pp. 598–605).
9. Wang, C, Chen, J, Zhou, B. B., & Zomaya, A. Y. (2012) Just satisfactory resource provisioning for parallel applications in the cloud. IEEE eighth world congress on services (pp. 285–292). https://doi.org/10.1109/SERVICES.2012.38.
10. Ramachandran, L., Narendra, N. C., & Ponnalagu, K. (2012). Dynamic provisioning in multi-tenant service clouds. *SpringerLink Journal of Service-Oriented Computing and Applications, 6*, 283–302. https://doi.org/10.1007/s11761-012-0116-0
11. Zaman, S, & Grosu, D. (2011). Combinatorial auction-based mechanisms for VM provisioning and allocation in clouds**.** IEEE third international conference on cloud computing technology and science. https://doi.org/10.1109/CloudCom.2011.24.
12. Zhu, Q., & Agrawal, G. (2012). Resource provisioning with budget constraints for adaptive applications in cloud environments. *IEEE Transactions on Services Computing, 5*, 497–511. https://doi.org/10.1109/TSC.2011.61
13. Verma, A., Cherkasova, L., & Campbell, R. H. (2011). Resource provisioning framework for MapReduce jobs with performance goals. *SpringerLink Lecture Notes in Computer Science, 7049*, 165–186. https://doi.org/10.1007/978-3-642-25821-3_9
14. He, S., Guo, L., Guo, Y., Wu, C, Ghanem, M, & Han, R. (2012). Elastic application container: A lightweight approach for cloud resource provisioning. IEEE international conference on advanced information networking and applications. https://doi.org/10.1109/AINA.2012.74.
15. Nelson, V., & Uma, V. (2012). Semantic-based resource provisioning and scheduling in inter-cloud environment. IEEE xplore recent trends in information technology (pp. 250–254). https://doi.org/10.1109/ICRTIT.2012.6206823.
16. Pawar, C. S., & Wagh, R. B. (2013). Priority based dynamic resource allocation in cloud computing. International conference on intelligent systems and signal processing. https://doi.org/10.1109/ISSP.2013.6526925.

17. Zhanga, T, Dua, Z., Chen, Y., Xiang J. C, & Wang, X (2011) Typical virtual appliances: An optimized mechanism for virtual appliances provisioning and management. *Elsevier Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2010.11.925.
18. Javadi, B., Abawajy, J., & Buyya, R. (2012). Failure-aware resource provisioning for hybrid cloud infrastructure. *Elsevier Journal of Parallel and Distributed Computing, 72*, 1318–1331. https://doi.org/10.1016/j.jpdc.2012.06.012
19. Wang, Y., Nakao, A., Vasilakos, A. V., & Ma, J. (2011). On the effectiveness of service differentiation based resource provision incentive mechanisms in a dynamic and autonomous P2P network. *Elsevier Journal of Computer Networks, 55*, 3811–3831. https://doi.org/10.1016/j.comnet.2011.07.011
20. Biswas, S. Majumdar, B. Nandy, A. & El-Haraki (2014) Automatic resource provisioning: A machine learning-based proactive approach. IEEE 6th international conference on cloud computing technology and science (pp. 168–173). https://doi.org/10.1109/CloudCom.2014.147.
21. Gulati, A., Shanmuganathan, G., Holler, A., & Ahmad, I. (2011). Cloud-scale resource management: challenges and techniques. Proceedings of the 3rd USENIX conference on hot topics in cloud computing. https://dl.acm.org/doi/10.5555/2170444.2170447.
22. Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. *IEEE Computer Society, 13*, 14–22. https://doi.org/10.1109/MIC.2009.119
23. Amazon CloudWatch. http://aws.amazon.com/cloudwatch.
24. Wang, W., Niu, D., Li, B., & Liang, B. (2013) Dynamic cloud resource reservation via cloud brokerage. *Distributed Computing Systems*. https://doi.org/10.1109/ICDCS.2013.20.
25. Buyya, R., Garg, S. K., & Calheiros, R. N. (2011). SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. Proceedings of the 2011 international conference on cloud and service computing. https://doi.org/10.1109/CSC.2011.6138522.
26. Moore, L. R., Bean, K., & Ellahi, T. (2013). A coordinated reactive and predictive approach to cloud elasticity. Fourth international conference on cloud computing, GRIDs, and virtualization.
27. Bodik, P. et al., (2009). Statistical machine learning makes automatic control practical for internet data centers. In Proc. HotCloud (pp. 1–5).
28. Liu, H., & Wee, S. (2009) Web server farm in the cloud: Performance evaluation and dynamic architecture. In Proceeding of 1st International Conference on Cloud Computing (pp. 369–380).
29. Bu, X., Rao, J., & Xu, C.-Z. (2009). A reinforcement learning approach to online web systems auto-configuration. In Proceeding 29th IEEE ICDCS.
30. Tesauro, G., Jong, N. K., Das, R., & Bennani, M. N. (2006). A hybrid reinforcement learning approach to autonomic resource allocation. In Proceeding of IEEE international conference on autonomic computing
31. Bodik, P. et al., (2009). Automatic exploration of datacenter performance regimes. In Proceeding of 1st workshop ACDC (pp. 1–6). https://doi.org/10.1145/1555271.1555273
32. Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., & Tantawi, A. (2005). An analytical model for multi-tier internet services and its applications. In Proceeding of ACM SIGMETRICS international conference on measurement and modeling of computer systems SIGMETRICS.
33. Rao, J., & Xu, C.-Z. (2010). Online capacity identification of multitier websites using hardware performance counters. *IEEE Transactions on Parallel and Distributed Systems, 22*, 426–438.
34. Dejun, J., Pierre, G., & Chi, C.-H. (2011). Resource provisioning of web applications in heterogeneous clouds. In Proceeding of 2nd USENIX Conf. Web Appl. Develop.
35. Villela, D., Pradhan, P., & Rubenstein, D. (2007). Provisioning servers in the application tier for e-commerce systems. *ACM Transactions on Internet Technology*. https://doi.org/10.1145/1189740.1189747.
36. Sharma, A., et al. (2008). Automatic request categorization in internet services. *SIGMETRICS Performance Evaluation Review, 36*, 16–25.

37. Bodik, P., Fox, O., Franklin, M. J., Jordan, M. I., & Patterson, D. A. (2010) Characterizing, modeling, and generating workload spikes for stateful services. In Proceedings of the 1st ACM SoCC (pp. 241–252).
38. Gracia-Tinedo, R., García-López, P., Gómez, A., & Illana, A. (2016) Understanding data sharing in private personal clouds. 2016 IEEE 9th international conference on cloud computing. https://doi.org/10.1109/CLOUD.2016.0059.
39. Breiman, L. (1996). Bagging predictors. *Machine Learning, 26*, 123–140.
40. Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.
41. Breiman, L.. https://www.stat.berkeley.edu/~breiman/RandomForests/
42. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*, 119–139.
43. Freund, Y. (2009) A more robust boosting algorithm. arXiv:0905.2138v1.
44. Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*, 1189–1232.
45. Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics, 28*, 337–407.
46. Schapire, R. E., et al. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics, 26*, 1651–1686.
47. Seiffert, C., Khoshgoftaar, T., Hulse, J., & Napolitano, A. (2008) RUSBoost: Improving classification performance when training data is skewed. 19th international conference on pattern recognition (pp. 1–4).