StackInsights: Cognitive Learning for Hybrid Cloud Readiness

Mu Qiao, Luis Bathen, Simon-Pierre Génot, Sunhwan Lee, Ramani Routray
IBM Almaden Research Center, San Jose, California, CA
Email: {mqiao,bathen,sgenot,shlee,routrayr}@us.ibm.com

***Abstract*—Hybrid cloud is an integrated cloud computing environment utilizing a mix of public cloud, private cloud, and on-premise IT infrastructures. Workload awareness, defined as a detailed full range understanding of each individual workload, is essential in implementing the hybrid cloud. While it is critical to perform an accurate analysis to determine which workloads are appropriate for on-premise deployment versus which workloads can be migrated to a cloud off-premise, the assessment is mainly performed by rule or policy based approaches. In this paper, we introduce StackInsights, a novel cognitive system to automatically analyze and predict the cloud readiness of workloads for an enterprise. Our system harnesses the critical metrics across the entire stack: (1) infrastructure metrics, (2) data relevance metrics, and (3) application taxonomy, to identify workloads that have characteristics of (a) low sensitivity with respect to business security, criticality and compliance, and (b) low response time requirements and access patterns. Since the capture of the data relevance metrics involves an intrusive and in-depth scanning of the content of storage objects, a machine learning model is applied to perform the business relevance classification by learning from the meta level metrics harnessed across stack. In contrast to traditional methods, StackInsights significantly reduces the total time for hybrid cloud readiness assessment by orders of magnitude.**

***Keywords*-hybrid cloud, cognitive learning, sensitivity, infrastructure, classification**

I. Introduction

Hybrid cloud, which utilizes a mix of public cloud, private cloud, and on-premise, has become the dominant cloud deployment architecture for enterprises. Public cloud offers a multi-tenant environment, where physical resources, such as computing, storage and network devices, are shared and accessible over a public network, whereas private cloud is operated solely for a single organization with dedicated resources. Hybrid cloud inherits the advantages of these two cloud models and allows workloads to move between them according to the change of business needs and cost, therefore resulting in greater deployment flexibility. The global hybrid cloud market is estimated to grow from USD 33.28 Billion in 2016 to USD 91.74 Billion in 2021 [1].

Business sensitivity is one of the main factors that enterprises consider when deciding which cloud model to deploy. For example, an enterprise can deploy public clouds for test and development workloads, where security and compliance are not an issue. However, it is hard to meet PCI (payment card industry) or SOX (Sarbanes-Oxley) compliance in public clouds due to the nature of multi-tenancy. On the other hand, because private clouds are dedicated to a single organization, the architecture can be designed to assure high level security and stringent compliance, such as HIPAA (health insurance portability and accountability act). Therefore, private clouds are usually deployed for business sensitive and critical workloads. Infrastructure is another important factor to consider when choosing between public and private clouds. Since private cloud is a single-tenant environment where resources can be specified and highly customized, it is ideal to host data which are frequently accessed and require fast response times. For example, high-end storage system can be used in private cloud to deliver IOPS (input/output operations per second) with a guaranteed response time.

Moreover, business sensitivity and infrastructure are traditionally considered in two separate schools of work. However, not all data is created equal, neither is the infrastructure. In this paper, we introduce StackInsights, a novel cognitive learning system to automatically analyze and predict the cloud readiness of workloads for an enterprise by considering both business sensitivity and infrastructure. StackInsights classifies the entire data into several subspaces, as shown in Figure 1, where the X-axis indicates the infrastructure heat map (e.g., storage access intensity) and the Y-axis represents the business sensitivity. A threshold on the X-axis is set to determine if the data is "*cold*" or "*hot*" with respect to infrastructure related performance metrics, and on the Y-axis, the data is classified into three categories: "*sensitive*", "*non-sensitive*", or "*non-classifiable*". Formally, we define sensitive data as the data owned by the enterprise, which if lost or compromised, bares financial, integrity, and compliance damage. There are many different forms of sensitive data, such as sensitive personal information (SPI), personal health information (PHI), confidential business information, client data, intellectual property, and other domain-specific sensitive information. The category of "*non-classifiable*" includes structured data, such as databases, the sensitivity of which can be analyzed using domain knowledge. For example, the databases storing employment information in the HR department should be highly sensitive. All the data which are cold and non-sensitive can be migrated to public clouds while the rest should reside in private clouds. The thresholds on the X-axis and Y-axis can also be adjusted by users. The areas of the subspaces indicate the size of data migrating to different clouds, thereby, serving as a cloud sizing tool. The hotness of data on the X-axis can be obtained by measuring infrastructure performance metrics. The key issue therefore lies in how to determine the business sensitivity of data on the Y-axis.

To the best of our knowledge, StackInsights is the first cognitive system that uses machine learning to understand data sensitivity based on metadata, as it correlates application, data, and infrastructure metrics for hybrid cloud readiness assessment. In contrast to traditional methods which require content scanning for sensitivity analysis, StackInsights significantly reduces the total running time through predictive analytics. It advises users what data are appropriate to be stored on premises, or to be migrated to the cloud, and the specific cloud deployment model, by integrating both data sensitivity and hotness in terms of infrastructure performance.

The rest of the paper is organized as follows. We describe our motivation and contribution in Section II. The relevant work is reviewed in Section III. In Section IV, we introduce the framework of StackInsights as well as the cognitive learning components. Section V is on the experiments and results. Finally, we conclude in Section VI.

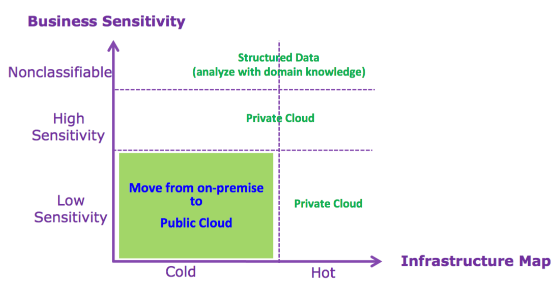


Figure 1: Hybrid cloud migration overview

II. Motivation and Contribution

The classification of data sensitivity belongs to the general domain of data classification, which allows organizations to categorize data by business relevance and sensitivity in order to maintain the confidentiality and integrity of their data. Data classification is a very costly activity. In large organizations, data is usually stored and secured by many repositories or systems in different geographical locations, which may have different data privacy and regulatory compliances. Various security access approvals have to be obtained in order to get access to data. In addition, traditional sensitivity assessment approaches require an intrusive and in-depth content scanning of the objects, which is not scalable in this big data era, where numerous structured and unstructured data are generated in real-time. To solve this issue, we develop a machine learning model in StackInsights, which can perform a business sensitivity classification by learning from file metadata, which is much easier and cost-efficient to collect. By using meta data, we can already obtain a sensitivity prediction model with high accuracy. Therefore, we do not have to perform a detailed content analysis on all the files. Instead, intensive content analysis only needs to be conducted on the predicted non-sensitive files for further screening. Our model-based approach significantly reduces the total sensitivity assessment time.

When migrating workloads among private, public, and hybrid clouds, one of the biggest challenges is the storage layer. An enterprise's infrastructure might consist of a mixture of file, block, and object storage, which have different properties and offer their own advantages. Enterprises big or small tend to manage large and heterogeneous environments. For example, one of our IT environments supports around 100 business accounts, spread over several geographical locations, amassing a total of 200 PB of block storages alone. Similarly, our file storage fabric is also massive, where file shares (mapped to volumes/q-trees) may be in the TB, or even PB scale.

Given such a large storage infrastructure with a number of volumes, we need to determine their cloud migration priority, i.e., which volumes should be migrated first. Data sensitivity is one of the most important factors in cloud migration. Volumes of low sensitivity can be sanitized first and then migrated to the public cloud. From the point of view of cloud migration service admins, it is not critical to know the exact sensitivity of the volumes, but rather the sensitivity "level" of the volumes, so that the migration priority can be assigned. The traditional sensitivity assessment approaches, which require content scanning, are very expensive. It is impractical and not necessary to perform a full content scanning on all the volumes in order to obtain the priority. Machine learning can help predict the sensitivity of files based on the easily collected file meta data, and then obtain the migration priority within a much shorter time.

It is expensive to determine the business sensitivity of each storage volume. We therefore develop a clustering component in StackInsights, which identifies groups of volumes that share similar characteristics. Specifically, the volumes are clustered based on their meta level information, which are obtained by aggregating the file metadata at the volume level. The sensitivity of a representative volume in each cluster is used as the representative sensitivity of all the volumes in the same cluster. To obtain the sensitivity of a representative volume, we apply a machine learning model to predict the sensitivity of each single file on that volume. The sensitivity of a volume is defined as the number of sensitive files divided by the total number the files. Similarly, we also obtain the IOPS of each volume and compute its IO density, which is defined as IOPS per GB. All the volumes which have both low business sensitivity and low IO density can be the candidates to be migrated to public clouds while the other volumes should remain on premise or be migrated to private clouds.

III. RELATED WORK

In the marketplace of enterprise softwares, there are tools developed for data classification in regard of data governance or life cycle management. For example, [2] [3] provide data classification services for managing and retaining various types of data such as emails and other unstructured data through pre-determined rules or dictionary searches. Data privacy and security have become the most pressing concerns for organizations. To embrace the General Data Protection Regulation (GDPR) by European Union, enterprises are making great efforts in addressing key data protection requirements as well as automating the compliance process. For example, IBM Security Guardium solutions [14] help clients secure their sensitive data across a full range of environments. Data classification, as the first step to security, has become extremely important. Only after we understand which data are sensitive through classification, we can then better protect the data. On the other hand, [13] assesses the cloud migration readiness by providing a questionnaire to the owner of the infrastructure. Many of existing tools lack of cognitive capability, and even if there is, the data preparation step requires the scanning of file content, which is not scalable or only supports certain file types.

Besides the rule-based approaches to classify data, there are also attempts leveraging a predictive model. Model-based approaches are much more systematic and scalable because there is no need to generate a rule to classify files manually. The proof-of-concept data classification system that crawls all files in order to analyze data sensitivity was studied in [4]. A new nearest neighbor algorithm was proposed in [5] to determine the sensitivity of files. [8] proposed to use the decision tree classifier for finding associations between a file's properties and metadata. [6] introduced a general-purpose method to detect sensitive information from textual documents using the information theory. The application to data loss prevention by using automatic text classification algorithms for classifying the sensitivity of business documents was introduced in [7]. However, most of the existing works require an exhaustive process to crawl the contents of data, which is impractical in many applications due to privacy, governance, or regulation.

There are preliminary works in the field of hybrid cloud migrations whose components include a data classification method. The tool of migrating enterprise services into hybrid cloud-based deployment was introduced in [9]. The complexity of enterprise applications is highlighted and the model, accommodating the complexity and evaluating the benefits of a hybrid cloud migration, is developed. Though the work sheds insight on the security of data, the tool does not assess the sensitivity of applications at file level. Rule-based decision support tool is deployed in [10] as a modeling tool to compare various infrastructures for IT architects. However, in many practical cases, IT architects have no visibility into the content of data. Therefore, it is not straightforward to model their applications, data, and infrastructure requirements without understanding the nature of data such as sensitivity. In addition, [11] developed a framework to automate the migration of web server to the cloud.

As observed in previous works, data classification and hybrid cloud migration are explored separately although they are tightly related. In contrast, our proposed framework covers the whole process including classifying data, assessing the readiness of cloud migration, and finally a decision support for hybrid cloud migration. Furthermore, we develop an efficient and scalable method to determine the cloud readiness by considering data sensitivity and infrastructure performance through a cognitive learning process.

IV. STACKINSIGHTS FRAMEWORK

We show the high-level framework of StackInsights in Figure 2. In order to gain insights into an existing IT environment, we scan various layers across the entire stack: (1) the application layer, (2) the data layer, and (3) the infrastructure layer. The application layer tells us the types of the running workloads, what components they depend on, and the specific requirements. The data layer provides file metadata as well as content. Finally, the infrastructure layer provides performance metrics, such as, how often the data are accessed, and where they are stored.

Workload Scan: This can be done through IBM's Tivoli Application Dependency Discovery Manager (TADDM) [12], which provides an automated discovery and application mappings. This step is important as we need to understand the running of workloads or applications before starting the scan. For example, the file content scanning can be invasive to latency-sensitive workloads and interfere with their running.

Infrastructure Scan: Our framework is composed of a set pluggable modules that can be adapted to scan different infrastructures. This is important as infrastructures tend to be heterogeneous in nature. For example, a storage infrastructure may have a mixture of different storages, management technologies developed by multiple providers, as well as different ways of accessing data (e.g., via block, file, or object stores). If the workload scan tells us that the storage layer consists mostly of storage filers, we can assume that at this layer, most data is reached through protocols such as Network File System (NFS), and Common Internet File System (CIFS), as well as manufacturer-specific APIs such as NetApp's ONTAP [15]. Once the infrastructure scan is completed, we can then build a location map of all the data and identify the storage volumes or file shares that are most critical for scanning. For the rest of the paper, we will use volume and file share interchangeably as NetApp filers have the notion of q-trees/volumes as mapping points for file shares, which are exposed to users through protocols such as

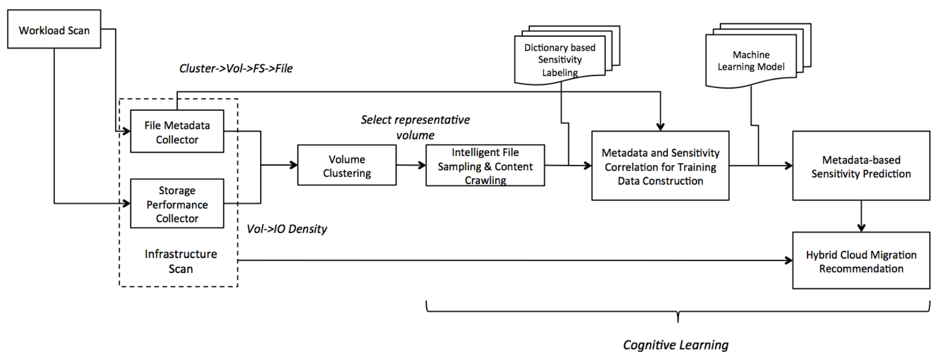


Figure 2: StackInsights framework

CIFS. The root q-tree/volume for each share is mounted on our virtual machines as read-only directories. Similarly, with block storage, we care mostly about volume granularity, as most migration utilities operate at this level.

Volume Clustering: Given a large heterogeneous IT infrastructure, we first apply a clustering method to identify groups of volumes that share similar characteristics. The volume are clustered based on their meta level information which are obtained by aggregating the metadata of all the files on the same volume. We apply the K-means algorithm to obtain the volume clusters. After all the volumes are clustered, we select a representative volume from each cluster and further analyze the business sensitivity of every representative volume. We assume the volumes in the same cluster share a similar sensitivity score.

Cognitive Learning: Business sensitivity is a critical factor in hybrid cloud migration. We need to analyze the sensitivity of the representative volume in each volume cluster, which is defined as the number of sensitive files divided by the total number of files on that volume. The current approach to detecting sensitive files requires in-depth content scanning, which is intrusive and expensive. However, even a single storage volume may contain millions of files, in TB or PB scale. It will take a tremendous amount of time to do a full content scan. In StackInsights, we develop a cognitive learning component to predict the sensitivity of files based on easily obtained metadata, which significantly reduces the total running time.

File Content Crawling: We randomly sample a subset of files from the selected representative storage volume, crawl their content, and apply the traditional rule based approach to determine the sensitivity. Files are identified as sensitive or non-sensitive by matching against a list of regular expressions and keywords predefined by users. The definition of sensitive files can be modified or extended with the domain knowledge of specific industry verticals. The crawling output contains attributes such as file name, file path, the number of total tokens excluding stop words, the number of matching key words, email address, phone number, social security number, and credit card number. Users can define the file sensitivity labeling rule. For example, a file can be labelled as sensitive if it contains any sensitive information in the dictionary. Users can also specify a more stringent rule, such as the file is sensitive only if the percentage of sensitive information is above a certain threshold. The sensitivity labels of these files are correlated with their metadata, which compose the training data for building our sensitivity prediction model.

Intelligent File Sampling: The quality of training data has a significant impact on the performance of machine learning model. In StackInsights, we develop a clustering based progressive sampling method to obtain a “good” training data. All the files on a storage volume are first clustered using their metadata, for example, via K-means. We then compute the percentage of data points assigned to each cluster. A random sampling is performed on each cluster to select data points proportionally, with respect to the previously obtained percentages. We crawl the content of the selected files and determine their sensitivity using the approach introduced in file content crawling. A progressive sampling method is applied to determine the final total sampling size. We start from a relatively small sampling percentage and apply the aforementioned clustering-based sampling to obtain a set of training data. A machine learning model is trained on this data set. We obtain the model’s classification accuracy on a held-out test dataset. Comparing with the classification accuracy from the previous run, if the accuracy improves, we will do an incremental sampling on all the clusters, per user defined sampling size. If the change of classification accuracy is within a predetermined threshold or the total sampling size reaches an upper bound, the sampling process will be stopped.

Metadata-based Sensitivity Prediction: We use the newly sampled dataset to train a binary classification machine learning model. Each file is represented as a feature vector, derived from the file metadata.The output is the classification label “sensitive” or “non-sensitive”. Once the machine learning model is built, we can then apply it to classify the remaining files on the volume based on the available metadata obtained from the infrastructure scan.

V. Experiment

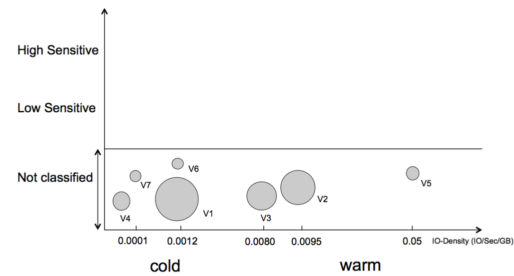
Our environment consists of roughly 100 different accounts, with a wide variety of storage requirements. Each account has a mixture of file, block, and object storage. We choose a mid-size account, whose storage infrastructure is predominantly file storage. This account has two data centers, each one with a set of NetApp filers in clustered mode (roughly 33.8 TB). We install one secure virtual machine inside each site preloaded with our StackInsights scanning codebase. We use those virtual machines to scan the environment and extract the necessary information for further analysis. The storage filer gives us a map of what the file storage infrastructure looks like. We first build a tree of the infrastructure by starting from the cluster and then down to the file level. In parallel, we poll the storage filers for IOPS for each storage volume. This will allow us to measure the IO density for each volume. We use IBM's TADDM tool to get a picture of the different running workloads. Because the filers in the environment are all NetApp filers, we use the ONTAP APIs to extract file metadata, as well as performance metrics from the filers.

A. IOPS and file metadata collection

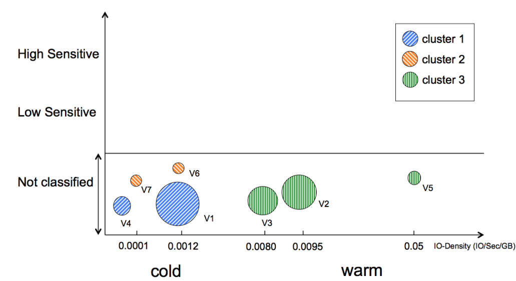
The IOPS for each volume is collected over a four-week time window. We then compute the hourly-average IO density (IO per second per GB) for each volume. In Figure 3(a), all the volumes are aligned along the X-axis according to their IO density. As we can see, except V5, the IO density of all the other volumes is between 0.0 and 0.01, which is relatively cold. Note that since the highest IO density is only around 0.05, we indicate the corresponding hotness as "warm" on the X-axis.

In total, we extract the metadata for more than 13 million files. One file meta data example is shown in Table I. The volume level metadata is then obtained by aggregating the metadata of all the files on the same volume. One volume metadata example is shown in Table II. The "Top3ExtensionbySize" attribute is the top three file extensions ranked by their total size. "NotModifiedin1YearCount" is the percentage of files on that volume that have not been modified in the past one year in terms of file counts. Similarly, "notAccessedin1YearSize" is the percentage of files on that volume that have not been accessed in the past one year in terms of file sizes. All the other attributes are likewise.

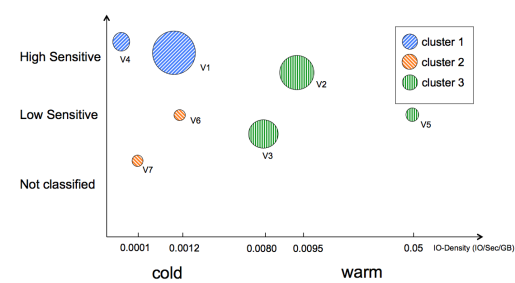
After the volume level metadata is obtained, we apply K-means to do clustering on all the volumes. Due to the relatively small number of volumes, we empirically set $K = 3$. The optimal value of K can be determined by the elbow method, which considers the percentage of variance explained by the clusters against the total number of clusters. Figure 3(b) shows the clustering results. Note that we have not considered the data sensitivity yet, so all the volumes are still aligned along the X-axis. We select a representative volume from each cluster, which is defined as the one with the minimum total distance to the other volumes in the same cluster. A sensitivity analysis is performed on each representative volume. The volumes in the same cluster are assumed to share similar sensitivity score.



(a) Volumes are aligned by IO density



(b) Volume clustering



(c) Volume sensitivity and IO density map

Figure 3: StackInsights volume analytical maps

B. Metadata-based Sensitivity Prediction

1) Training Data: For each representative volume, we build a machine learning model to learn the sensitivity of all the files. One of the selected volume from the first data center contains 3.9 million files, which are 2.64 TB in size. From the infrastructure scan introduced in Section IV, we obtain the metadata of all the files. We randomly select a subset of the files to crawl their content and determine the sensitivity using the dictionary based approach introduced in Section IV. Specifically, we use Apache Tika to scan the file content. A file is considered as sensitive if it contains any sensitive information in the dictionary defined by the user. We finally obtain a set of 114,854 files with both their metadata and sensitivity labels, where 66,221 (57.65%) files

Table I: A file metadata example

Attribute	File name	File extension	File path	Last accessed time	Creation time	Changed time	Last modified time	File size (bytes)	Bytes used
Value	Feedback Survey 2015.docx	.docx	/path/to/file/location	2015-10-21 09:28:37.00	2015-10-21 09:28:31.00	2016-08-08 20:51:35.00	2015-10-21 09:28:37.00	20195	20480

Table II: A volume metadata example

Attribute	Volume size	Total file count	Total file size	Top3ExtensionbySize	Top3ExtensionbyCount	NotModifiedin1YearCount	NotModifiedin1YearSize	NotModifiedin3YearCount
Value	6.06 TB	3,627,061	2.64 TB	nsf, zip, xls	doc, xls, pdf	94.18%	89.44 %	0.0%
Attribute	NotModifiedin3YearSize	NotAccessedin1YearCount	notAccessedin1YearSize	NotAccessedin3YearCount	NotAccessedin3YearSize	NotAccessedAfter2WeekCount	NotAccessedAfter2WeekSize	IO Density
Value	0.0%	81.03%	76.28%	0.0%	0.0%	49.10%	53.13%	7.89E-3

labelled as sensitive. Similarly, we obtain another set of 39,571 files with both their metadata and sensitivity labels from a presentative volume in the second data center. This data set includes 21,284 sensitive files (53.79%). In the following, we will refer to these two data sets as dataset I and dataset II.

2) Feature Engineering: Given the training data, we derive features from file metadata for the classification model. Specifically, the features are divided into several categories: file name, file extension, file path, file size related, and time related. *Name Features:* we extract the name of the file in plain text. The file name can contain textual information that indicate the file sensitivity. For example, a file named "patent disclosure review_Feb2_2015.docx" probably contains intellectual property information and should be considered as sensitive. We model each file name using the bag-of-word approach (removing all numeric characters, punctuation marks, and stop words) and represent it as a vector $v = [v_1, ..., v_n]$, where n is the size of the vocabulary and v_i is the frequency of word i in the file name. *Path Features:* the file paths are extracted from the file system. We extract all the folders that are d levels away from the root. Assuming that m folders are found, we store them in a list $l = [l_1, ..., l_m]$. For a given file f, we represent it as a feature vector $v = [v_1, ...v_m]$ of length m, where $v_i = 1$ if f belongs to folder l_i, otherwise 0. *Extension Features:* we collect all the extensions that belong to our training set and store them in a list $e = [e_1, ..., e_m]$, where m is the total number of extensions. For a given file f, we will represent it as a feature vector $v = [v_1, ..., v_m]$, where $v_i = 1$ if f has extension e_i, otherwise 0. *Size Related Features:* the file size represents the size that was allocated on disk to the file, while the bytes used represents the bytes that were actually written. *Time Related Features:* we include three time related features for each file: specifically, the time difference between the last accessed time and the creation time, the difference between the changed time and the creation time, and the difference between the last modified time and the creation time. All the differences are in the number of days. *Feature Summary:* After the features in each category are collected, we concatenate them into a larger feature vector to represent each file. Because the size of the file path feature grows exponentially with the depth d, we choose $d = 2$ empirically in our experiments. All the features are normalized into the range $[0, 1]$. Table III summarizes the total feature size for each category and the overall size of the feature vector.

Table III: Feature summary

Feature category	dataset I	dataset II
File name	29736	14861
File path	788	315
File extensions	1170	316
File size related	2	2
Time related	3	3
Total feature size	31699	15497

Table IV: Top ten features

Feature Type	dataset I	dataset II
Extension	.url	.xls
Extension	.properties	.txt
Extension	.mdm	.html
Extension	.pas	.net
File Size	-	-
Bytes Used	-	-
Last access time diff	-	-
Change time diff	-	-
Last modified time diff	-	-
Text	"feature"	"username"

3) Feature Selection: We investigate which features are the most significant in the machine learning model. We apply mutual information to select the top features. Table IV shows the top ten selected features. Among the top ten features, file extension features take the most percentage. In addition, the two file size related and three time related features are also significant. Text tokens "feature" and "username" are also among the top ten. Note that we use "username" to replace an actual username due to privacy. We do not find any file path features in the list, which indicate that the location of the file in the filesystem may carry less significance. For example, one particular folder may include both sensitive and non-sensitive files.

4) Prediction Models: After all the features are extracted, we build machine learning models on our training data and apply them to predict the file sensitivity using meta data. Specifically, we compare the performance of several well-known classification models: Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and Random Forest.

All the experiments are conducted using 10 fold cross validation. Naive Bayes has the advantage of having no parameters to optimize on. Logistic Regression has only one parameter: the regularization parameter C. In order to select the optimal value C, we run grid search using

10-fold cross-validation on multiple values of C. We find that the best C value is 0.9. For SVM, the linear kernel is selected. In practice, we find that the RBF kernel takes a very long time to converge. The optimal regularization parameter C is selected following the same procedure as with Logistic Regression. The optimal C value is set to be 0.8 for linear SVM. We use the default parameter setting for Random Forest, where the number of tree in the forest is 10, no maximum tree depth constraint, and the samples are drawn with replacement. Table V shows the performance of each model in terms of overall accuracy, precision, recall, and F1 score. In our classification problem, the positive class is "sensitive" while the negative class is "non-sensitive". Specifically, the precision is defined as $tp/(tp+fp)$, where tp is the number of true positives and fp the number of false positives. The recall is $tp/(tp+fn)$, where fn is the number of false negatives. Accuracy is defined as the number of samples that are correctly classified divided by the total number of samples. The F1 score is $2 \times (precision \times recall)/(precision + recall)$.

Table V: Models on datasets of balanced classes

Model (dataset I)	Accuracy	Precision	Recall	$F1$
Naive Bayes	0.8044	0.8348	0.8238	0.8293
Logistic Regression	0.8115	0.8664	0.7959	0.8296
SVM	0.8309	0.8730	0.8269	0.8493
Random Forest	**0.9014**	**0.9250**	**0.9022**	**0.9135**

Model (dataset II)	Accuracy	Precision	Recall	$F1$
Naive Bayes	0.7780	0.7855	0.8081	0.7966
Logistic Regression	0.7923	0.8350	0.7652	0.7985
SVM	0.8055	0.8493	0.7762	0.8111
Random Forest	**0.8739**	**0.8926**	**0.8703**	**0.8813**

Table VI: Model confusion matrices

Random Forest (dataset I)	Non-Sensitive	Sensitive
Non-Sensitive	21493 (0.88)	2824 (0.12)
Sensitive	3999 (0.12)	29112 (0.88)

Random Forest (dataset II)	Non-Sensitive	Sensitive
Non-Sensitive	7897 (0.86)	1247 (0.14)
Sensitive	1535 (0.14)	9107 (0.86)

In Table V, the percentages of sensitive files in dataset I and II are 57.65% and 53.79%, respectively. Therefore, the classes in the training data are roughly balanced. As we can see, Random Forest has the best performance among all the models over all the metrics. The precision and recall on dataset I are above 90%. In contrast to other models, Random Forest, as an ensemble method, combines the predictions of several base estimators, i.e., decision trees. Each tree in the ensemble is built from a sample drawn with replacement from the training set. When splitting a node during the construction of the tree, the split is chosen as the best split among a random subset of the features. Since both the feature size and sample size are large in our classification, as a result of this randomness, the variance of the forest is reduced due to averaging, hence yielding an overall better model. In practice, the percentage of sensitive files in the training data depends on the specific domain and sensitivity labeling. In some domain, the sensitivity labelling may be stringent, resulting in a relatively small percentage of sensitive files. We also design experiments to test the performance of machine learning models for such case. Still, Random Forest has the best performance among all the models over all the metrics. Due to space limit, we do not show the results here. We select Random Forest as the final machine learning model.

To have a detailed analysis of the classification results, we show the confusion matrices of Random Forest (one fold in a two-fold cross validation) in Table VI, which allows us to see how well the model performs on the classification of each class. Overall, the error ratios on false positives and false negatives are balanced on both datasets.

5) Prediction Model Usage: We do not intend to use the above prediction model to completely replace the traditional content scanning method. As we can see, the prediction model is based on meta data and cannot achieve 100% accuracy. In data governance and security, the misclassification of sensitive data can be catastrophic for an organization. Therefore, a thorough sensitivity screening has to be performed in order to make sure all the sensitive information are identified. From the machine learning model, all the files that are predicted as sensitive will be labelled as sensitive data. We can then perform intensive content scanning method on all the files that are predicted to be non-sensitive. For example, after applying Random Forest on dataset I, 25,492 files are predicted as non-sensitive. The content scanning based method will then be applied to these files, so that the 3,999 misclassified sensitive files can be identified. In contrast to the content scanning of all the 57,428 files, we now only need to perform content scanning on 25,492 files, significantly smaller than the original number of files. There are certainly non-sensitive files miscalssified as sensitive files. For example, 2824 non-sensitive files are misclassified as sensitive files. As a result, they will be "over-protected". However, the percentage of such files only takes 4.91% of the total files. As a simple comparison baseline, with the percentage of sensitive files in the training data (i.e., 57.65%) for dataset I, a user can randomly selects 57.65% data, and label them as sensitive and the remaining as non-sensitive, without using the prediction model. They can then perform content scanning on the previously labelled non-sensitive files in order to identify any sensitive information. Note that in this baseline, among the 57.65% files that are labelled as sensitive, 42.35% files are actually non-sensitive (based on the percentage of non-sensitive files in the training data), therefore, $57.65\% \times 42.35\% = 24.41\%$ amount of non-sensitive files are misclassified as sensitive, therefore "over-protected", in contrast to 4.9% that are misclassified by the prediction model.

6) Prediction Ranking and Running Time: After the machine learning model is trained, we apply it to predict the sensitivity of the remaining files on the same volume. For dataset I, we apply Random Forest to predict the sensitivity of the remaining 3.9 million files and measure its running time. On a local machine with 2.5 GHz Intel Core i7 CPU and 16GB of RAM, the total running time is 112 minutes. As a comparison, it took more than 30 hours to scan the content of only 228,000 files (about 5.85% of 3.9 million), in order to determine their sensitivity. StackInsights therefore reduces the total running time by orders of magnitude.

We also apply the trained learning model to predict the sensitivity of files on other volumes in the same data center. Table VII shows the prediction results: the predicted sensitive files number, volume sensitivity, and running time in seconds. As we can see, V1 and V4 have sensitivity close to 1. They also belong to the same cluster in Figure 3(b). V2, V3, and V5 have sensitivity between 0.45 and 0.70, which are in the same cluster. V6 and V7 are in the same cluster, with predicted sensitivity 0.5758 and 0.1728, respectively.

Table VII: Prediction results on all the volumes

	Total file #	Sensitive file #	Sensitivity	Running time (sec)
V1	400415	385369	0.9624	286.28
V2	7798224	5501763	0.7055	5993.20
V3	3894711	1804798	0.4633	6720
V4	170808	170808	1.0000	134.01
V5	1481322	902804	0.6095	1133.58
V6	686	395	0.5758	0.51
V7	81	14	0.1728	0.06

7) Migration Insights: After the sensitivity of all the files on a storage volume is predicted, we can compute the sensitivity score of volumes. The sensitivity score is defined as the number of sensitive files divided by the total number of files on that volume. As shown in Figure 3(c), the volumes are ordered based on their sensitivity level and hotness. Therefore, all the volumes which are cold and low sensitive can be migrated to the public cloud. The remaining should be migrated to the private cloud or remain on premise.

VI. Conclusions and Future Work

We have introduced StackInsights, a cognitive learning system which automatically analyzes and predicts the cloud readiness of workloads. StackInsights correlates the metrics from application, data, and infrastructure layers to identify the business sensitivity of data as well as their hotness in terms of infrastructure performance, and provides insights into hybrid cloud migration. Given the scale of data and infrastructure, a machine learning model is developed in StackInsights to predict file sensitivity based on the metadata. In contrast to traditional approach which requires intrusive and expensive content scanning, StackInsights significantly reduces the total running time for sensitivity classification by orders of magnitude, therefore, is scalable to be deployed in large scale IT environment. Our current system is mainly focused on understanding the sensitivity of textual files. In the future, we will analyze the sensitive information from multimedia data, such as images, videos, and audios. Similarly, we can predict their sensitivity based on the meta level information. Last but not the least, the cognitive learning capabilities of StackInsights can be greatly enhanced by collecting more metadata across the stack.

References

[1] Markets and Markets. http://www.marketsandmarkets.com/.

[2] Symantec Data Classification Services https://support.symantec.com/en_US/article.HOWTO124009.html.

[3] Varonis Data Classification Framework. https://www.varonis.com/products/data-classification-framework/.

[4] Y. Park, S. C. Gates, W. Teiken, et al. An experimental study on the measurement of data sensitivity. In Proceedings of BADGERS, pages 70-77, 2011.

[5] M. Ali and L. T. Jung. Confidentiality based file attributes and data classification using tsf-knn. In Proceedings of ICITCS, pages 1-5, 2015.

[6] D. Sanchez, M. Batet, and A. Viejo. Detecting Sensitive Information from Textual Documents: An Information-Theoretic Approach. In Proceedings of MDAI, pages 173-184, 2012.

[7] M. Hart, P. Manadhata, and R. Johnson. Text Classification for Data Loss Prevention. In Proceedings of PETS, pages 18-37, 2011.

[8] M. Mesnier, E. Thereska, G. R. Ganger, et al., File classification in self-* storage systems. In Proceedings of ICAC, pages 44-51, 2004.

[9] M. Hajjat, X. Sun, Y.-W. E. Sung, et al. Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud. In Proceedings of SIGCOMM, 2010.

[10] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda. Decision support tools for cloud migration in the enterprise. In Proceedings of IEEE Cloud, pages 541-548, 2011.

[11] M. Menzel and R. Ranjan. Cloudgenius: Decision support for web server cloud migration. In Proceedings of WWW, pages 979-988, 2012.

[12] IBM - Tivoli Application Dependency Discovery Manager. http://www-03.ibm.com/software/products/en/tivoliapplicationdependencydiscoverymanager

[13] IBM - Cloud Brokerage Solutions. https://www-935.ibm.com/services/us/en/it-services/systems/cloud-brokerage-service/.

[14] IBM Security Guardium Family. http://www-03.ibm.com/software/products/en/ibm-security-guardium-family.

[15] NetApp ONTAP http://www.netapp.com/us/products/platform-os/ontap/index.aspx.