Survivability of Cloud Databases - Factors and Prediction

Jose Picado*
Oregon State University
picadolj@oregonstate.edu

Willis Lang
Microsoft Gray Systems Lab
wilang@microsoft.com

Edward C. Thayer
Microsoft
edth@microsoft.com

ABSTRACT

Public cloud database providers observe all sorts of different usage patterns and behaviors while operating their services. Service providers such as Microsoft try to understand and characterize these behaviors in order to improve the quality of their service, provide new features for customers, and/or increase the efficiency of the operations. While there are many types of patterns of behavior that are of interest to providers, such as query types, workload intensity, and temporal activity, in this paper, we focus on the lowest level of behavior – how long do public cloud databases survive before being dropped? Given the large and diverse relational database population that Azure SQL DB has, we present a large-scale survivability study of our service and identify some factors that can demonstrably help predict the lifespan of cloud databases. The results of this study are being used to influence how Azure SQL DB operates in order to increase efficiency as well as improve customer experience.

ACM Reference Format:
Jose Picado, Willis Lang, and Edward C. Thayer. 2018. Survivability of Cloud Databases - Factors and Prediction. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3183713.3190651

1 INTRODUCTION

The emergence of cloud computing services as the dominant growth area for technology enterprises is well accepted as world-wide public cloud revenues are forecasted to more than double to $20B (Platform-as-a-Service) and $100B (Software-as-a-Service) from 2016 to 2020 [16]. Service providers generally compete against one another within a few measurable categories such as: price, feature set, performance, and user experience/satisfaction. While surveying customers is a well-established method of understanding which categories need improving and what users would like to see, cloud service providers also have vast quantities of "telemetry" data generated by their service that can be data mined. In today's hyper-paced market, the provider that most quickly is able to leverage this data will have a highly desirable advantage going forward.

Understanding user behavior and usage patterns is key. It is the first step to improving operating efficiency (lowering costs) and maintaining stable performance and high user satisfaction by minimizing service disruptions. This has recently been a hot topic within the database research community. Modeling database workload patterns [22, 25, 34] and predicting user behavior [14, 33, 35] has been well studied. These studies and others in Section 6, have focused on what a user is doing with their database and when they are doing it. This paper focuses on the "lowest-level" question: after a database has been created, *how long will it survive before being dropped.*

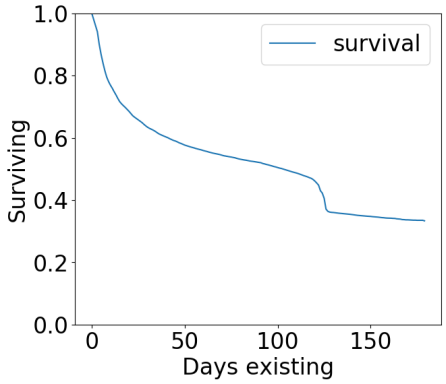


Figure 1: Kaplan-Meier survival curve for singleton databases (with a 2 day survival minimum).

Why is the *survivability* of a database important to a cloud provider? Obviously, it is tied directly to revenue – a provider wishes that databases never get dropped. However, a more subtle reason is the impact of 'short-lived' versus 'long-lived' databases in determining resource provisioning and partitioning. Creating databases is a straight-forward, but non-trivial operational task that requires free resources to be found. Dropping databases also runs counter to some load-balancing/fragmentation policies. Moreover, as we identify in this study, certain customers have usage patterns that call for frequent cycling of databases. Identifying these customers and keeping their databases apart from databases that are long-lived may alleviate noisy neighbor issues and improve back-end efficiency (more in Section 3.1). Ultimately, in this study, we would like to understand the survivability of databases in Microsoft Azure SQL Database [5, 6], and, given a small observation period after database creation, predict whether the database will survive beyond 30 days.

We study the survivability of databases in this paper in the same way as medical researchers study survival outcomes. Using the Microsoft Azure SQL Database (SQLDB) telemetry at our disposal, we consider large populations sets from multiple production SQLDB regions. Figure 1 depicts a Kaplan-Meier survival curve [19] of singleton SQLDB databases (with at least a 2 day survival minimum) over a five month period from a single Azure region. Azure SQLDB also sells databases in elastic pools, which we ignore in this study (see Section 2). We provide the background for Kaplan-Meier (KM)

*Work done at Microsoft Gray Systems Lab.

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3190651

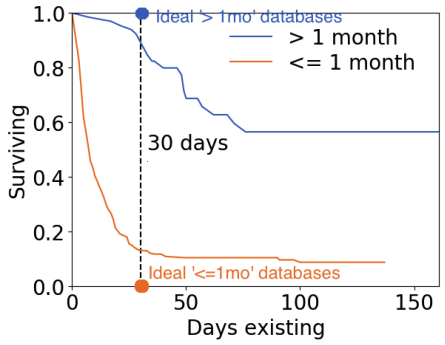


Figure 2: Kaplan-Meier survival curves of a population subgroup classified into 'short-lived' and 'long-lived' after supervised learning. Ideal classification results in the blue and orange curves meeting their respective dots on the plot.

curves in Section 3. Essentially, this figure provides an estimate for the percentage of databases (y-axis) that survive beyond some number of days after their creation (x-axis). (Note: This figure does not reveal database and/or subscription counts, nor creation rates within the region. That is, even though many databases are being dropped, more databases are being created, many times by the same user – revealed both by inspection and our results. *As such, we would persuade the reader not to make any business conclusions from this paper.*) A key issue with survival analysis is the problem of "right-censoring", where we do not know the outcome for a particular individual or sample (i.e., we do not know the actual lifespan of an undropped database created 5 days ago) – KM curves handle this for us, as well as allowing us to compare the survivability of different population sets in a principled way through log-rank testing.

Our goal is to use machine learning to classify the population into subgroups such that their survivability outcomes are significantly different from one another (i.e., *the differences between their KM curves are statistically significant*). In our study, we consider a number of important and intuitive factors and their predictive capabilities on the longevity of a public cloud database including data volume, purchased performance objectives, and owning subscription. Of course, we have access to hundreds of data features. However, we have chosen to focus on a manageable few. Nonetheless, those that we study show statistically significant and marked improvement over a naïve weighted random variable baseline.

Figure 2 depicts one of the results of our longevity predictions to classify databases into two groups: short-lived databases (30 days or less), and long-lived databases (more than 30 days)[1]. Similar to Figure 1, the data is from a single region over the five month span. This result was generated using random forests and the details can be found in Section 4. In this figure, the orange KM survival curve (the classified group "survival less than or equal to 30 days") ideally should terminate on the x-axis at day 30 (marked with an orange dot), and the blue KM survival curve (the classified group "longer than 30 days") ideally should not fall below 100% until day 31 (marked with a blue dot). The distance between the blue curve (at $x = 30$) and the blue dot indicate prediction error (similarly the distance between the orange curve at $x = 30$ and the orange dot).

[1]We plot smoothed KM curves using Python libraries, see Section 3.

The prediction results that we present later in the paper show that we do not achieve perfect classification, but do achieve significant accuracy levels (over 90%), and that the partitioning is statistically significant (according to log-rank test due to the divergence of the KM curves). Keep in mind, this is a hard problem, we cannot read the minds of our users. Classifying the entire population as a whole will only get us so far; identifying a subpopulation where we have a good shot at successful classification versus another that is inherently hard to classify is important (we show this in our results.)

This work presents an industrial study on the factors affecting the survivability of production cloud databases at scale and we present an attempt to classify databases using random forest machine learning. The findings of this study will guide tenant placement and resource provisioning policies in production Azure SQLDB and are the basis for additional SQL Server and cloud infrastructure research and development. We believe this study presents a valuable look at how real users are actually using cloud database services; it provides an inside look otherwise previously unavailable to the general research community. The topics of discussion in this paper are as follows:

- Presenting the analysis of database longevity in at-scale production environments using the well-accepted survival analysis approach from the life sciences.
- Formulating a classification prediction problem for use in a resource provisioning framework based on database longevity.
- An evaluation of random forest models to complete, real-world production data from Microsoft Azure SQLDB showing substantial predictive improvements over a weighted random variable baseline. Additional partitioning techniques presented improve accuracy, precision, and recall to 90% and beyond in many cases.

2 BACKGROUND

Microsoft Azure SQL Database:

Microsoft Azure SQL Database (SQLDB) is one of the leading public, relational database services (Platform-as-a-Service model) today. The service offers databases in a variety of price/performance flavors. As of preparing this paper, the service leverages both a remote storage tier and a local storage tier. The three database editions sold, Basic, Standard, and Premium, are split across the two different storage tiers – Basic and Standard on remote storage and Premium on local storage. Each of these editions further provides multiple service level objectives (SLO) that vary in performance as well as redundancy, back-up retention, data volume, etc.

To provide flexible options for elasticity, after a database is created under a certain edition and SLO, a user can readily change the database to a new SLO or even a new edition with a single command. Doing so allows a user to tailor performance levels on-demand and also manage service costs. For instance, it has been observed that users scale down their SLOs on Fridays and scale them back up on Monday morning for the pending work week.

SQLDB also provides an "elastic pool" provisioning model for the different editions just discussed. That model, as the name suggests, allows users to create many databases that share resources from a single resource pool; in contrast to singleton databases that have

dedicated resources that are not shared. In this work, we ignore elastic pool databases and focus on the "single database" option.

Azure SQL Database telemetry:

For this study, we use the SQLDB telemetry that is emitted from each unique database from its creation through to when it is dropped. Each database emits all sorts of telemetry such as utilization levels [21], query meta-data, file sizes, and other database properties. Using these telemetry streams, we try to find features that are predictive of different survival lengths of our cloud databases. For business and privacy reasons [4], and discussion length, we are not able to examine and/or discuss all possible features at our disposal, but we present some intuitive features that may be predictive.

Machine learning:

In a supervised machine learning task, we are given training data $D = \{(\mathbf{x_1}, y_1), \cdots, (\mathbf{x_n}, y_n)\}$ and the goal is to learn a *model* h that approximates the function f such that $f(\mathbf{x_i}) = y_i$. Each item $(\mathbf{x_i}, y_i) \in D$ is called a *training example*. In a classification task, y_i is a categorical value, while in a regression task, y_i is a numerical value. Each vector $\mathbf{x_i}$ consists of a set of features, where each *feature* represents a property of training example i. In this paper, we use *random forests* as our model [10]. A random forest model is an ensemble of decision trees. Each tree is learned independently using a subset of features, which are selected randomly. The final prediction of a random forest model is obtained by aggregating over the predictions of all learned trees. Random forests serve as a general-purpose technique, are fast to train, and have been shown to produce highly accurate predictions [2, 7].

3 ANALYZING DATABASE SURVIVAL

3.1 Longevity-guided Resource Provisioning

As we discussed briefly in the introduction, there are obvious business reasons that can be treated as motivations for this work. A database that survives an additional hour translates to direct revenue for the provider. Needless to say, there are many business and user experience improvements that are derived from a deep understanding of the segments of databases and segments of users that drop their databases and why.

Our motivations, instead, are on the back-end improvements to a cloud service that are possible if we were able to characterize users and databases into different classifications of database longevity. This would allow us to partition users and their databases and the resources that we provision accordingly for a number of reasons and, in this discussion, we describe two. First, we may be able to avoid any sort of service disruption (e.g., performance) due to maintenance if we know that a database will soon be dropped anyway. Providers must constantly roll out service updates and some of these may impact performance if only for a short time. We have observed that there are many users that frequently churn through databases for various reasons including the nature of their applications. If a set of updates is non-critical, say a new feature, there is no need to risk impacting a database that will be dropped; the user will simply receive the update when they create a new database that launches the latest software. By provisioning a subcluster of resources for these databases we can easily employ such a policy.

Secondly, as we mentioned in Section 2, Azure SQLDB aims to provide flexibility in terms of database price/performance. Changing database service level objectives (SLOs) up and down the performance ladder involves resource allocation and deallocation respectively. These events occur alongside database create and drop. If we can classify databases as short versus long-lived, we can alleviate resource allocation contention between long-lived databases that change their SLOs and users that frequently create and drop databases. Furthermore, dropping a database after a load-balancer has moved it lowers operational efficiency. Again, this points to a partitioning of resources and databases on some longevity basis. Doing so may improve the quality of service for all users.

3.2 Survival Analysis Tools

Survival analysis is a collection of statistical techniques for analyzing the expected duration of time until an event occurs [20]. Survival analysis has been traditionally used in life sciences and medical fields, where the focus is on the survivability of individuals in a population that suffer from some condition or the outcome when a treatment is applied. We use survival analysis tools to analyze the survivability of cloud databases in Microsoft Azure SQL Database (SQLDB). In our case, the population consists of databases, and the *event* of interest is the death of a database, i.e., a database is dropped.

Survival analysis tools are useful for determining the proportion of a population that will survive after a given time t. It is likely that, at the time of answering this question, some individuals in the population have not been subject to the event of interest. For instance, at the time of writing of this paper, vast numbers of SQLDB databases in our population have not been dropped (thankfully). The individuals in the population that have not experienced the event of interest are labelled as *right-censored* [20]. This means that we do not have the complete data about the lifespan of these individuals, and only observe the current lifespan duration.

The *survival function*, defined as $S(t) = P(T > t)$, gives the probability that the event of interest has not yet occurred at time t for a randomly chosen individual with lifespan T. We use the survival function to analyze the lifespan of databases. For instance, we can compute the probability that a database will survive a given amount of time, and compare the probability of survivability of databases in different populations, e.g., Basic vs. Premium databases.

The survival function is theoretically a smooth curve. It can be empirically estimated using the *Kaplan-Meier (KM) estimator* [19]. The KM estimator is given by $\hat{S}(t) = \prod_{i:t_i<t} \frac{n_i - d_i}{n_i}$, where n_i is the number of individuals at risk and d_i is the number of events at time t_i. The larger the population, the better the KM estimator can estimate the survival function. An important property of the KM estimator is that it can take into account right-censored individuals. The KM estimator can be visualized using the KM curve, which plots time t against $\hat{S}(t)$, as seen in Figure 1. We use the survival analysis tools in the Lifelines package [3] in Python.

3.3 Survival Analysis of Azure SQL Databases

In this section we analyze the survivability of SQLDB databases. We create a dataset consisting of cloud databases in SQLDB created over a period of five months. We consider three of the largest

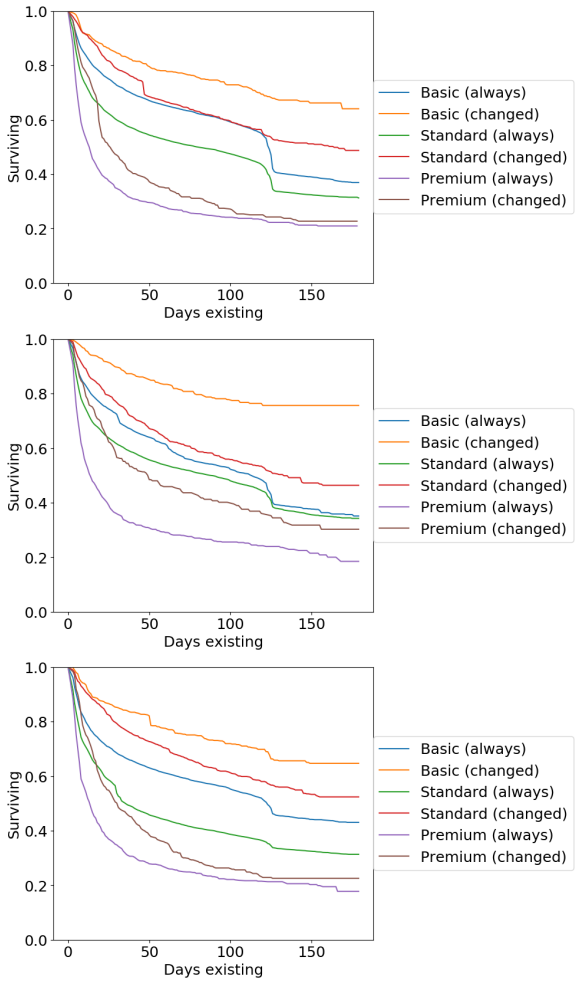


Figure 3: Kaplan-Meier curves for Basic, Standard, and Premium databases, sub-categorized by whether they changed edition, over Region-1 (top), Region-2 (middle), and Region-3 (bottom).

Azure regions around the world which we call Region-1, Region-2, and Region-3. Additionally, we have filtered the databases down to singleton databases of Basic, Standard, or Premium editions, belonging strictly to external clients. (There are internal databases that Microsoft provisions for internal users as well as for serving other external facing products and services that Microsoft sells; these are not included.)

In this section we provide some interesting observations that we discovered from our dataset. These observations are seen in the three regions that we analyzed. Note that these are not experiments; this is a survivability study of SQLDB databases. Let T be the lifespan of database I. We label I as *ephemeral* if $T \leq 2$ days, *short-lived* if $2 < T \leq 30$ days, and *long-lived* if $T > 30$ days.

On Azure, databases are associated with a subscription belonging to a customer (customers can have many subscriptions). We found that a small subset of all subscriptions create *only* databases that are ephemeral. Many of these databases are the result of frequent cycling of databases, a usage pattern identified for some customers. These databases represent a significant percentage of the total population. Therefore, by simply looking at historical data, we can identify customers that follow this pattern, and keep their databases separately from short-lived and long-lived databases.

Observation 3.1. *A low percentage of all subscriptions create only ephemeral databases. These subscriptions show a usage pattern that calls for frequent cycling of databases.*

In this discussion, we filter out ephemeral databases from our dataset, and focus on short-lived and long-lived databases. Even though we omit a significant number of databases, we are still analyzing the lifespan of databases associated with a high percentage of subscriptions. This is because a large percentage of all subscriptions create both ephemeral, as well short-lived or long-lived databases.

Figure 1 depicts the KM survival curve of all databases hosted in Region-1. The KM curves for Region-2 and Region-3 follow a similar decay trend. In all of the curves there is a drop at around 120 days. This can be explained *in part* by certain special incentive offerings ending and databases drop at around that time. Generally, Basic and Standard edition databases contribute to this drop-off. The curves flatten around 0.4, meaning that there is a probability of 40% that a database (in our defined population) is alive after 130 days.

A user can change a database to a new edition during its lifetime. We sub-categorize databases in each edition into two groups: databases that did not change edition during their lifetime, which we refer to as "**always**", and databases that changed edition during their lifetime, which we refer to as "**changed**". Figure 3 shows the KM curves for Region-1, Region-2, and Region-3, categorized by edition and sub-categorized into "always" or "changed". Databases of different editions follow different trends. For instance, Basic databases have a rate of decay significantly lower than Premium databases.

Observation 3.2. *The survival function of databases is different for each edition.*

In Figure 3, we can see that the survival function is also different for Basic-always and Basic-changed groups, as well as Standard-always and Standard-changed groups. However, we noticed that proportionally few databases in Basic and Standard edition switch to a new edition during their lifetime. This is not the case for Premium databases. This is expected, as Premium is the most expensive edition offered in SQLDB, and it is common for users to downgrade databases from Premium to another edition for a forthcoming low-utilization period and upgrade back when necessary.

Observation 3.3. *Proportionally fewer Basic and Standard databases (compared to Premium) change edition during their lifetime.*

These observations provide insights about how customers use the SQLDB service. Further, we use these insights to formulate our prediction problem in Section 4, as well as to design experiments in Section 5.

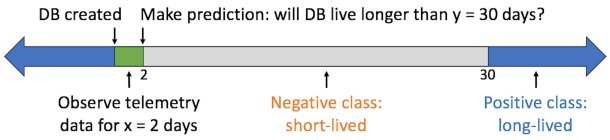


Figure 4: After $x = 2$ days from creation, can we predict whether the database will live longer than $y = 30$ days?

4 PREDICTING DATABASE LIFESPAN

4.1 Problem Formulation

Given the potential benefits of partitioning databases into short-lived and long-lived, as described in Section 3.1, we pose the following problem: *Given the telemetry data produced by a database instance I in the first x days after I is created, can we predict whether I is going to live more than y days?* We perform a classification task where the two classes are: live more than y days, and live less than or equal to y days. As we are making a prediction x days after database I is created, we assume that I lives longer than x days. If this is not the case, then there is no need to make a prediction. In this paper, we focus on the case where $x = 2$ days and $y = 30$ days. That is, after just 2 days, we predict whether a database will be short-lived or long-lived.

Example 4.1. Consider the timeline presented in Figure 4. Let I be a database instance created on June 1st at 10:00am, marked in the figure. We observe telemetry data produced by I for 2 days, represented by the green region. On June 3rd at 10:00am (2 days after I is created), we predict whether I will be long-lived (be alive after July 1st at 10:00am, 30 days after it is created), or short-lived (dropped before July 1st at 10:00am, represented by the grey region).

We learn a random forest [10] model to classify databases as short-lived or long-lived. Random forests produce highly accurate predictions, even when trained with relatively small amounts of data. Furthermore, one can compute the importance of the input features, which is useful for identifying the most predictive factors for determining the lifespan of cloud databases.

4.2 Features

Telemetry streams consist of raw data that capture information about databases and events related to them. We perform feature engineering to obtain a set of features that are useful in our prediction task. These features are derived from the telemetry data, and are used as input to the random forest model.[2] Features fall into the following categories.

Creation time: Telemetry data includes the date and time in which a database is created. We first localize the date and time according to the region in which the database is hosted. Then, we compute the following features:

- Day of the week (1-7)
- Day of the month (1-31)
- Week of the year (1-52)
- Month of the year (1-12)
- Hour of the day (0-23)

These features allow the model to capture temporal patterns. For instance, the model may discover that databases created on weekends tend to live less than 30 days. One reason for this trend may be that these database are created and dropped by an automated process.

Server and database names: A database is created in an Azure SQL Database logical server. The user must provide a server name and database name. For server and database names, we compute the following features:

- Length
- Number of distinct characters
- Distinct character rate (number of distinct characters / length)
- Whether name contains letters and digits
- Whether name contains upper and lower case letters
- Whether name contains non-alphanumeric symbols

The goal of these features is to determine whether a server/database is created manually or by an automated process. For instance, a name that is manually written by a user may have a low character rate, as the words composing the name may contain multiple repeated characters. On the other hand, a randomly created name generated by an automated process is more likely to have a high character rate.

We experimented with features based on n-grams (character level) from server and database names. We discuss the impact of these features in Section 5.4.

Database size: The size of the database may change during the time that we observe telemetry data. We consider features that capture both the absolute size of a database, as well as changes in size during the observed period of time. We consider the following features:

- Maximum, minimum, average, and standard deviation of the absolute database size in megabytes
- Rate of change in size from day of creation to day of prediction

The rate of change in size may provide clues about the database lifespan. For example, if the size of a database does not change in the first 2 days, then it may the case that the database will live longer than 30 days.

Edition and performance level: At any moment, a database belongs to an edition and performance level. A user may change the edition and performance level of a database throughout its lifetime. Each performance level is assigned a number of database transaction units (DTUs) [5]. We compute the following features for editions and performance levels:

- Number of edition/performance level changes
- Number of distinct editions/performance levels
- Edition at the time of prediction
- Performance level at the time of prediction
- Difference between edition/performance level at time of creation and edition/performance level at time of prediction
- Maximum, minimum, and average DTUs assigned to the database based on its performance level

Subscription type: When a database is created, it is associated with a subscription. Azure offers several types of subscriptions, e.g., trial, consumption, benefit programs, etc. We create a feature for

[2]We also examined many additional features, in adherence to our privacy policy [4].

each subscription type t, which takes a value of 1 if the subscription associated with the database is of type t at the time the database is created, and 0 otherwise.

Subscription history: We created features to capture the historical behaviour of the user associated with the subscription. For instance, if all databases associated with a subscription live less than 30 days, then it is likely that a newly created database associated with this subscription will also live less than 30 days. Let I be the database for which we are computing features, C be the subscription associated with I, T_c be the time in which I was created, and T_p be the time of prediction for I. We obtain all databases associated with subscription C. These include databases that are alive between T_c and T_p, as well as databases that were dropped before T_c. We group these databases in three groups: 1) databases created before T_c and dropped after T_c, 2) databases created before T_c and dropped any time (even before T_c), and 3) databases that are created after T_c and before T_p. Notice that group 1 is a subset of group 2. For each of these groups, we compute the following features:

- Number of databases
- Maximum, minimum, average and standard deviation of the size of all databases, where size is the maximum size that the database had (only for groups 1 and 2)
- Maximum, minimum, average, and standard deviation of the lifespan of all databases (only for groups 1 and 2)

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

Dataset: We obtain the telemetry data produced by Microsoft Azure SQL DB databases in three regions, which we call Region-1, Region-2, and Region-3. This telemetry spanned 5 months within the past year. As seen in Section 3, databases in each edition have different usage patterns. Therefore, over each region, we divide the databases into three subgroups, according to the edition in which they were created: Basic, Standard, and Premium. Notice that even though the edition of a database may change, e.g., from Basic to Standard, all subgroups are mutually exclusive. We run experiments on each subgroup, resulting in a total of nine sub-experiments (three regions and three subgroups per region).

In our experiments, we use telemetry data for $x = 2$ days to predict whether a database will live more than $y = 30$ days, i.e., discriminate between short-lived and long-lived databases. A database is classified as positive if it lived more than 30 days, otherwise it is classified as negative. We also experimented with different values for x and y. However, in this paper, we only present the results for the case where $x = 2$ and $y = 30$ days.

Approach: We learn a random forest model for each subgroup in the dataset. We divide the databases in each subgroup into 80% training and 20% testing sets. We perform parameter tuning for each model by doing grid search using 5-fold cross-validation over the training set. We use the tuned model to make predictions on the testing set. We perform each experiment 5 times and report the average accuracy, precision, and recall over the testing set. Accuracy is the ratio of correctly classified databases. Precision is the fraction of examples correctly classified as positive among all examples classified as positive. Recall is the fraction of examples correctly classified as positive among all actual positive examples. We use the scikit-learn package [29] in Python.

Baseline: We use a weighted random classifier as our baseline. The random classifier makes predictions the following way. It first computes the probability p that an example is positive solely based on the class distribution in the training data. For each example in the testing set, it computes a random number r between 0 and 1. If $r < p$, it classifies the example as positive; otherwise, it classifies it as negative.

We first present the 'whole population' results, where we trained and classified the population from a region as a whole. While the results are good, we sought improvement. Subsequently, we devised a way to determine when our classifications were more reliable (*confident*) or less reliable (*uncertain*). We also present those classification scores, which show significant improvement when we deemed a classification confident. Ultimately, we show that for confident classifications (roughly 60% of the whole population), we achieve significantly improved scores compared to the whole population scores.

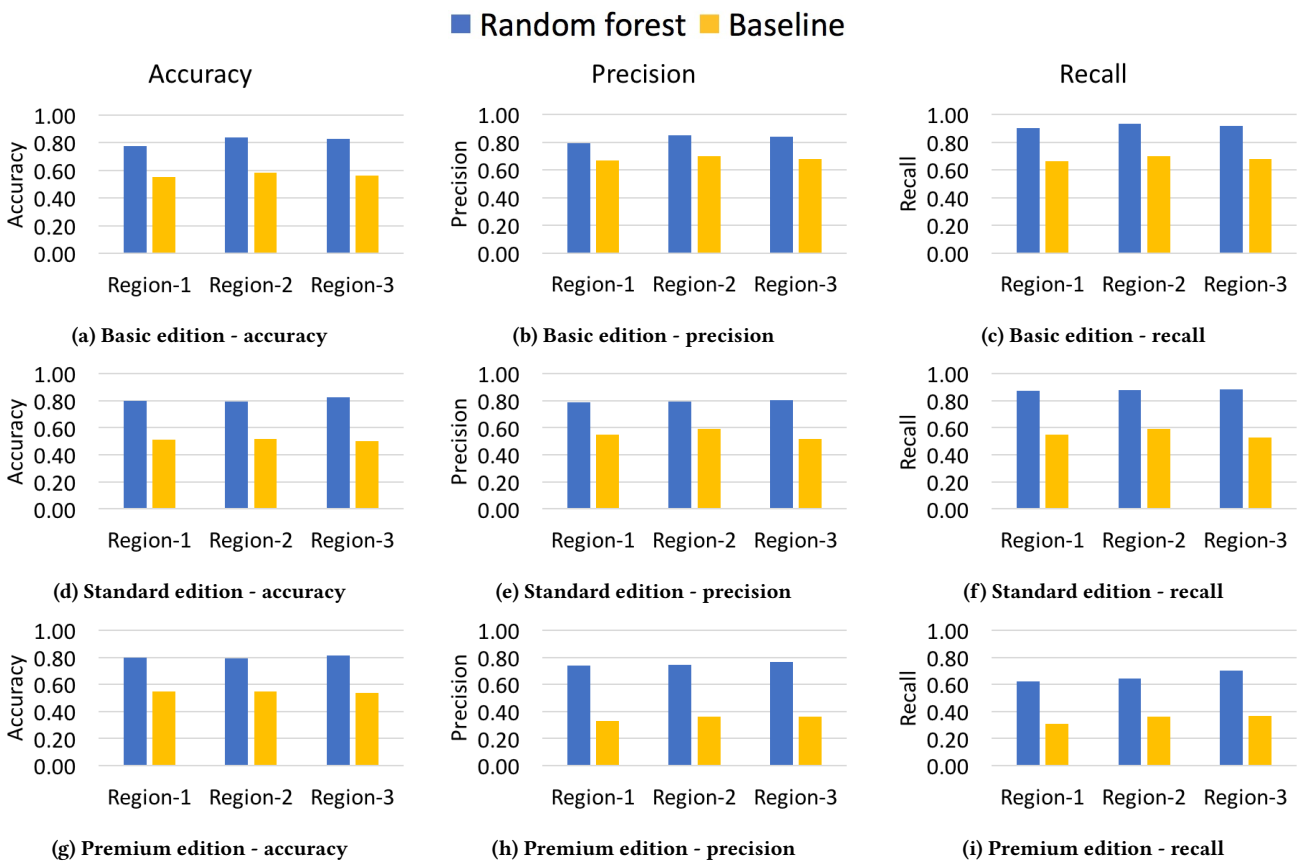
5.2 Results - Whole Population

Figure 5 shows the accuracy, precision, and recall scores over Basic, Standard, and Premium databases, over the three regions. The blue bars represent the predictions of the random forest model, while the yellow bars represent the predictions of the baseline. The random forest model significantly outperforms the baseline over all editions and all regions. Over Basic edition, our model obtains an average accuracy of 0.81 compared to 0.56 by the baseline, 0.83 precision compared to 0.68 by the baseline, and 0.92 recall compared to 0.68 by the baseline. Over Standard edition, our model obtains an average accuracy of 0.81 compared to 0.51 by the baseline, 0.79 precision compared to 0.55 by the baseline, and 0.88 recall compared to 0.56 by the baseline. Over Premium edition, our model obtains an average accuracy of 0.80 compared to 0.55 by the baseline, 0.75 precision compared to 0.35 by the baseline, and 0.66 recall compared to 0.35 by the baseline.

Overall, our model obtains an accuracy higher than 0.80. This means that our model makes correct predictions at least 80% of the time. To better interpret the results, it is useful to look at the precision and recall scores. For instance, over Basic edition, our model obtains precision of 0.83 and recall of 0.92, on average. This means that from the databases that our model predicts to live longer than 30 days, 83% of these predictions are correct. Further, the model is able to identify 92% of the databases that live longer than 30 days.

Over the Premium edition, it is difficult to predict whether a database will be long-lived. This is reflected in the low recall scores in Figure 5i. There are two reasons for this. First, the population of Premium databases is significantly smaller than the population of Basic or Standard databases. Therefore, we have fewer training examples. Second, the positive and negative class distribution is more imbalanced among the Premium databases than the Basic or Standard databases. This is reflected on the low scores obtained by the baseline.

Statistical significance of classifications: To determine how well our model separates short-lived and long-lived databases, we divided the databases in the testing set according to their predicted



(a) Basic edition - accuracy (b) Basic edition - precision (c) Basic edition - recall

(d) Standard edition - accuracy (e) Standard edition - precision (f) Standard edition - recall

(g) Premium edition - accuracy (h) Premium edition - precision (i) Premium edition - recall

Figure 5: Results of predicting database lifespan over Basic (top), Standard (middle), and Premium (bottom) editions for the whole population.

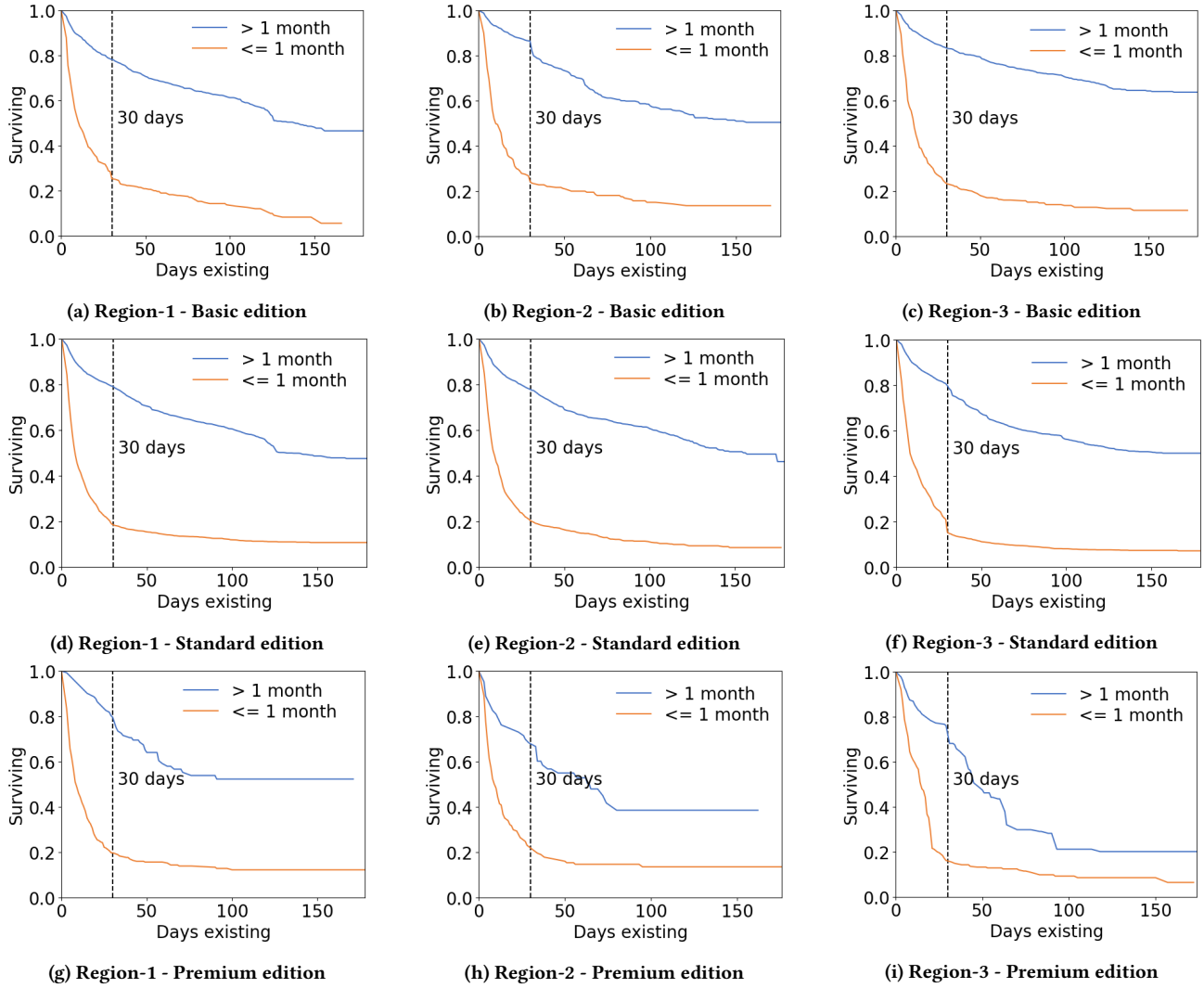
class, and then plotted the KM curves for each group. Figure 6 shows the KM curves for Basic, Standard, and Premium databases predicted to be short-lived (orange) and long-lived (blue). Clearly, our model is able to separate the databases with relative success. However, our model is not perfect: the blue lines should start dropping after the 30-day mark, while the orange lines should have completely dropped by the 30-day mark (recall the dots in Figure 2). To test how significantly different are the classified groupings, we performed log-rank tests. The *log-rank test* is a hypothesis test that compares the survival distributions of two samples, where the null hypothesis is that the survival distributions are identical [20]. All classified groupings by the random forest model have p-values below 0.0000001. Therefore, the separation of the two classes is statistically significant. On the other hand, the classified groupings by the baseline have p-values greater than 0.05, which is not considered statistically significant.

5.3 Results - Confidence Partitioning

According to our motivation, we would like to make resource provisioning decisions based on the predicted classification of databases. There may be high costs of making incorrect decisions. For instance, if a database that is predicted to be short-lived actually lives more than 30 days, it may impact our load balancing policies. Therefore, besides predicting the class of an example, we would like some confidence level about the prediction. Random forests output an estimate of the probability that an example belongs to a class. We use this probability estimate as a confidence level [36]. Notice that by confidence level, we simply mean the probability estimates generated by our model; it is different from its meaning in common statistical terms.

In this section, we use the confidence levels of the predictions to divide predictions into two groups: **confident** and **uncertain**. Confident predictions should have high confidence levels, i.e., the model predicts that an example belongs to a class with high probability. *Therefore, if we want to mitigate misclassification costs, we can take actions only on confident predictions.* For instance, if the model predicts that a database will live more than 30 days with 95% probability, then this database can be safely moved to a server that only contains long-lived databases. Ideally, confident predictions should obtain high accuracy, precision, and recall scores.

In decision trees, the probability that an input example is classified as positive (negative) is equal to the fraction of positive (negative) examples in the leaf making the prediction. For instance, if there are 10 examples in a leaf node and 8 of them are positive, when the decision tree classifies an example as positive based on this leaf node, the probability that the example is positive is 80%. The class



(a) Region-1 - Basic edition (b) Region-2 - Basic edition (c) Region-3 - Basic edition

(d) Region-1 - Standard edition (e) Region-2 - Standard edition (f) Region-3 - Standard edition

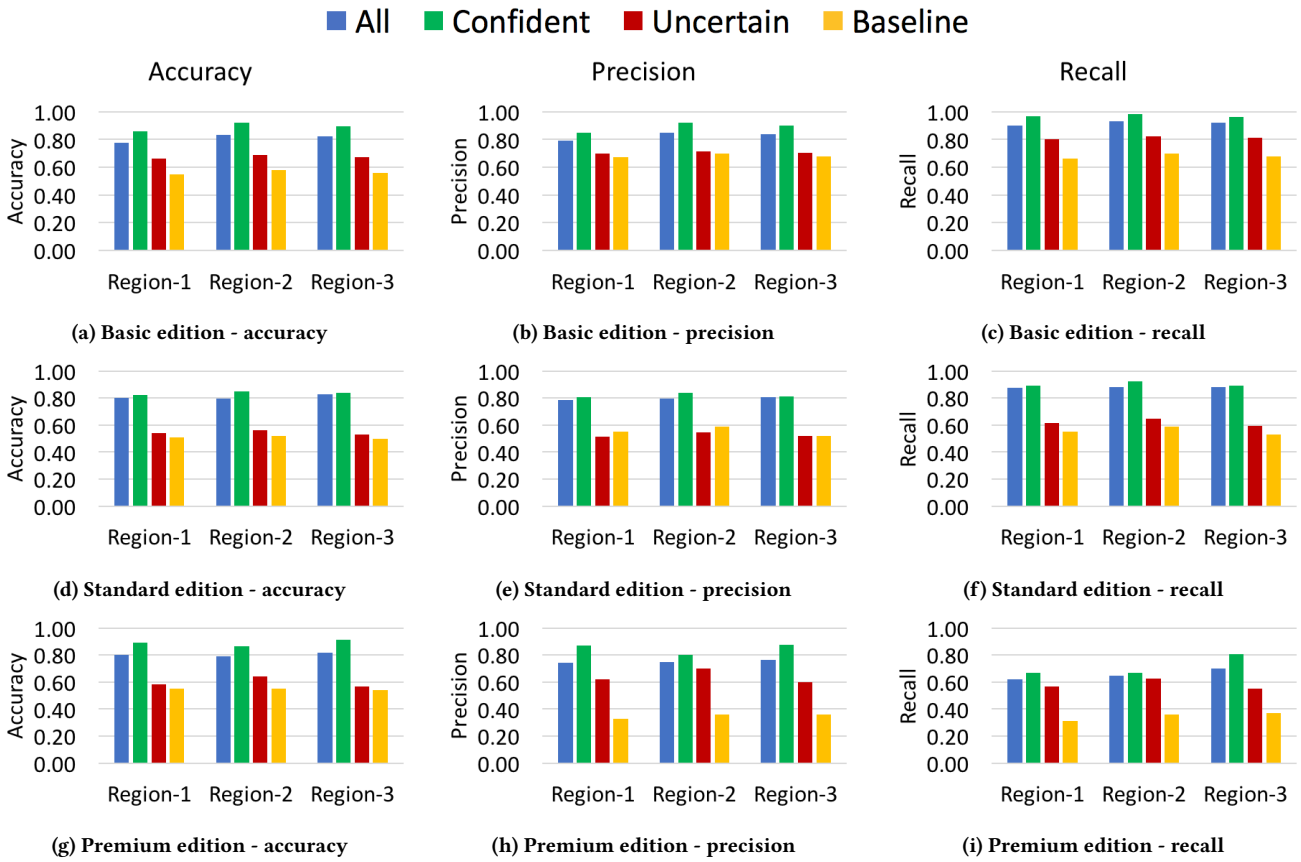
(g) Region-1 - Premium edition (h) Region-2 - Premium edition (i) Region-3 - Premium edition

Figure 6: Kaplan-Meier curves for whole-population classified groupings over Basic (top), Standard (middle), and Premium (bottom) editions.

probabilities in a random forest are the result of averaging over the class probabilities of the trees in the forest. An example is classified as positive if the probability of being positive is greater than 0.5; otherwise it is classified as negative. We use these probabilities as confidence levels for the predictions.

To determine whether a prediction is confident or uncertain, we set a threshold t, where $0.5 \leq t \leq 1$, and use it in the following way. Let p the predicted probability that an example is classified as positive. As normal, if $p > 0.5$, the example is classified as positive; otherwise it is classified as negative. If $p \geq t$ or $p \leq 1 - t$, the prediction is considered as *confident*. On the other hand, if $1 - t < p < t$, the prediction is considered as *uncertain*. That is, predictions where the predicted probability is close to 0.5 are considered as uncertain. To determine the value for the threshold t we use the distribution of classes in the training data. Let q be the percentage of positive examples in the training data. Then, we set $t = \max(q, 1 - q)$. For example, if 70% of the training examples are positive, then $q = 0.7$. Thus, $t = \max(0.7, 0.3) = 0.7$.

Figure 7 shows the accuracy, precision, and recall scores for all predictions (blue bars, same scores as in Figure 5), confident predictions (green bars), uncertain predictions (red bars), and the baseline (yellow bars, same scores as in Figure 5). The blue and and yellow bars are included in this figure for ease of comparison. Table 1 shows the percentage of predictions that are confident or uncertain. Confident predictions consistently improve our previous results, in some cases reaching an accuracy of 0.92. We see the biggest gains in Basic and Premium editions, where confident predictions cover on average 63% and 71% of all predictions, respectively. There is not much improvement over the Standard edition because the distribution of short-lived and long-lived databases is balanced, i.e., approximately the same amount of positive and negative examples. Therefore, the threshold for separating confident and uncertain



(a) Basic edition - accuracy (b) Basic edition - precision (c) Basic edition - recall

(d) Standard edition - accuracy (e) Standard edition - precision (f) Standard edition - recall

(g) Premium edition - accuracy (h) Premium edition - precision (i) Premium edition - recall

Figure 7: Results of predicting database lifespan over Basic (top), Standard (middle), and Premium (bottom) editions with confident and uncertain partitioning.

predictions is low. This results in confident predictions covering almost all predictions. This is confirmed in Table 1, where confident predictions over the Standard edition cover in average 90.3% of all predictions. Interestingly, uncertain predictions still outperform the baseline in most cases, particularly in Premium edition.

Statistical significance of classifications: As in Section 5.2, we divided the databases in the testing set according to their predicted class, and plotted the KM curves for each group. Figures 8 and 9 show the KM curves for confident and uncertain predictions, respectively. As can be seen, confident predictions better separate the two classes. This is reflected in higher accuracy, precision, and recall scores. The log-rank tests over confident classified groupings output p-values below 0.0000001, making the separation statistically significant.

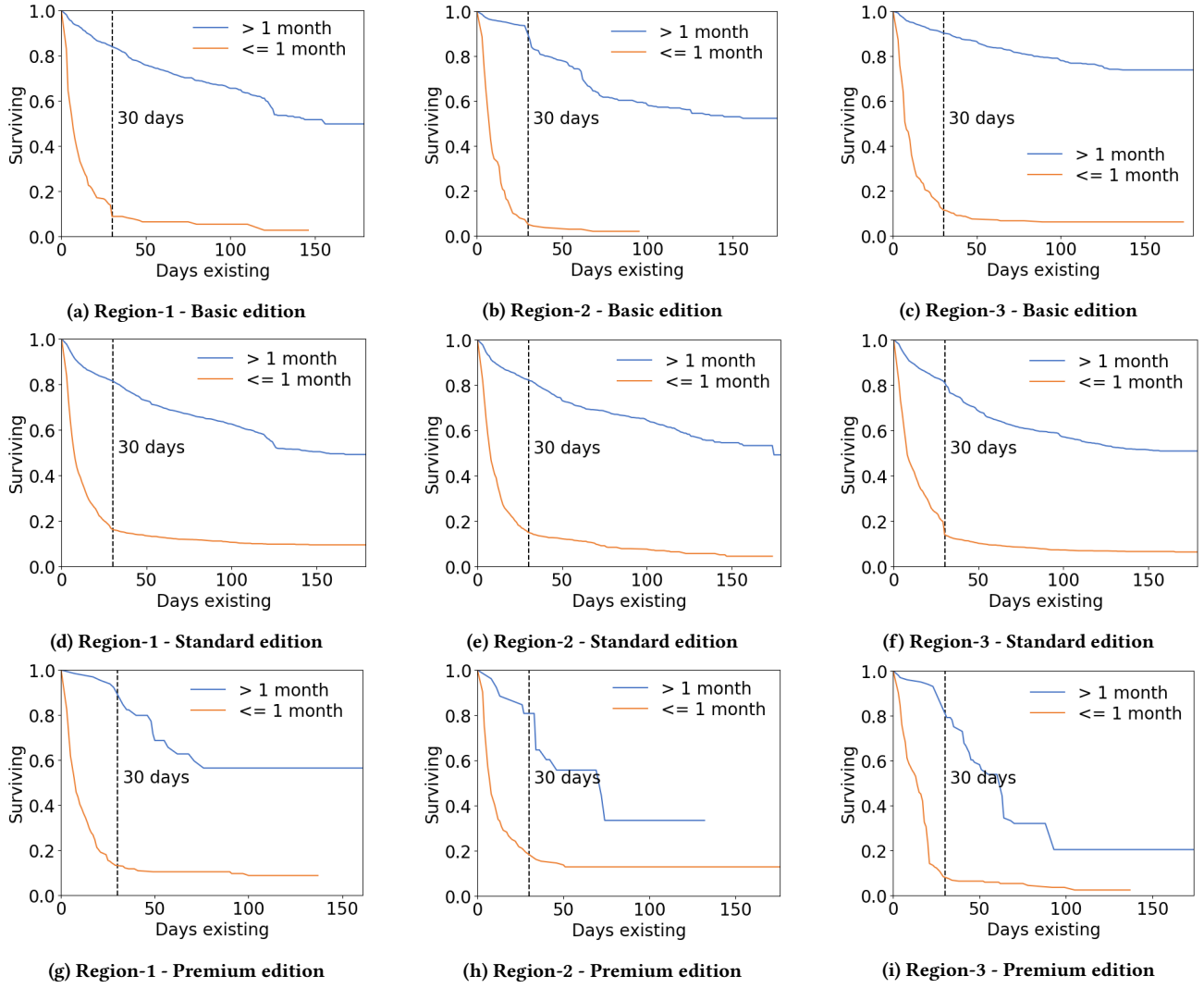
On the other hand, uncertain predictions cannot successfully separate the two classes. This is reflected in Figure 9, where the blue and orange curves are mostly close to each other. Table 2 shows the p-values obtained from performing log-rank tests over the uncertain classified groupings. Over the Basic edition, the separation is still statistically significant. However, this is not the case over Standard and Premium editions. Specifically, in Region-1 and Region-3 over the Standard edition, and Region-3 over the Premium edition, the separation of classes is not statistically significant. Figure 9 shows that the separation of classes over these sub-groups is as good as a random classifier.

Edition	Region	Confident	Uncertain
Basic	Region-1	58%	42%
	Region-2	63%	37%
	Region-3	68%	32%
Standard	Region-1	92%	8%
	Region-2	82%	18%
	Region-3	97%	3%
Premium	Region-1	71%	29%
	Region-2	69%	31%
	Region-3	73%	27%

Table 1: Percentage of confident and uncertain predictions.

5.4 Predictive Factors

One benefit of random forests is that one can extract the importance of features, i.e., which features are most predictive. We use the *gini-importance* to measure the feature importance. The gini-importance is defined as the total decrease in node impurity averaged over all trees in the forest. We use the gini-index to measure the node impurity, defined as $2p(1-p)$, where p is the proportion of positive examples in a node [18].



(a) Region-1 - Basic edition (b) Region-2 - Basic edition (c) Region-3 - Basic edition

(d) Region-1 - Standard edition (e) Region-2 - Standard edition (f) Region-3 - Standard edition

(g) Region-1 - Premium edition (h) Region-2 - Premium edition (i) Region-3 - Premium edition

Figure 8: Kaplan-Meier curves for *confident* classified groupings over Basic (top), Standard (middle), and Premium (bottom) editions.

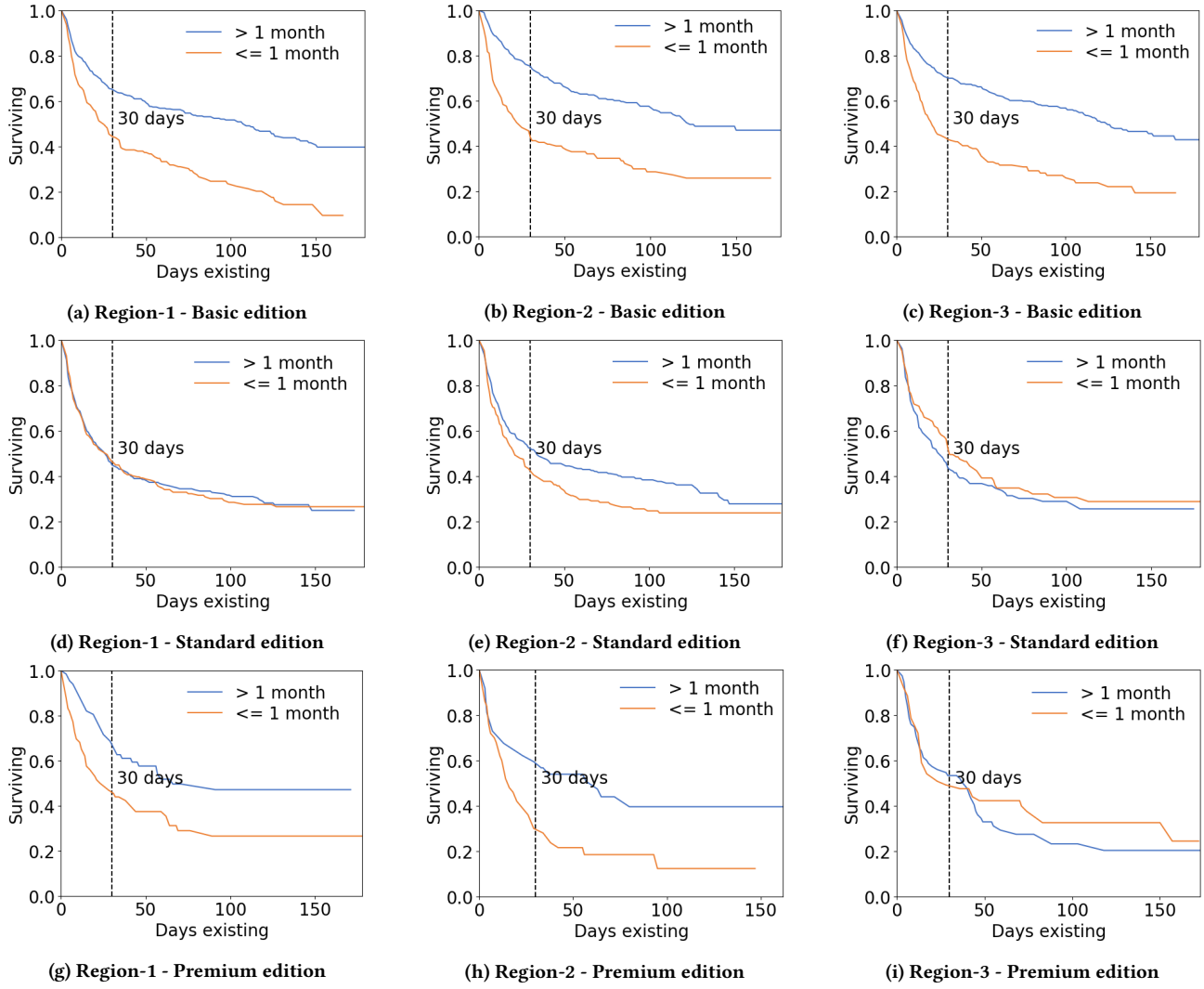
Edition	Region	P-value
Basic	Region-1	< 0.0000001
	Region-2	< 0.0000001
	Region-3	< 0.0000001
Standard	Region-1	0.925429
	Region-2	0.010043
	Region-3	0.379127
Premium	Region-1	0.004774
	Region-2	0.008219
	Region-3	0.371621

Table 2: P-values resulting from log-rank tests over uncertain classified groupings.

The most predictive features are related to the subscription history, i.e., history of databases owned by the subscription. This is intuitively expected. For instance, if all databases associated with a subscription are short-lived, then it is likely that a new database associated with this subscription will also be short-lived. Also, the features indicating the number of databases created by the subscription have high importance.

The second most predictive features are related to the server and database names. This confirms our hypothesis that these features are useful for identifying whether a database is created manually or by an automated process. However, we did not see any improvement in accuracy when using features based on n-grams from names. In some cases, top n-grams came from common server and database names, which caused the model to overfit.

Finally, the third most predictive features are related to the creation time, specifically the hour of the day, day of the month, and



(a) Region-1 - Basic edition (b) Region-2 - Basic edition (c) Region-3 - Basic edition

(d) Region-1 - Standard edition (e) Region-2 - Standard edition (f) Region-3 - Standard edition

(g) Region-1 - Premium edition (h) Region-2 - Premium edition (i) Region-3 - Premium edition

Figure 9: Kaplan-Meier curves for *uncertain* classified groupings over Basic (top), Standard (middle), and Premium (bottom) editions.

week of the year in which a database was created. Hour of the day may indicate whether or not the databases are created under automation versus during waking business hours. Similarly, databases created during a regional holiday (our data trace spans five straight months, covering a number of regional holidays in each data center region) also may imply automated database creation.

5.5 Summary of Results

It is non-trivial to predict the lifespan of databases. Generally, customers follow different patterns of behavior. Therefore, two databases may have completely different characteristics even though they belong to the same class. Also, the population contains outliers that have different characteristics from all other databases. Even with these difficulties, our model is able to achieve 80% accuracy over all predictions, and in many cases 90% accuracy over confident predictions. Furthermore, our model can separate classes so that the difference of survival distributions of databases in each class is statistically significant.

It is important to note that we have been able to segment databases (and users) into those that are easier to classify correctly versus those databases (and users) that we struggle with. First, this segmentation does not prune away a significant subpopulation; Table 1 shows that at worst we still have confidence over at least 58% of an edition subpopulation and in some cases over 95% of an edition subpopulation. Second, having a principled way to segment the population allows us to build in policies for uncertain classifications into our longevity-designated resource provisioning schemes (Section 3.1). For instance, we may leave a designated pool of resources for users that we are unable to accurately classify.

Another difficulty in classifying databases as short-lived vs. long-lived is that some databases in each class may have similar characteristics. For instance, there may be not much difference between a database that lived 28 days, compared to a database that lived 32 days. We noticed that a significant percentage of databases that cannot be classified confidently have a lifespan close to 30 days. It is difficult for our model to correctly classify these databases. Therefore, in the case that there are misclassification costs, it is useful to identify which databases are classified with some level of confidence, and only take actions on these databases. Our model is able to make confident predictions in 63% of Basic databases, 90.3% of Standard databases, and 71% percent of Premium databases, on average.

6 RELATED WORK

As we discussed earlier in Section 1, there has been tremendous research interest in all aspects of cloud database services. Some of these focus on learning about user workloads and resource demands of queries [13, 15, 24, 26, 28]. These are all extremely valuable and important studies that suggest that database service providers can take the workload and utilization telemetry that they have to either efficiently allocate resources, or help the database learn how to perform faster.

From a service back-end perspective there are many different issues related to operational reliability and efficiency that demand industrial studies. Some examples of industrial topics and studies include: cost management [17], performance isolation [27], demand forecasting [8, 12, 22], as well as analysis [9], machine learning [32], and data management at scale [30]. There have been a number of survivability studies in other domains, such as studies about the survivability of data in storage systems [23, 31]. Similar to us, these studies analyze the correlation between distinct factors and longevity [31], and share our motivation of understanding survivability for planning and designing provisioning and storage solutions [23]. However, none of the studies discuss the survival of cloud databases, and few of them present real-world data at the scale and detail that we show here. We hope that our work spurs other cloud service providers to also discuss some of the behavior and data that they see in order to share these real-world insights with other researchers.

There are many different statistical and machine learning techniques to perform the analysis that we have done in this paper. The goal of our work was not to compare different approaches, but rather understand database survival in a cloud setting, and identify features that are predictive of lifespan. We used random forests as our model of choice [10]. Random forests have properties that make them useful for our task. They can handle a large number of features without overfitting, and they produce highly accurate predictions [7]. Studies have shown that random forests give good performance over a wide range of metrics [11]. Ensemble of decision trees, such as random forests, have been known to dominate data science competitions [1, 2]. We use the estimated probabilities by the random forests to divide predictions into confident and uncertain predictions. Random forests can successfully estimate class probabilities, even without calibration [11].

7 CONCLUSIONS

Besides the obvious business reasons for database service providers to wish to understand how the service is being used and for what reasons databases are being dropped, there are significant back-end infrastructure and policy improvements that can influence efficiency, user experience, and performance. Consequently, this study represents a production-scale, broad analysis of database survival in Azure SQL Database. We have presented a first look at public cloud database survivability and evaluated random forest classification over our population into short-lived and long-lived classes. We find that we can achieve over 90% accuracy in the many statistically significant subpopulations evaluations. Importantly, doing so allows us to identify users (subscriptions) that generally create short-lived or long-lived databases and with this knowledge, we will intelligently provision designated resources for different pools of databases.

REFERENCES

[1] Algorithms. https://www.kaggle.com/wiki/Algorithms. Accessed: 2017-11-17.
[2] Lessons from 2 million machine learning models on kaggle. https://www.kdnuggets.com/2015/12/harasymiv-lessons-kaggle-machine-learning.html. Accessed: 2017-11-17.
[3] Lifelines. https://lifelines.readthedocs.io/. Accessed: 2017-11-17.
[4] Microsoft privacy policy. https://www.microsoft.com/en-us/TrustCenter/Privacy/default.aspx.
[5] Microsoft corporation. http://azure.microsoft.com/en-us/pricing/details/sql-database, 2016.
[6] P. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, D. Lomet, R. Manne, L. Novik, and T. Talius. Adapting Microsoft SQL Server for Cloud Computing. In *ICDE*, pages 1255–1263, April 2011.
[7] G. Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13:1063–1095, 2012.
[8] J.-H. Böse, V. Flunkert, J. Gasthaus, T. Januschowski, D. Lange, D. Salinas, S. Schelter, M. Seeger, and Y. Wang. Probabilistic demand forecasting at scale. In *VLDB*, pages 1694–1705, 2017.
[9] E. Boutin, P. Brett, X. Chen, J. Ekanayake, T. Guan, A. Korsun, Z. Yin, N. Zhang, and J. Zhou. Jetscope: Reliable and interactive analytics at cloud scale. In *VLDB*, pages 1680–1691, 2015.
[10] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
[11] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, 2006.
[12] S. Das, F. Li, V. R. Narasayya, and A. C. König. Automated demand-driven resource scaling in relational database-as-a-service. In *SIGMOD*, pages 1923–1934, 2016.
[13] S. Das, V. Narasayya, F. Li, and M. Syamala. CPU Sharing Techniques for Performance Isolation in Multi-tenant Relational Database-as-a-Service. In *PVLDB*, 2013.
[14] J. Duggan, U. Cetintemel, O. Papaemmanouil, and E. Upfal. Performance Prediction for Concurrent Database Workloads. In *SIGMOD*, 2011.
[15] A. J. Elmore, S. Das, A. Pucher, D. Agrawal, A. El Abbadi, and X. Yan. Characterizing Tenant Behavior for Placement and Crisis Mitigation in Multitenant DBMSs. In *SIGMOD*, pages 517–528, 2013.
[16] Gartner. www.gartner.com/newsroom/id/3815165.
[17] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *SIGCOMM*, 39(1):68–73, 2008.
[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
[19] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958.
[20] D. G. Kleinbaum and M. Klein. *Survival Analysis: A Self-Learning Text*. Springer Science and Business Media, LLC, 2005.
[21] W. Lang, F. Bertsch, D. J. DeWitt, and N. Ellis. Microsoft Azure SQL Database Telemetry. SoCC, pages 189–194, 2015.
[22] W. Lang, K. Ramachandra, D. J. DeWitt, S. Xu, Q. Guo, A. Kalhan, and P. Carlin. Not for the Timid: On the Impact of Aggressive Over-booking in the Cloud. *PVLDB*, 2016.
[23] Y. Li, E. L. Miller, and D. D. E. Long. Understanding data survivability in archival storage systems. In *SYSTOR*, 2012.
[24] R. Marcus and O. Papaemmanouil. Wisedb: A learning-based workload management advisor for cloud databases. In *VLDB*, pages 780–791, 2016.
[25] B. Mozafari, C. Curino, A. Jindal, and S. Madden. Performance and Resource Modeling in Highly-concurrent OLTP Workloads. In *SIGMOD*, 2013.

[26] V. Narasayya, I. Menache, M. Singh, F. Li, M. Syamala, and S. Chaudhuri. Sharing Buffer Pool Memory in Multi-tenant Relational Database-as-a-service. *PVLDB*, pages 726–737, 2015.

[27] V. R. Narasayya, S. Das, M. Syamala, B. Chandramouli, and S. Chaudhuri. Sqlvm: Performance isolation in multi-tenant relational database-as-a-service. In *CIDR*, 2013.

[28] Y. Park, A. S. Tajik, M. Cafarella, and B. Mozafari. Database learning: Toward a database that becomes smarter every time. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 587–602, 2017.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. In *VLDB*, pages 1816–1827, 2015.

[31] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *FAST*, 2007.

[32] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. Data management challenges in production machine learning. In *SIGMOD*, pages 1723–1726, 2017.

[33] J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, and A. Zeier. Predicting in-memory database performance for automating cluster management tasks. pages 1264–1275, April 2011.

[34] R. Taft, W. Lang, J. Duggan, A. J. Elmore, M. Stonebraker, and D. J. DeWitt. STeP: Scalable Tenant Placement for Managing Database-as-a-Service Deployments. SoCC, 2016.

[35] L. Viswanathan, B. Chandra, W. Lang, K. Ramachandra, J. Patel, A. Kalhan, D. J. Dewitt, and A. Halverson. Predictive provisioning: Efficiently anticipating usage in azure sql database. In *IEEE ICDE*, pages 1111–1116, 2017.

[36] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, 2001.