Forecasting IT Security Vulnerabilities - An Empirical Analysis

Emrah Yasasin[a]*, Julian Prester[b], Gerit Wagner[a], Guido Schryen[c]

[a] *Department of Management Information Systems, University of Regensburg, Universitätsstraße 31, 93053 Regensburg, Germany*
[b] *School of Information Systems, UNSW Business School, Kensington NSW 2052, Australia*
[c] *Faculty of Business Administration and Economics, Paderborn University, Warburger Strasse 100, 33098 Paderborn, Germany*

Abstract

Organization have to deal with a plethora of IT security threats nowadays and to ensure smooth and uninterrupted business operations, firms are challenged to predict the volume of IT security vulnerabilities and to allocate resources for fixing them. This challenge requires decision makers to assess which system or software packages are prone to vulnerabilities, what impact exploits might have, and how many vulnerabilities can be expected to occur during a certain period of time. The academic literature has increasingly drawn attention to the need for predicting IT security vulnerabilities. However, only limited research has addressed the problem of forecasting IT security vulnerabilities based on time series that deal with the specific properties of IT security vulnerabilities, i.e., rareness of occurrence and high volatility. To address this shortcoming, we apply established methods which are capable of forecasting events characterized by rareness of occurrence and high volatility. Based on a dataset taken from the National Vulnerability Database (NVD), we use the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to measure the forecasting accuracy of single, double and triple exponential smoothing methodologies, Croston's method, ARIMA, and a neural network-based approach. We analyze the impact of the applied forecasting methodology on the prediction accuracy with regard to its robustness along the dimensions of the examined system and software packages "operating systems", "browsers" and "office solutions" and the applied metrics. To the best of our knowledge, this study is the first that analyzes the effect of prediction techniques and applies forecasting metrics that are suitable in this context. Our results show that the optimal forecasting methodology depends on the software or system package as some methods perform poorly in the context of IT security vulnerabilities, that absolute metrics can cover the actual prediction error precisely and that the prediction accuracy is robust within the two applied forecasting-error metrics.

Keywords: Security vulnerability, Prediction, Forecasting, Competition setup, Time series

*Corresponding author
Email address: emrah.yasasin@wiwi.uni-regensburg.de (Emrah Yasasin[a])

1. Introduction

The impact of information technology (IT) security vulnerabilities can be substantial: In an industry study, IBM estimates the reputation-related costs which result from software security vulnerabilities leading to a disruption of business operations to range in the millions of dollars per disruption (IBM Global Study, 2013). The economic consequences of breaches have been examined by FireEye, a network security company. Specifically, their data breach cost report for 2016 revealed that 76 % of respondents would take their business away from a vendor that had demonstrated negligent data handling practices (eWeek, 2016; FireEye, 2016). Similarly, the 2016 Cost of Data Breach report by the Ponemon Institute and IBM Security showed that the average total cost of a breach is US$4 million, an increase of 29% since 2013, with disruptions in daily operations being the most severe category of impact (Ponemon Institute, 2016). In the aftermath of a breach, firms are challenged to mitigate the long-term financial impact by restoring customers' trust. In essence, these reports indicate that vulnerabilities pose permanent risks for firms for which they need to be prepared to deal with. These risks are as diverse as they are plentiful, e.g., network attacks (GhasemiGol et al., 2016), loss or theft of personal data, loss or theft of commercially sensitive information, inoperable IT systems (making the business unable to function after being hacked), intellectual property infringement, and defamation or extortion, which can lead to serious financial damage (ContractorUK, 2016).

These economic damages raise the general question of how to control the impact of such vulnerabilities. In particular, this challenge requires decision makers to assess which system or software packages are prone to vulnerabilities, what impact exploits might have, and how many vulnerabilities can be expected to occur during a certain period of time. The importance of this assessment as an input for system and software acquisition, maintenance, and replacement is reinforced by a recent study: Results from Veracode's Bug Bounty survey of 500 IT decision makers working in cybersecurity revealed that 83% of vendors have released code before testing or resolving security issues for bugs (Veracode, 2016; Software Testing NEWS, 2016).

Extant literature offers a plethora of managerial decisions which are contingent on accurate predictions of vulnerabilities. For instance, the expected number of vulnerabilities can be used as a measure of trustworthiness before a certain system or software package is acquired (Kim et al., 2007) or discontinued. Furthermore, assessing the expected number of vulnerabilities can provide valuable input for allocating and prioritizing limited resources for inspecting, patching and testing of an existing software portfolio (Kim et al., 2007; Shin et al., 2011; Walden et al., 2014). Plus, predicting trends in the number of known vulnerabilities that could occur helps decision-makers to take proactive actions to minimize the threats that vulnerabilities may pose (Venter and Eloff,

2004).

The overall impact of security vulnerabilities can be estimated based on the amount of the potential collateral damage and the frequency of occurrences. Our study focuses on the research challenge of predicting the number of security vulnerabilities in subsequent periods of time. To reliably predict the number of vulnerabilities for a particular system or software package, forecasting methods must account for three fundamental properties of security vulnerabilities (Gegick et al., 2009): First, vulnerabilities are rare events (Shin et al., 2011); to be specific, it is not uncommon that there are several months in which no vulnerabilities are reported. Second, with respect to those months where vulnerabilities are observed, there are a few periods where a comparatively high number of vulnerabilities is reported. For instance, 19 vulnerabilities (CVE-2012-1126 to CVE-2012-1144) were reported for the Firefox browser in April, 2012 (MITRE Corporation, 2017a), while there were none in May and June, 2014. And third, time series of vulnerabilities are not necessarily stationary[1], which means that they do not have the same expected value and the same variance at each point in time. A reason for this is the development of software within the version history. While some versions represent minor changes, other versions include substantial changes in the software. For example, the completely overhauled Firefox implemented in the new Quantum version represented major changes in performance and security. These include a stricter and more confined framework for extensions and additional sandboxing (Mozilla, 2017). In our study, we therefore take into account different versions of each package and examine them separately.

The academic literature dealt with the study of IT security vulnerabilities using regression techniques for prediction (Shin and Williams, 2008; Chowdhury and Zulkernine, 2011; Shin et al., 2011; Zhang et al., 2011; Shin and Williams, 2013; Walden et al., 2014), machine learning techniques (Neuhaus et al., 2007; Gegick et al., 2009; Nguyen and Tran, 2010; Scandariato et al., 2014), statistical analyses with the help of reliability growth models and vulnerability discovery models (Ozment, 2006; Ozment and Schechter, 2006; Joh, 2011) and time series analysis (Roumani et al., 2015; Last, 2016). While an evaluation of these methods shows sound performance values, we observe that none of these approaches considered methods which account for the unique rareness of occurrence and high volatility of vulnerabilities. Furthermore, only two recent studies (Roumani et al., 2015; Last, 2016) focus the prediction from a time series perspective. While Roumani et al. (2015) uses ARIMA and exponential smoothing for the prediction of security vulnerabilities, Last (2016) analyzed the forecast of vulnerabilities from different browsers, operating systems, and video players using both regression models (Linear, Quadratic, and Combined) and machine learning techniques. Both studies show an acceptable fit and can be helpful to predict vulnerabilities. However, the techniques

[1] "A time series is *stationary* if its statistical properties (mean, variance and autocorrelation) are held constant over time" (Ferreiro, 1987, p. 65).

applied in these studies are not appropriate for the specific properties of security vulnerabilities discussed before (rareness of occurrence and high vulnerability). There can be methods used and evaluated, in particular Croston's method which is designed for time series with a lot of null zero values [2]. Consequently, this implies that the prediction accuracy can differ due to the characteristics of the forecasting methodology.

Furthermore, the particular system or software package under consideration needs attention as different packages have different release cycles and different number of vulnerabilities that is not taken into account when they are not grouped together. It is necessary to differentiate between different versions due to changes within the development history. We therefore argue that the prediction accuracy depends on the system or software packages. For instance, the number of vulnerabilities is related to the market share and the maturity stage of the product: Alhazmi et al. (2007) for example points out that if a system or software starts to attract attention and users start switching to it, the number of vulnerabilities will increase. Another example is the degree of maturity. A system or software is likely to have more vulnerabilities in their early stages rather than a mature one which has been used and tested for years.

Finally, the usage of suitable accuracy metrics is also a crucial point when examining the forecast quality. The academic literature provides a lot of accuracy metrics (cf. the literature reviews on accuracy metrics Hyndman and Koehler (2006); Hyndman et al. (2006); Willemain et al. (2004); Willmott et al. (1985)), however not all are suitable when the time series are zero-inflated. For example, prediction accuracy metrics which compute the percentage error of the forecast and actual vulnerabilities are not adaptable by definition. These metrics produce infinite / undefined values when there are no actual vulnerabilities reported for a time t.

The aforementioned arguments concerning the methodology, object and metrics of vulnerability prediction result in the research question

"How accurately can different forecasting methodologies predict IT security vulnerabilities?",

where we analyze the accuracy with regard to its robustness along the dimensions of examined system and software packages and applied metrics. To the best of our knowledge, this study is the first that analyzes the effect of forecasting methodologies which take into account the uniqueness and rareness of vulnerability time series and applies forecasting metrics that are suitable in this context.

The remainder of the paper is structured as follows: Next, we provide an overview of related work. In Section 3, we explain our methodology and the data set. In Section 4, we present and discuss the results of our empirical study. The paper closes with a summary.

[2] In our study *zero* means that no IT security vulnerabilities are reported within the observed time horizon.

2. Research Background

In this section, we give a short overview of related research by discussing and highlighting current research streams of IT security vulnerabilities and their forecasting.

2.1. IT Security Vulnerabilities

Currently, there is no standardized definition of the term *security vulnerability*, and answering the question "what a security vulnerability is" remains a challenge (Microsoft Corporation 2015). We adopt the terms "vulnerability" and "exposure" of the U.S. MITRE Corporation as "security vulnerability" for two reasons: First, the "Common Vulnerabilities and Exposures" (CVE) entries are not only used by many empirical papers (Singh et al., 2016; Johnson et al., 2016; Younis et al., 2016; Chatzipoulidis et al., 2015; Ozment, 2006; Ozment and Schechter, 2006; Joh, 2011; Wang et al., 2008; Last, 2016) but also by information security product and service vendors (Schryen, 2011, 2009) such as Adobe, Apple, IBM or Microsoft (MITRE Corporation, 2017b); and second, the definition of vulnerabilities in the context of the CVE program covers weaknesses in the computational logic found in software and hardware components that, when exploited, result in a negative impact on confidentiality, integrity, or availability (MITRE Corporation, 2017c). Therefore, we adopt the CVE system's definition of an information security vulnerability being "*a mistake in software that can be directly used by a hacker to gain access to a system or network*" (MITRE Corporation, 2017c). Accordingly, vulnerabilities allow attackers to successfully violate security policies, for example, by executing commands as another user, by reading or changing data although such access should be restricted, by posing as another entity, or by conducting a denial of service attack (MITRE Corporation, 2017c; Telang and Wattal, 2007).

A schematic classification of vulnerabilities is shown in the figure below:

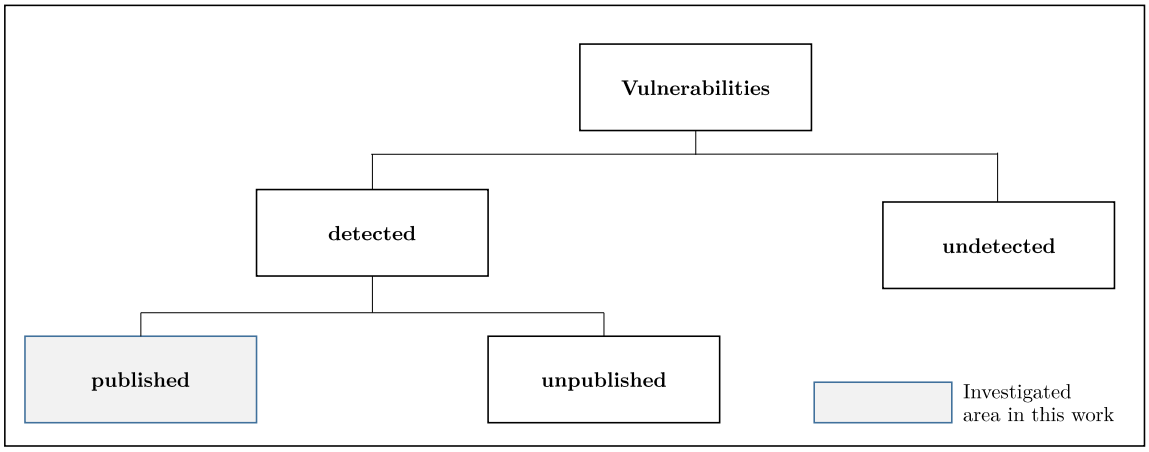

Figure 1: Classification of Vulnerabilities (Schryen, 2011).

Vulnerabilities can occur for several reasons, starting from programming errors, malicious software engineers or unintentional behaviors (Schryen, 2011) and, even though firms strive to reduce security vulnerabilities through technological attempts (Hinduja and Kooi, 2013), it remains a challenging task to detect vulnerabilities. Furthermore, if a vulnerability is detected, an open question is still whether to publish the vulnerability or not. While there are some critical voices to publish as the probability of vulnerability rediscovery to be negligible (e.g., Rescorla (2005)), studies show that vulnerabilities are correlated in terms of rediscovery, and therefore should be announced publicly (Ozment, 2005). In this work, we focus on the published vulnerabilities in order to better forecast them in the future.

2.2. IT Security Vulnerability Forecasting

The following table illustrates the IT security vulnerability forecasting literature we could identify:

Table 1: IT Security Vulnerability Prediction Models

Article	Applications	Used Predictors	Prediction Technique	Data Source
Regression Techniques				
Shin and Williams (2008)	JavaScript Engine of Firefox	Code Complexity	Logistic Regression	Mozilla Foundation Security Advisories & Bugzilla
Chowdhury and Zulkernine (2011)	Firefox Web Browser	Complexity, Coupling and Cohesion	Naive Bayes, Decision Tree, Random Forest, Logistic Regression	Mozilla Foundation Security Advisories & Bugzilla
Shin et al. (2011)	Firefox Web Browser, Red Hat Linux Kernel	Complexity, Code Churn, Developer Activity	Logistic Regression	Mozilla Foundation Security Advisories & Bugzilla & National Vulnerability Database & Red Hat Security Advisory
Smith and Williams (2011)	WordPress, WikkaWikki	SQL Hotspots	Logistic Regression	WordPress & WikkaWikki Vulnerability Reports

Zhang et al. (2011)	Adobe, Internet Explorer, Linux, Apple, Windows	Period of Time between Vulnerabilities	Linear & Regression Models (Least Mean Square & Multi-Layer Perceptron & RBF Network & SMO Regression % Gaussian Processes)	National Vulnerability Database
Shin and Williams (2013)	Firefox Web Browser	Complexity, Code Churn, Prior Faults	Logistic Regression	Mozilla Foundation Security Advisories & Bugzilla
Walden et al. (2014)	PHPMyAdmin, Moodle, Drupal	Complexity, Source Code, Vulnerability Locations	Random Forest	National Vulnerability Database, Project Announcements
Machine Learning				
Neuhaus et al. (2007)	Mozilla Project	Imports and Function Calls	Support Vector Machine	Mozilla Foundation Security Advisories & Bugzilla
Gegick et al. (2009)	Cisco Software System	Non-Security Failures	Classification and Regression Tree Models	Cisco Fault-Tracking Database
Nguyen and Tran (2010)	JavaScript Engine of Firefox	Component Dependency Graphs	Bayesian Network, Naive Bayes, Neural Networks, Random Forest, Support Vector Machine	Mozilla Foundation Security Advisories & Bugzilla

Scandariato et al. (2014)	Android Applications	Text Mining of Java Code	Decision Trees, k-Nearest Neighbor, Naive Bayes, Random Forest, Support Vector Machine	Source Code of Used Applications with Fortify Source Code Analyzer
Statistical Models				
Ozment (2006)	OpenBSD	Number of Failure Data	Reliability Growth Models	OpenBSD Web Page, ICAT, Bugtraq, OSVDB, ISS X-Force
Ozment and Schechter (2006)	OpenBSD	Time between Failures	Statistical Code Analysis, Reliability Growth Models	OpenBSD web page, ICAT, Bugtraq, OSVDB, ISS X-Force
Joh (2011)	Windows XP, OS X 10.6, IE 8, Safari	Number of Vulnerabilities	Vulnerability Discovery Models	NVD, Secunia, OSVDB
Time Series Analysis				
Last (2016)	Different Browsers, Operating Systems, Video Players	Number of Vulnerabilities	Linear & Regression Models (Linear, Quadratic, and Combined), Machine Learning	National Vulnerability Database
Roumani et al. (2015)	Chrome, Firefox, Internet Explorer, Safari, Opera	Number of Vulnerabilities	ARIMA, Exponential Smoothing	National Vulnerability Database

The above table shows that the extant literature mainly uses regression techniques for prediction (Shin and Williams, 2008; Chowdhury and Zulkernine, 2011; Shin et al., 2011; Zhang et al., 2011; Shin and Williams, 2013; Walden et al., 2014). For instance, Shin and Williams (2008) adopted code complexity that differentiate vulnerable functions and investigated whether code complexity can be useful for vulnerability detection. The results indicate that complexity can predict vulnerabilities at a low false positive rate, but at a high false negative rate. In a similar work, Shin et al. (2011) examined if complexity, code churn, and developer activity can be used to distinguish vulnerable from neutral files, and to forecast vulnerabilities. Shin and Williams (2013) showed that fault prediction models and vulnerability prediction models provide good accuracy in forecasting vulnerable code locations across a wide range of classification thresholds. Chowdhury and Zulkernine (2011) developed an approach to automatically predict vulnerabilities based on historical data, complexity, coupling, and cohesion by using four alternative statistical and data mining techniques. The results indicate that structural information from the non-security realm such as complexity, coupling, and cohesion is useful in vulnerability prediction. In their study they were able to predict approximately 75 % of the vulnerable-prone files. Walden et al. (2014) compared the vulnerability prediction effectiveness based on complexity, source code, and vulnerability locations in the source code for the forecast of vulnerable files. They showed that text mining provides a high recall for PHPMyAdmin, Moodle, and Drupal code analysis.

Besides approaches using mainly regression techniques, there are other used predictors and techniques as well. For example, Smith and Williams (2011) analyzed whether SQL hotspots provide a useful heuristic for the prediction of web application vulnerabilities. Their analysis reveals that the more SQL hotspots a file contains per line of code, the higher the probability that this file will contain vulnerabilities. Neuhaus et al. (2007) introduced a support vector machine based tool that achieved high accuracy in predicting vulnerable components in software code based on imports and function calls. Furthermore, Gegick et al. (2009) created a classification and regression tree model to determine the probability of a component having at least one vulnerability. The evaluation shows that non-security failures provide useful information as input variables for security-related prediction models. Nguyen and Tran (2010) demonstrated that dependency graphs are another viable option to predict vulnerable components and Scandariato et al. (2014) used the source code of Android applications as input for text mining approaches, statistical methods and artificial intelligence techniques to determine which components of a project are likely to contain vulnerabilities. After validating their approach by applying it to various Android applications, they determined that a dependable prediction model can be built.

Statistical models were also used to examine vulnerability predictions. For instance, (Ozment, 2006; Ozment and Schechter, 2006) used reliability growth models and statistical analyses showed that these have acceptable one-step-ahead predictive accuracy for the set of independent data points.

Joh (2011) applied vulnerability discovery processes in major web servers and browsers: The analyses show reasonable prediction capabilities for both time-based and effort-based models for datasets from Web servers and browsers.

More recently, time series analysis has also been used to forecast the number of vulnerabilities. For example, Roumani et al. (2015) considered time series models (ARIMA, exponential smoothing) for the prediction of security vulnerabilities. The results reveal that time series models provide a good fit and can be helpful to predict vulnerabilities. Last (2016) analyzed the forecast of vulnerabilities from different browsers, operating systems, and video players using both regression models (Linear, Quadratic, and Combined) and machine learning techniques. The evaluation of these methods indicates significant predictive performance in forecasting zero-day vulnerabilities.

However, a more detailed analysis of these approaches uncovers three issues: 1) The literature on predicting the number of IT security vulnerabilities from a time series approach is rather sparse. 2) Predictions on which software components are more likely to be vulnerable do not provide insights into the volume of vulnerabilities that will occur. And 3), none of these research foci address the uniqueness of vulnerabilities, namely, rareness of their occurrence and high volatility (as noted in Section 1). We therefore concentrate on predicting the number of IT security vulnerabilities from a time series perspective taking into account methods and accuracy metrics that are suitable for these two properties inter alia. The next section explains the different methods and accuracy metrics we used in this study.

3. Methodology and Data

In this section, we motivate and outline the forecast methodologies implemented in our study and introduce a consistent notation (Subsection 3.1), present accuracy metrics to compare the different forecast approaches, which are suitable in the context of security vulnerability forecasting (Subsection 3.2). Finally, we describe the data set in terms of analyzed software systems (Subsection 3.3).

3.1. Forecasting Methodologies

In line with the study of Nikolopoulos et al. (2016), we implement a multiple forecasting approach, where we compare several forecasting methods and evaluate their performance in terms of forecasting accuracy.

We forecast time series of monthly security vulnerabilities using the forecasting horizons of one, two, and three months. We evaluate the results against a test set of held out security vulnerability data. Time-series forecasting approaches are organized in five main research streams: (Exponential) Smoothing methods, regression methods, (advanced) statistical models, neural networks and (other) data mining algorithms (Wang et al., 2009). We refer to Chatfield (2000), who identifies key aspects

which need to be considered when choosing a forecasting method. These include the properties of the time series being forecasted and the forecast accuracy of the method.

In our study, we use two types of forecasting methods. The first group of forecasting methods we use are not in particular designed for the purpose of zero-inflated time series [3]. Yet, these methods are used both in practice and academic literature widely, and very recently for predicting the number of IT security vulnerabilities (Roumani et al., 2015). These forecasting methods within this first group comprise single, double and triple exponential smoothing methods (SES, DES, and TES) which are also referred to as single exponential, Holt's linear trend method, and Holt-Winter's method. In addition, we implement an ARIMA based approach, which is an advanced statistical model.

Regarding our context, time series of IT security vulnerabilities differ from conventional series in the respect that they have multiple periods of zero values. Forecasting methodologies that are appropriate for zero-inflated time series are thus especially suitable in our context (Ogcu Kaya and Demirel, 2015). Such time series with a lot of zero values are well-known in intermittent demand analysis: Many scholars have recognized and contributed to the problems of predicting infrequent and irregular demand patterns, i.e., the observed demand during many periods is zero, interspersed by occasional periods with irregular non-zero demand (Johnston and Boylan, 1996).

We therefore use a second group of forecasting methodologies that are designed for the purpose of handling such time series. In particular we apply Croston's method and a Neural Network based approach. Croston (1972) highlighted the inadequacies of common methods for intermittent demand forecasting and developed a method, which is one of the widely used forecasting methods for intermittent demand (Shenstone and Hyndman, 2005; Syntetos et al., 2015). From a methodological point of view, it is built upon the estimation from the demand size and inter-arrival rate: The original time series is decomposed into a time series without zero values and a second one that captures durations of zero valued intervals (Herbst et al., 2014). In addition, we want to shed light on the following methodological association with Croston's method and SES: When data is aggregated, i.e. in our case if we had grouped the different versions together, the zero-inflation of the data would have been decreased. In the academic literature, it is discussed that such an aggregation could lead to time series containing no zero values for the higher aggregation levels (where the mean intermittent demand interval will be equal to unity) (Petropoulos and Kourentzes, 2015). In this special case, Croston's method is equivalent to SES in the case where all periods have non-zero demands and the literature suggests to use SES instead (Petropoulos and Kourentzes, 2015). However, as we separated different versions of software and system packages, this is not the case for our data.

[3] Zero-inflated time series are time series which contain a lot of zero values and show a high volatility when a value occurs.

Therefore, we include Croston's method. Furthermore, the suitability of Croston's method for such time series has been empirically shown. It performs more effectively in forecasting zero-inflated and intermittent demand time series data (e.g., Kourentzes (2013); Gutierrez et al. (2008)). For example Willemain et al. (1994) have demonstrated that Croston's method gives superior forecasts to some competing methods when predicting zero-inflated time series.

Besides Croston's method, we use a Neural Network based approach that "*are used to provide dynamic demand rate forecasts, which do not assume constant demand rate in the future and can capture interactions between the non-zero demand and the inter-arrival rate of demand events*" (Kourentzes, 2013, p. 198). Kourentzes (2013) have shown evidence for the applicability of neural network approaches in predicting zero-inflated time series. Therefore, we include both Croston's method and artificial neural networks, which better address the specific characteristics of security vulnerability time series data.

The predicted outcome variable $\hat{y}_{t+h|t}$, used throughout the paper, is defined as the forecasted value $\hat{y}$ at time $(t+h)$, where t is the starting time and h the proposed forecast horizon. In our study we test three different forecasting horizons covering short (one month, $h = 1$), medium (two months, $h = 2$), and long (three months, $h = 3$) time frames.

3.1.1. Exponential Smoothing Methods

Single Exponential Smoothing

The idea behind SES is to weigh the most recent observations against the observations from the more distant past using the parameter α. Forecasts are calculated using weighted averages where the weights decrease exponentially as observations lie further in the past. In other words, smaller weights are associated with older observations. SES only depends on the linear parameter l_t, which denotes the level of the series at time t. Due to this definition, SES predicts every value into the future with the same value, derived from the last observed level. Our outcome variable can in this case be described as $\hat{y}_{t+1|t}$. For smaller values of α more weight is given to the observations from the more distant past. The equation for single exponential smoothing is listed in the following:

$$\begin{aligned} \hat{y}_{t+h|t} &= \hat{y}_{t+1|t} = l_t \\ l_t &= \alpha y_t + (1-\alpha) l_{t-1} \end{aligned} \tag{1}$$

Double Exponential Smoothing

Single exponential smoothing can be extended to allow forecasting of data with a linear trend which is called the double exponential smoothing method. This was done by Charles C. Holt in 1957. This method is slightly more complicated than the original one without trend. In order to

add the trend component to the outcome variable $\hat{y}_{t+1|t}$ the term b_t, which denotes the slope of the time series at time t:

$$\begin{aligned} \hat{y}_{t+h|t} &= l_t + hb_t \\ l_t &= \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \end{aligned} \tag{2}$$

While Parameter l_t still denotes the level, b_t represents the slope of the time series. The weight β is used to weigh the slope between the two most recent observations against the observations from the more distant past using the parameter α.

Triple Exponential Smoothing

This approach is an extension of DES with added seasonality often referred as triple exponential smoothing (TES). There are three components in this model (cf. Equation 3). As in the previous model, the first denotes the level, while the second represents the trend component. In TES, the third term s_t denotes the seasonality component. The outcome variable $\hat{y}_{t+1|t}$ can thus be defined as follows:

$$\begin{aligned} \hat{y}_{t+h|t} &= l_t + hb_t + s_{t+h_m-m} \\ l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \end{aligned} \tag{3}$$

Where $h_m = [(h-1) \mod m] + 1$, which ensures that the estimates of the seasonal parameters came from the correct season.

While Parameter l_t and b_t are analogously defined to SES and DES, the weight γ is introduced to weigh the seasonality component over the m most recent time periods.

3.1.2. ARIMA

In an Auto Regressive Integrated Moving Average (ARIMA) model, the future value of a variable is assumed to be a linear function of several past observations and random errors. ARIMA models combine differencing with auto-regression and a moving average model. We used the ARIMA(p, d, q) model where p is the order of the autoregressive part, d is the degree of first differencing involved and q is the order of the moving average part. The general equation of an ARIMA(p, d, q) model is the following (Der Voort et al., 1996; Hyndman and Athanasopoulos, 2018):

$$\hat{y}'_{t+h|t} = c + \Phi_1 y'_t + \Phi_2 y'_{t-1} + \ldots + \Phi_p y'_{t-p} + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q} + e_t \tag{4}$$

where y_t denotes the number of vulnerabilities at time t, $\hat{y}_{t+h|t}$ is the forecast of the time series y. c is a constant and Φ_p are the coefficients (to be determined by the model) of the autoregressive model. e_t is a zero mean white noise error factor and together with the coefficients θ_q forms the moving average terms. Since stationarity is a requirement for ARIMA forecasting models and security vulnerabilities have been found to be non-stationary (Arora et al., 2006, 2010), we appropriately transformed the data using differentiation. With this, $\hat{y}'_{t+h|t}$ and y'_t are the differenced series (degree of differentiation depending on d).

3.1.3. Croston's Method

In order to account for the characteristic properties of security vulnerability time series data, we choose Croston's method as an additional forecasting method, specifically the bias-adjusted version of Croston's method developed by Syntetos and Boylan (1999). The method of Croston (1972) separately forecasts the non-zero periods' magnitudes and the inter-arrival time between successive non-zero periods using SES. $\hat{y}_{t+h|t}$ is then defined as forecasted mean of security vulnerabilities. This method basically decomposes the intermittent vulnerabilities into two parts: the number of non-zero vulnerabilities $\hat{z}_{t+h|t}$ and the time interval between those vulnerability periods $\hat{v}_{t+h|t}$, and then applies the single exponential smoothing on both parts. Croston's method uses only one weight parameter α, for both SES parts, therefore, $\hat{y}_{t+h|t}$, the estimate of mean non-zero vulnerabilities at time t, is defined as follows:

$$\hat{y}_{t+h|t} = \frac{\hat{z}_{t+h|t}}{\hat{v}_{t+h|t}} \tag{5}$$

$$\hat{z}_{t+h|t} = \begin{cases} z_t & \text{if } y_t = 0 \\ \alpha y_t + (1-\alpha) z_t & \text{if } y_t \neq 0 \end{cases}$$

$$\hat{v}_{t+h|t} = \begin{cases} v_t & \text{if } y_t = 0 \\ \alpha y_t + (1-\alpha) \hat{y}_t & \text{if } y_t \neq 0 \end{cases}$$

Croston's method is widely used in the intermittent demand forecasting and furthermore "*the standard method to be used in the industry nowadays, being implemented in many ERP systems and dedicated forecasting software*" (Petropoulos et al., 2016).

3.1.4. Neural Network

The last method, which makes use of neural networks (Nnet), is also particularly useful when dealing with zero-inflated time series. It has been used extensively to predict lumpy and intermittent demand and has shown good accuracy (Gutierrez et al., 2008; Kourentzes, 2013; Amin-Naseri and Tabar, 2008). We applied a feed-forward neural network with a single hidden layer. While J denotes the number of time series observations used as input p_j for the neural network, the number of forecasted security vulnerabilities $\hat{y}_{t+h|t}$ are defined as follows:

$$\hat{y}_{t+h|t} = \beta_0 + \sum_{i=1}^{I} \beta_i g \left(\gamma_{0j} + \sum_{j=1}^{J} \gamma_{ij} p_j \right) \tag{6}$$

where $\mathbf{w} = (\boldsymbol{\beta}, \boldsymbol{\gamma})$ are the weights of the network with $\boldsymbol{\beta} = [\beta_1, ..., \beta_I]$ and $\boldsymbol{\gamma} = [\gamma_{11}, ..., \gamma_{IJ}]$ for the output and the hidden layers respectively. The β_0 and γ_{0j} are the biases of each neuron, which function as the intercept in a regression for each neuron. I is the number of hidden nodes in the network and $g(\cdot)$ is a non-linear transfer function, which is in our case the sigmoid logistic function and provides the nonlinear capabilities to the model.

3.2. Accuracy Metrics

The literature on accuracy metrics can be divided into four types of forecasting error metrics (Hyndman et al., 2006): *Absolute metrics* such as the mean absolute error (MAE) or root mean square error (RMSE), *percentage-error metrics* such as the mean absolute percent error (MAPE) or mean arctangent absolute percentage error (MAAPE), *relative-error metrics*, which average the ratios of the errors from a designated method to the errors of a naive method (e.g., Median Relative Absolute Error (MdRAE)) and *scale-free error metrics*, which express each error as a ratio to an average error from a baseline method (Mean Absolute Scaled Error (MASE)).

From the above-mentioned accuracy metrics, percentage-error metrics, relative-error metrics and the mean absolute scaled error are not suitable for the following reasons: As we deal with zero-inflated time series, percentage-error metrics such as the MAPE are not well-defined, i.e. MAPE has the significant disadvantage that it produces infinite or undefined values for zero or close-to-zero actual values (Kim and Kim, 2016). Other percentage-error metrics which were developed for zero-inflated time series have some other drawbacks. For instance, although MAAPE is being designed for the purpose of intermittent demand forecasting (Kim and Kim, 2016), it lonely to interpret the forecasting accuracy seems not to be sufficient due to its definition drawback: Regardless the prediction, it maps every value to the worst value of $\frac{\pi}{2}$ when the actual value is zero ($y_t = 0$).

Relative-error metrics have similar shortcomings because it would involve division by zero and therefore not adaptable to zero-inflated time series as well (Hyndman et al., 2006). The fourth group of metrics, the mean absolute scaled error, is also not suitable in our context as we applied a rolling origin forecasting evaluation. Due to this, it is not usable in our context as the denominator becomes indefinite. To sum up, neither of these metrics is appropriate for zero-inflated time series because zero observations may yield division by zero problems (Syntetos and Boylan, 2005).

Therefore and in line with other studies (e.g., Arora and Taylor (2016); Taylor and Snyder (2012); Zhao et al. (2014)), in this study we use absolute forecast accuracy metrics due to the following reasons: First, both the mean absolute error (MAE) and the root mean square error (RMSE) can reflect the prediction accuracy of zero-inflated time series. Second, both accuracy metrics are widely

used in the forecasting literature and third, as absolute error metrics are calculated as a function of the forecast errors so that we can interpret the deviation in alignment with the structure of the time series. In the academic literature, a combination of metrics of MAE and RMSE is suggested to assess the model performance (Chai and Draxler, 2014). In the next subsection, we explain the MAE and the RMSE and in Subsection 3.2.3, we associate the accuracy metrics and time series structure in order to interpret the MAE and RMSE values.

3.2.1. Accuracy Metric: Mean Absolute Error

In order to capture the absolute forecasting error and to interpret our results, we assessed the Mean Absolute Error (MAE). The MAE is one of the most commonly used metric for evaluating the absolute error defined as the average of the absolute errors between the measured and predicted values (Gospodinov et al., 2006):

$$\mathrm{MAE} = \frac{1}{N} \sum_{t=1}^{N} \left(|y_t - \hat{y}_t| \right). \tag{7}$$

The MAE is a scale-dependent accuracy metric and uses the same scale as the data being measured (Hyndman et al., 2006). As our datasets contain only IT security vulnerabilities, we can compare the absolute forecast errors between the different versions of software and system application packages.

A value of 0 means a perfect forecast accuracy: All predicted values are equal to the real values. To give a sense for interpretability, we want to provide some examples for MAE as well with the same examples we used before for explaining MAE's values.

Let us assume that the number of actually published security vulnerabilities during a period t equals $y_t = 10$. Let us further assume that the number of predicted vulnerabilities equals $\hat{y}_t = 11$. As we have only one observation, the value of MAE would get a value of 1, which is close to its theoretical minimum of 0.

Let us now assume that the number of actually published security vulnerabilities during a period t equals $y_t = 5$. Let us further assume that the number of predicted vulnerabilities equals 100, i.e. $\hat{y}_t = 100$. As we have only one observation, the value of MAE would get a value of 95. However, regarding MAE, the value of 95 is not enough to explain the interpretability of MAE solely which we want to highlight with the following example: If the actually published security vulnerabilities during a period t had been $y_t = 10000$ and the predicted vulnerabilities equaled 10095, the MAE still would have been 95 but on a reasonable fit as we had only an overestimation of 0.95% while in the first scenario we had an overestimation of 95%. These examples show that for the interpretability of MAE, we have to associate the MAE value with the actual published security vulnerabilities as MAE is a sum of error terms $e_t \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for $t \in \mathbb{N}$ with $e_t \colon (|y_t - \hat{y}_t|)$.

3.2.2. Accuracy Metric: Root Mean Square Error

We further assessed the Root Mean Square Error (RMSE) in order to capture the absolute forecasting error and to interpret our results. The RSME is also one of the most commonly used metric for evaluating the absolute error and is defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N} (y_t - \hat{y_t})^2}{N}}. \tag{8}$$

The RMSE is similar to the MAE a scale-dependent accuracy metric and uses the same scale as the data being measured (Hyndman et al., 2006). However, they are more sensitive to outliers than MAE by definition (Hyndman and Koehler, 2006).

A value of 0 means a perfect forecast accuracy: All predicted values are equal to the real values. To give a sense for interpretability, we want to provide some similar examples for RMSE.

Let us assume that the number of actually published security vulnerabilities during a period t equals $y_t = 10$. Let us further assume that the number of predicted vulnerabilities equals $\hat{y}_t = 11$. As we have only one observation, the value of RMSE would get a value of 1, which is close to its theoretical minimum of 0.

Let us now assume that the number of actually published security vulnerabilities during a period t equals $y_t = 5$. Let us further assume that the number of predicted vulnerabilities equals 100, i.e. $\hat{y}_t = 100$. As we have only one observation, the value of RMSE would get a value of 9025. This simple example shows that such outliers have significant impacts on the RMSE's value. However, regarding RMSE, the value of 9025 is not enough to explain the interpretability of RMSE solely which we want to highlight with the following example: If the actually published security vulnerabilities during a period t had been very large such as $y_t = 1000000$ and the predicted vulnerabilities equaled 1000095, the RMSE still is 9025. However, comparing the RMSE of 9025 in the latter case, we have a very low overestimation close to zero while in the first scenario, we have an overestimation of 95%. These examples reveal that to interpret RMSE, we have to associate its value with the actual published security vulnerabilities as RMSE is a mapping of error terms $e_t \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for $t \in \mathbb{N}$ with $e_t \colon (y_t - \hat{y_t})^2$.

3.2.3. Accuracy Metrics and Time Series Structure

We explained in the Subsections 3.2.1 and 3.2.2 how MAE and RMSE is defined. Comparing both metrics, MAE is less sensitive to extreme values than RMSE (Li and Heap, 2011; Willmott, 1982; Willemain et al., 2004). When the differences between the MAE and RMSE are close to each other, it means that very large errors are unlikely to have occured (Li and Shi, 2010). The academic literature does not provide exact ranges for both the MAE and RMSE as acceptable values depend on the underlying context (Willmott and Matsuura, 2005) but in general low values close to the

theoretical minimum of zero are considered to be good Chaplot et al. (2000). We can use both the MAE and RMSE to give a sense of the interpretabilitiy and the relation of both accuracy metrics regarding the predicted and the actual values. Consider the following exemplary time series of vulnerabilities by assuming y the actual published and $\hat{y}$ the predicted vulnerabilities in the time frame $\{t = 1 \dots 6\}$:

Table 2: Example of Actual Published and Predicted Vulnerabilities.

t	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
y	0	0	1	0	5	0
$\hat{y}$	0	0	0	2	5	0

A closer look at the predicted values in this examples reveals that only in $t = 3$ and $t = 4$ we have a slight mismatch between the actual and the predicted values with one being underestimated ($y_3 = 1$ and $\hat{y}_3 = 0$) and an overestimation in t_4 with $y_4 = 0$ and $\hat{y}_4 = 2$. So, all in all, the forecasted values are good which is reflected in the value of MAE and RMSE. The computation shows that MAE is rather low with 0.5 and is close to its theoretical minimum. The RMSE is 0.83 is very low as well and close to its theoretical minimum. In this example, the mean of of the actual published vulnerabilities is 0.5 and the mean of the predicted vulnerabilities is 1.16: Comparing the means with the MAE and RMSE values, it shows that there is a good fit of the predicted vulnerabilities.

We can state that a low mean of actual published vulnerabilities over a wide time frame (e.g., 5 years) indicates that the time series contains a lot of zero values. Using MAE and RMSE assures us to reflect upon the prediction accuracy in a meaningful manner. A low MAE and RMSE close to the mean of the actual vulnerabilities shows that there is a good fit of the prediction method. On the other hand, a high MAE and RMSE which means that they are greater than the mean of the actual vulnerabilities, indicates that the deviation of the predicted vulnerabilities is high and the prediction accuracy rather poor.

3.3. Dataset: National Vulnerability Database

We select a dataset from the National Vulnerability Database (NVD)[4], which provides a comprehensive list of unique vulnerability and exposure data and maps it to corresponding system or software package (Martin, 2001). The NVD is a freely available US government data source maintained by the National Institute of Standards and Technology (NIST). Since its launch in 1997, it has reported standardized information about almost 80,000 software vulnerabilities. Although there do exist other security vulnerability databases, which are often community projects, such as

[4]The NVD-XML-Files are available at https://nvd.nist.gov/download.cfm.

Table 3: Description of the Software and System Package

Application Domain	Software / System Package	Release Date	Open Source
Browser	Mozilla Firefox	2002	Yes
Browser	Google Chrome	2008	Partially
Browser	Internet Explorer	1995	No
Browser	Safari	2003	No
Office	Microsoft Office	1990	No
Office	Thunderbird	2004	Yes
OS	Mac OS X	2001	No
OS	Ubuntu	2004	Yes
OS	Microsoft Windows	1985	No

Vulners (www.vulners.com), The Exploit Database (www.exploit-db.com), or Packet Storm's Vulnerability Database (www.packetstormsecurity.com), the NVD database still remains widely used and the most exhaustive resource for security vulnerability data. The dataset has been shown to be particularly useful for "*understanding trends and patterns in software vulnerabilities, so that one can better manage the security of computer systems that are pestered by the ubiquitous software security flaws*" (Zhang et al., 2011).

Table 3 shows a description of the application domains and corresponding software and system packages covered in our analysis.

Within these application domains, we analyze a balanced mix of closed source and open source software packages comprising the most prevalent software solutions in terms of market share. Our dataset covers the time period from January 2002 to June 2016. We further distinguish the system and software packages along their major version releases, since a package's version can serve as a reliable predictor for its vulnerability discovery rate (Alhazmi et al., 2005). We use the version numbers provided by the NVD database for each security vulnerability and group them by their major releases. Since the objective of our paper is to forecast recently appearing security vulnerabilities we focus on the root version of the software product where the vulnerability appeared first. Some vulnerabilities remain unpatched over multiple software versions and are therefore listed under multiple versions in the NVD dataset. Despite the fact that this total number, as reported on the NVD website, accurately reflects the number of vulnerabilities present in a specific software product and version, we filter for the number of uniquely originating vulnerabilities. Although this approach results in different sample sizes, we avoid aggregating multiple versions of a particular package to account for the individual vulnerability characteristics of each major version [5]. Finally,

[5] An exception to this are the Firefox versions starting from version 7 and Thunderbird versions since the versioning

the vulnerabilities were aggregated *per month* to generate an adequate dataset for our analysis.

4. Empirical Results and Discussion

We predicted the number of IT security vulnerabilities based on the forecasting methodologies implemented in the R package "forecast" (Hyndman, 2017). Figures 3 to 8 present the prediction accuracy (MAE) for nine software and system packages subdivided into the major versions for the forecasting horizon of three months [6].

4.1. Results

Since there is no substantial difference in forecasting accuracy between forecasting horizons of one or two months, we focus on the results of the longest forecasting horizon and provide complementary results for the other two time horizons in the appendix. Throughout the paper, forecasting accuracy (MAE and RMSE) is reported for the whole time frame available (cf. Appendix D) and three month forecasting horizons, unless stated otherwise. We show the performance of all six forecasting methods and compare different versions of the system and software packages. Figure 2 plots the time series and forecasts for Internet Explorer (Version 6) for the different forecasting methodologies as a representative example [7]. The figures display the characteristics of the time series with regards to its volatility and many zero values (rareness of occurrence). The plots furthermore show that depending on the forecasting methodology, the difference between the predicted values and the actual values varies considerably. While SES and Croston's method produce smooth predictions with low variability, the other methods rapidly adapt to variations of the time series which has impacts on the prediction accuracy.

of these products does not reflect major changes in steps from one version to another. We labeled these as "rolling versions".

[6] Note that the time frame of "three months" (h=3) means that vulnerabilities were summed up quarterly. The prediction pertains to the next quarter.

[7] As we have a total of 270 different software and system packages and versions, we only provide one representative example here. The other plots can be obtained from the authors.

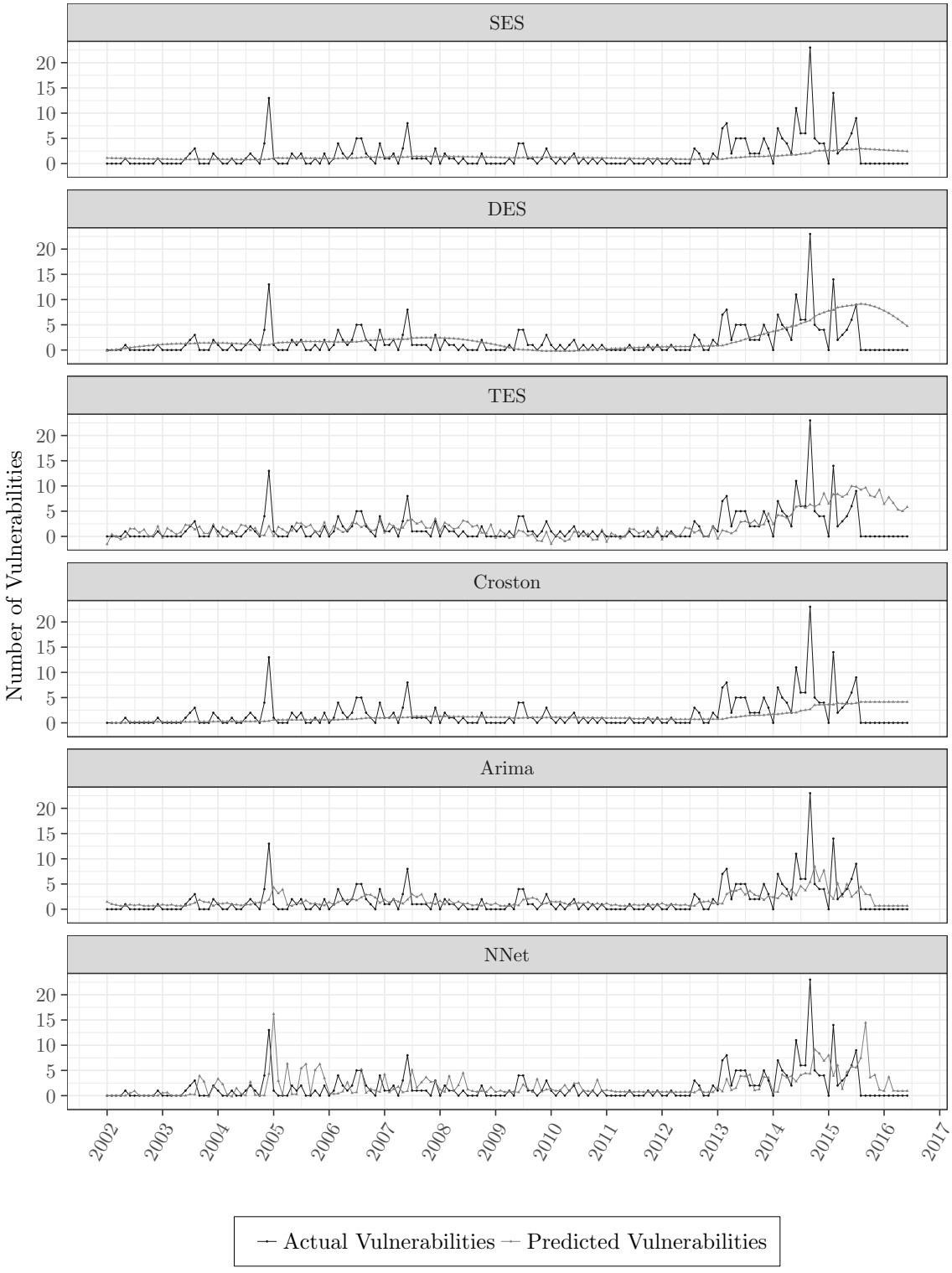

Figure 2: Time Series and Forecasts for Internet Explorer (v6)

The Figures 3 to 8 show the forecasting accuracy in terms of MAE and RMSE. We observe that the forecasting accuracy varies depending on the forecasting methodology.

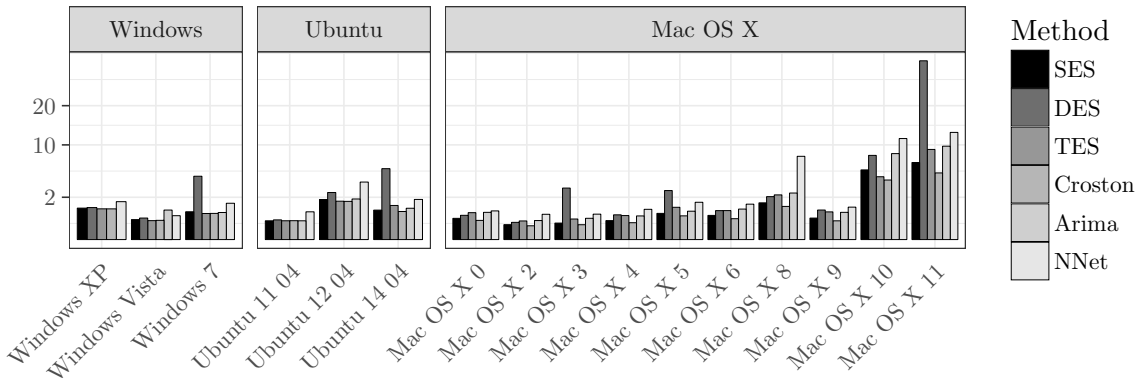

Figure 3: Prediction Accuracy (MAE) for Operating Systems, h=3 (months)

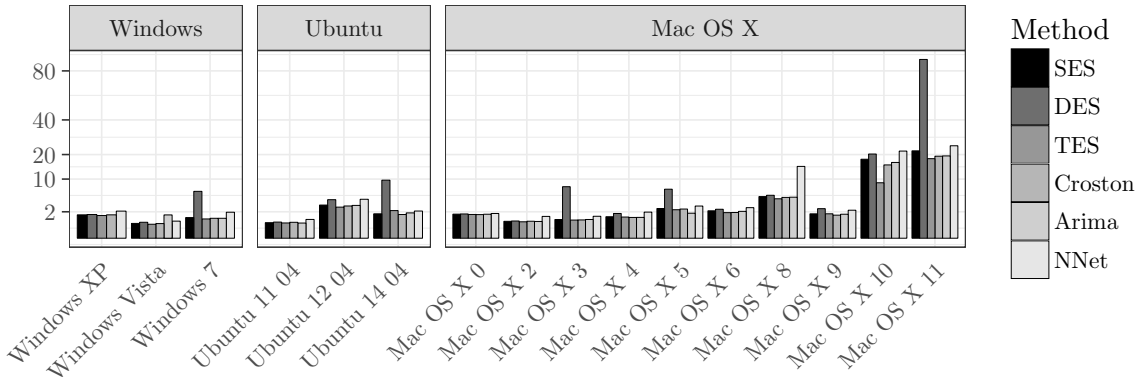

Figure 4: Prediction Accuracy (RMSE) for Operating Systems, h=3 (months)

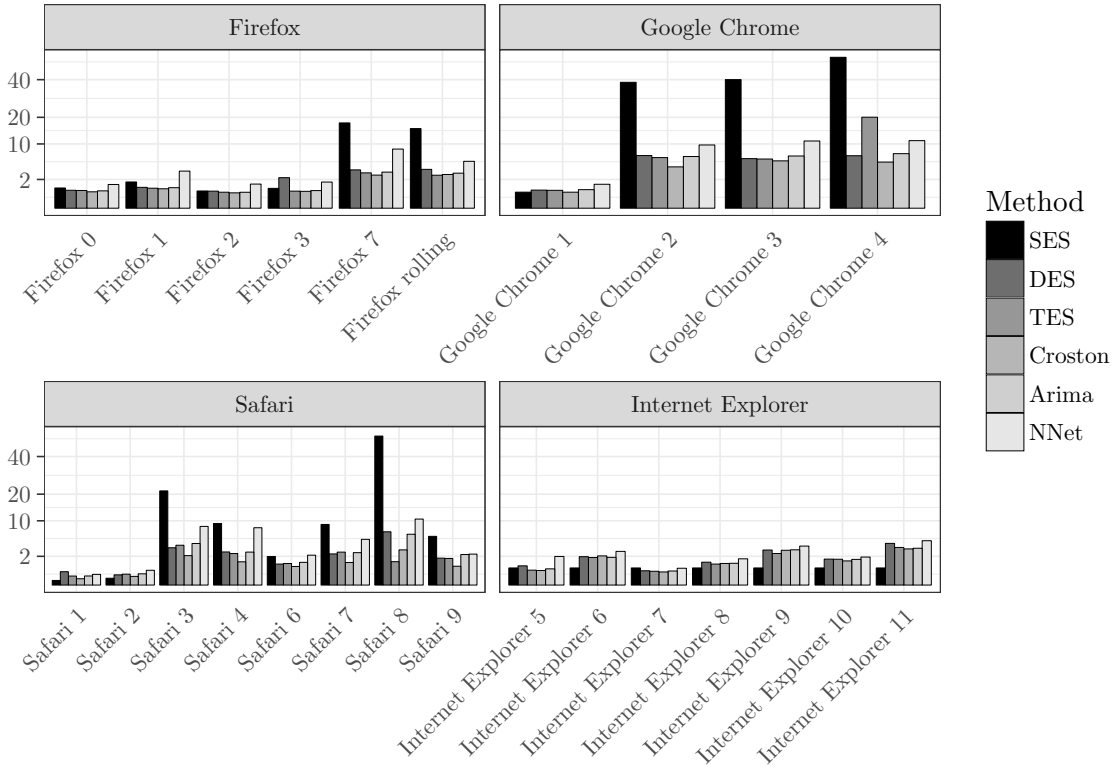

Figure 5: Prediction Accuracy (MAE) for Browsers, h=3 (months)

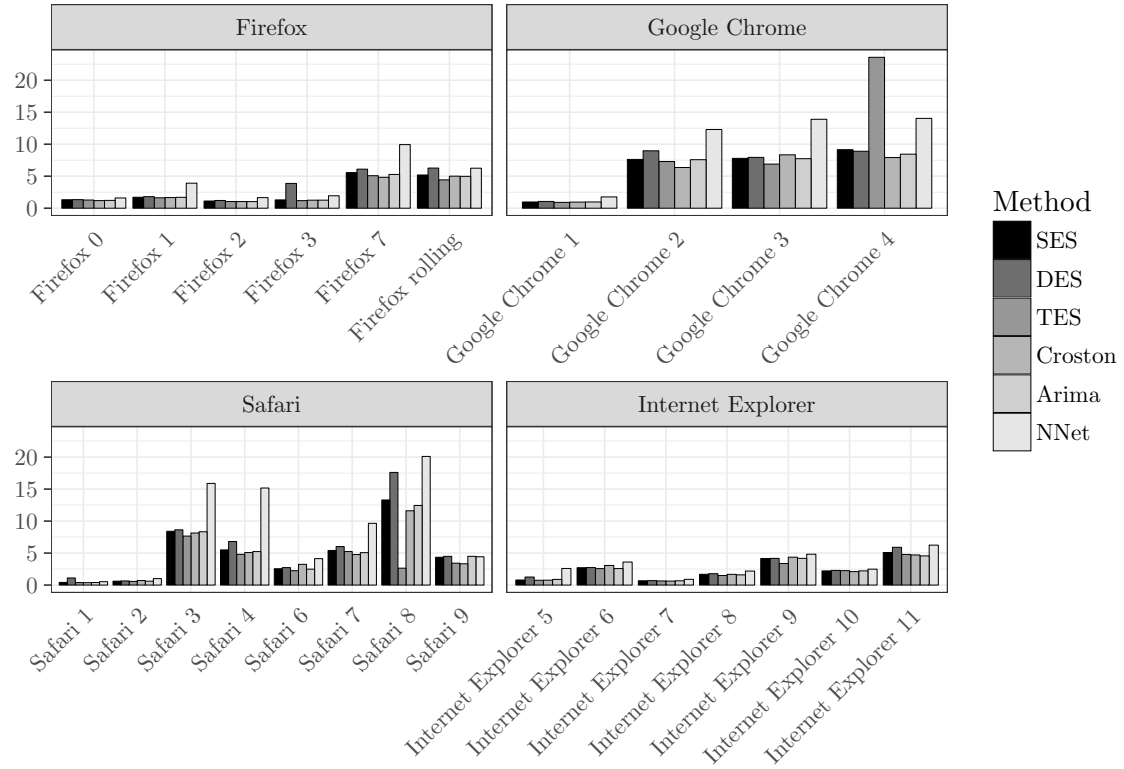

Figure 6: Prediction Accuracy (RMSE) for Browsers, h=3 (months)

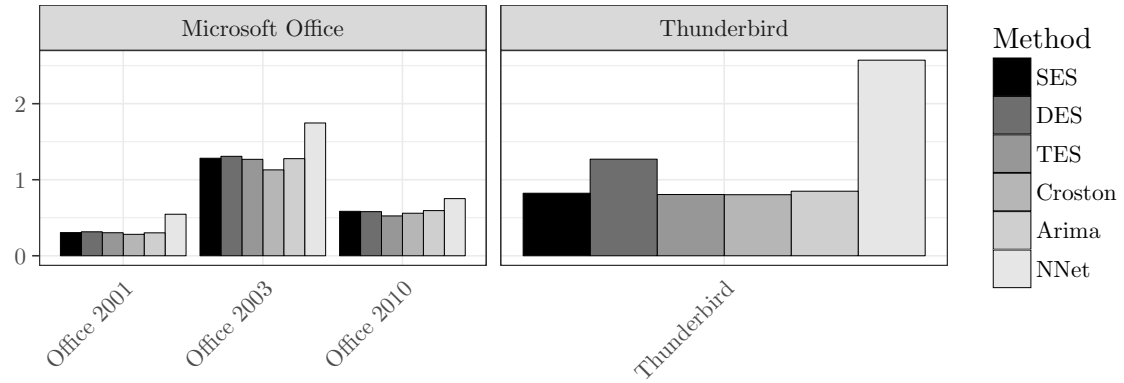

Figure 7: MAE for Office Solutions, h=3 (months)

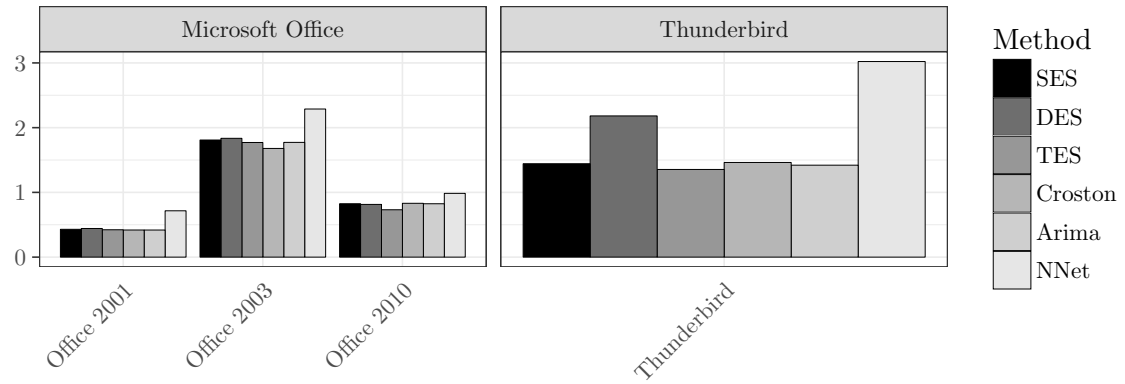

Figure 8: Prediction Accuracy (RMSE) for Office Solutions, h=3 (months)

4.2. Discussion

We focus our discussion on how the accuracy of the forecasting methods is affected by the software and system packages and we consider the robustness of our results with regard to different error metrics. Our discussion thereby contributes to the rising stream of literature analyzing IT security vulnerabilities from a time series perspective and it examines how the prediction accuracy of IT security vulnerabilities' time series is impacted by the applied forecasting methodology.

4.2.1. Prediction Accuracy Depending on Software and System Packages

With the variety of software and systems covered in our study, we have to examine how robust our results are regarding different packages. We make a few observations: First, there is a tendency, especially for the software package "browsers", that the forecasting methodology SES and Neural Network are less suitable (cf. Fig. 5 and Fig. A.9 and A.12). In the case of Neural Networks, this tendency applies to all packages. The reason for the poor results of SES and Neural Networks lies in the properties of security vulnerability time series data, rareness of occurrence and a tendency towards outliers. Both characteristics have an impact on the effectiveness of the applied forecasting methodology. In the intermittent demand literature, this phenomenon is widely known. The infrequent demand arrivals coupled with variable demand sizes whenever demand occurs render the problem of accurately estimating the demand especially challenging (Petropoulos and Kourentzes, 2015). Translating this to our context, this means that the sudden appearance of vulnerabilities for a few periods results in an overestimation of vulnerabilities in the following periods of time, i.e., vulnerabilities are predicted even if there were none. This explains the poor performance of these two methods.

Second, from an intra-related observation between the different packages, it is evident that for some methods, the prediction accuracy varies considerably. For example, within browsers, DES

together with SES and Neural Networks have achieved significantly poorer prediction accuracies for Mac OS X (cf. Fig. 5) or TES with SES and Neural Networks for some versions of Google Chrome (cf. Fig. 6). In general, we observe that the accuracy of the forecasting methodologies depends on the applied software and system packages, and we note that approximately more or less the same prediction accuracy being close to its theoretical minimum for both metrics, except for SES and Neural Networks (cf. Fig. 3 to Fig. 8 and Fig. A.9 to Fig. A.14). These observations have two implications: (1) We can state that the choice of a forecasting methodology depends on the software or system package as some methods are not suitable such as SES and Neural Networks, and (2) from a managerial point of view, the tendency of low prediction errors offers decision-makers a good choice to use these forecasting methodologies in order to anticipate the development of IT security vulnerabilities of their software and system applications in their organization's portfolio.

4.2.2. Robustness of Different Measures of Prediction Accuracy

Another issue deserving attention is the robustness of our results in terms of the used forecasting-error metrics MAE and RMSE. For instance, our discussion in the prior subsection revealed the poor performance of SES and Neural Networks. The crucial question is how to interpret these values and the robustness of the prediction accuracy within the two applied forecasting-error metrics.

We can observe that the poor performance of SES and Neural Networks is independent from the applied metrics MAE and RMSE - both metrics show the same tendency. For most of the cases, in the light of our discussion in the prior subsection, we observe that the values of MAE are close to zero meaning that the prediction accuracy was high. As RMSE is more sensitive to outliers (cf. Subsection 3.2.2), its values are higher but the overall tendency for the forecasting methodologies is the same: The actual absolute prediction accuracy for MAE as well as for RMSE was low in most of the cases (cf. Tables C.4 to C.6) with the exceptions of SES and Neural Networks for browsers. The tendency of the other methodologies shows that the forecasting methodologies' decomposition of the zero valued intervals is sufficient to capture the high volatility. Important implications from this result are that that the prediction error is independent from the applied metrics and the prediction accuracy was good for the forecasting methodologies despite dealing with time series that contain many zero values. This is backed up by Table D.7 in which the actual vulnerabilities within the time frame and the monthly averages are shown. Regarding the latter implication, a closer look to Safari v1 reveals that within the time of 13 years, 20 vulnerabilities occurred implying that this time series contains a lot of zero values and, if vulnerabilities appear, they are volatile.

Regarding the robustness of our results, we observe a same tendency for both forecasting error metrics which suggests that the outcome of the accuracy of a particular forecasting methodology is independent from the choice of the metric. There are only slight differences in the metrics: Using MAE, we get values closer to zero for almost every forecasting methodology which means it is

less sensitive to extreme values than RMSE. While RMSE estimation is based on the mean and is more sensitive to extreme cases and outliers, the MAE estimation is based on the median and is therefore more robust to outliers. In some cases using SES, we observe that RMSE dropped but MAE increased. This means that these metrics are better at accounting for extreme cases, but the solution is less robust. However, Tables 3 to 8 show only slight variations between MAE and RMSE for the different forecasting methods. When the MAE and RMSE are close to each other, it means that very large errors are unlikely to have occurred.

To conclude, we can derive the implications that (1) the metrics MAE and RMSE can measure the actual prediction error accurately in the context of IT security vulnerabilities and (2) the accuracy results of the forecasting methodologies are robust in terms of the independence from the applied metrics.

5. Conclusion

This paper addresses the problem of forecasting the number of IT security vulnerabilities of different system and software packages including operating systems, browsers and office solutions. The analysis of vulnerabilities with time series methods is a rising stream in the literature to which our study contributes an extensive analysis of forecasting methodologies. We review the pros and cons of forecasting error metrics and demonstrate the appropriateness of the absolute error forecasting metric. Using the metrics MAE and RMSE, we discussed the forecasting accuracy based on the robustness factors software and system packages and highlight the independence of the accuracy results from the metrics.

Our study reveals important implications: First, the selection of a forecasting methodology depends on the software or system package as some methods show poor performances (such as SES and Neural Networks). Second, our results are relevant to managerial decision makers as they demonstrate the accuracy of IT security vulnerability forecasts, which can inform critical decisions on organizational software portfolios. Third, we were able to show that absolute metrics overcome disadvantages of other, e.g., percentage-error metrics and that absolute metrics can cover the actual prediction error precisely in the context of IT security vulnerabilities. Fourth, we could show that the accuracy results of the forecasting methodologies are robust in terms of the independence from using absolute metrics.

Our study has a few limitations: Although we followed a structured and accurate search process to identify IT security vulnerabilities from the most extensive database NVD and linked the vulnerabilities uniquely to the corresponding root software and system package, we may have missed vulnerabilities or missed vulnerabilities which are not included in the NVD database. Further, we could only plot the time series and forecasts for Internet Explorer (Version 6) for the different forecasting methodologies as a representative example as we have a total of 270 different software and

system packages and versions.

To conclude, we hope that our study encourages academics to develop suitable forecasting methods for IT security vulnerabilities which subsequently improve prediction accuracy.

555

Acknowledgments. The research leading to these results was supported by the Hanns-Seidel foundation (HSS) (`https://www.hss.de/en/`) and by the "Bavarian State Ministry for Education, Science and the Arts" as part of the FORSEC research association (`https://www.bayforsec.de/en/start/`).

Appendix A. MAE Results for 1 and 2 Months

Appendix A.1. MAE Results for 1 Month

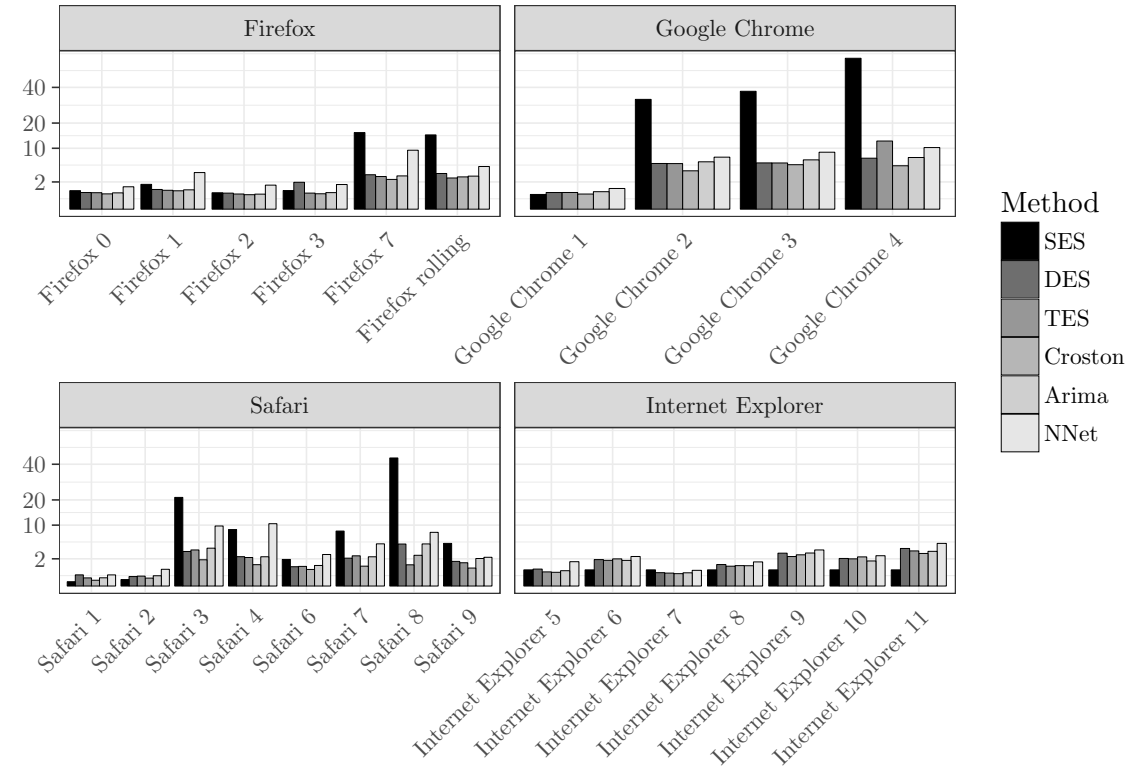

Figure A.9: Prediction Accuracy (MAE) for Browsers, h=1 (month)

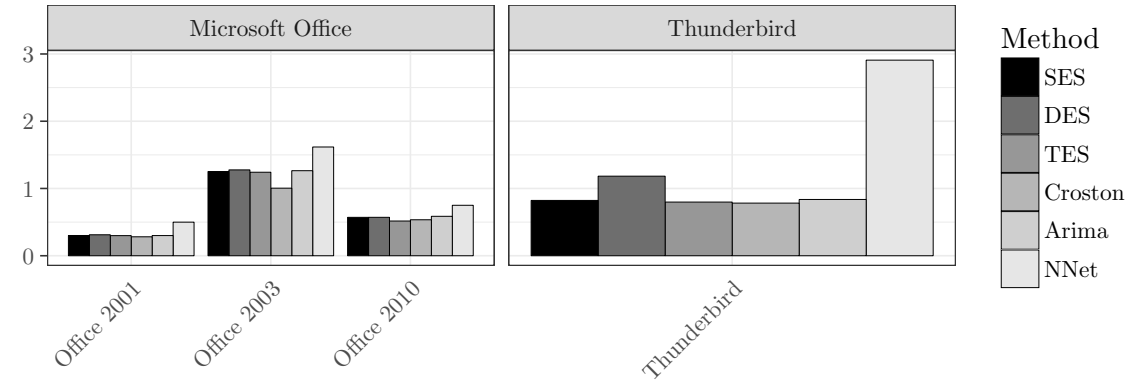

Figure A.10: Prediction Accuracy (MAE) for Office Solutions, h=1 (month)

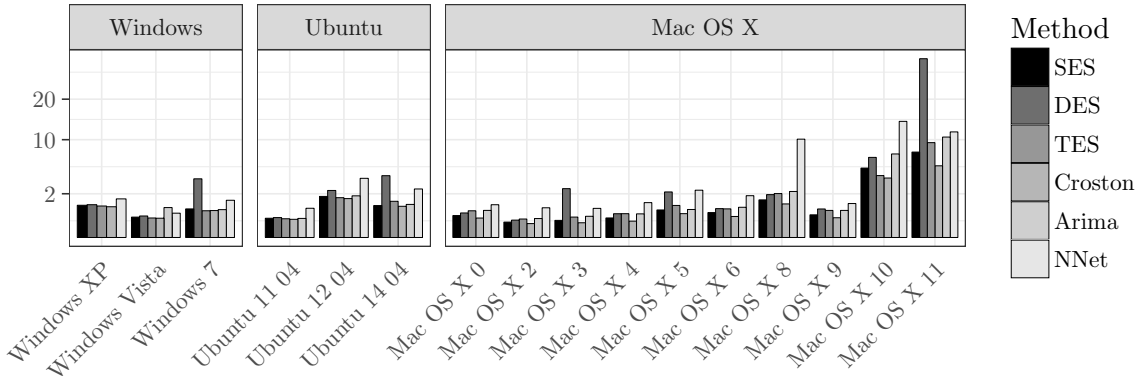

Figure A.11: Prediction Accuracy (MAE) for Operating Systems, h=1 (month)

Appendix A.2. MAE Results for 2 Months

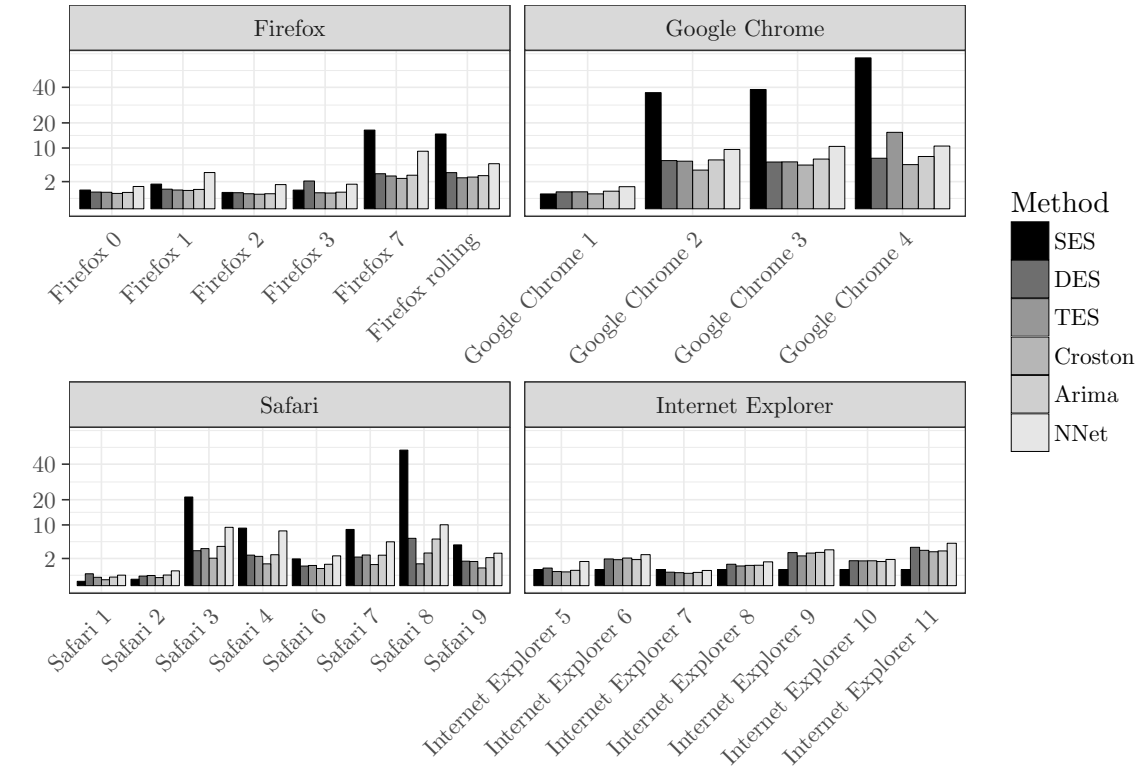

Figure A.12: Prediction Accuracy (MAE) for Browsers, h=2 (months)

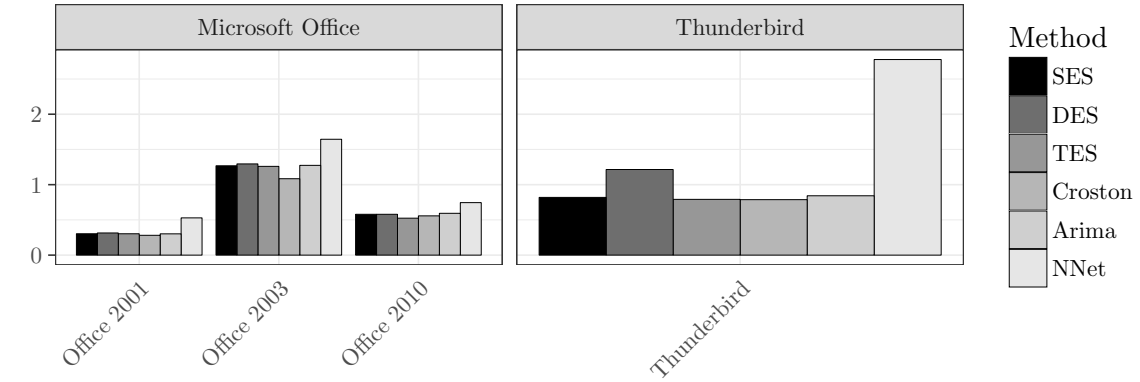

Figure A.13: Prediction Accuracy (MAE) for Office Solutions, h=2 (months)

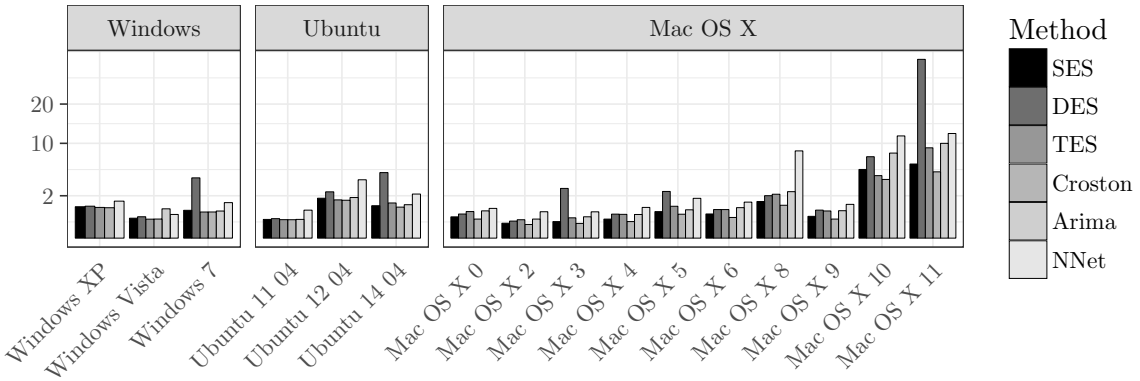

Figure A.14: Prediction Accuracy (MAE) for Operating Systems, h=2 (months)

Appendix B. RMSE Results for 1 and 2 Months

Appendix B.1. RMSE Results for 1 Month

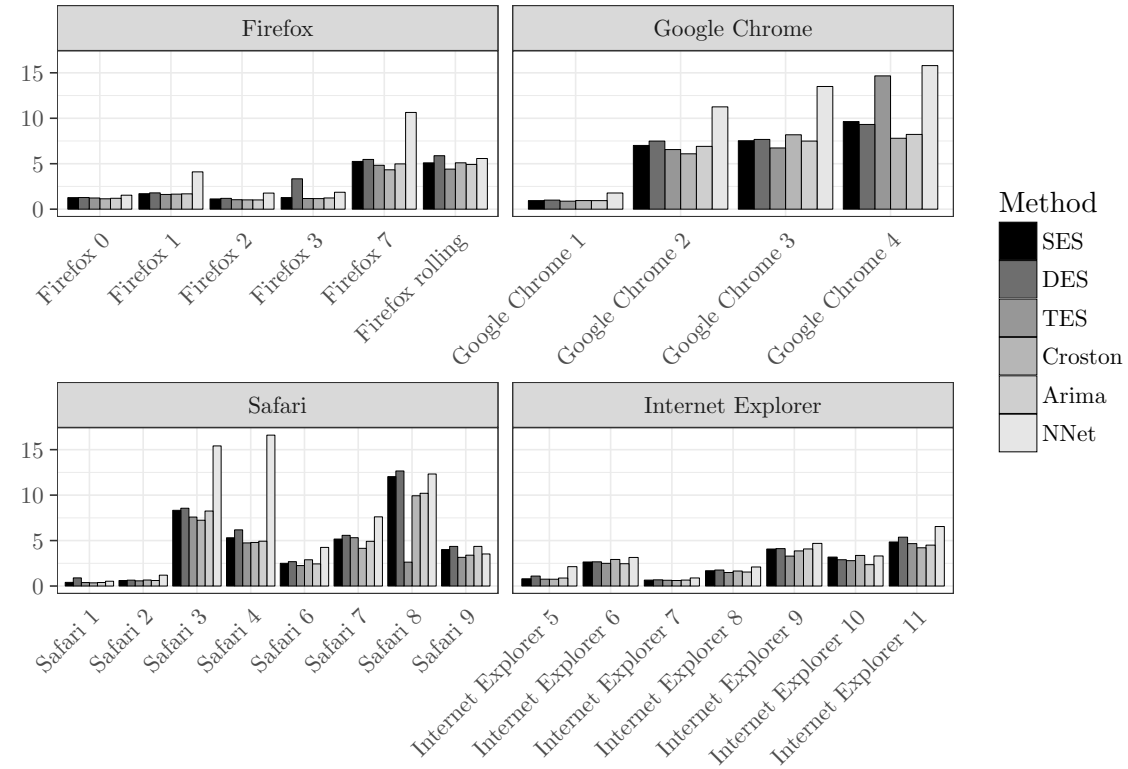

Figure B.15: Prediction Accuracy (RMSE) for Browsers, h=1 (month)

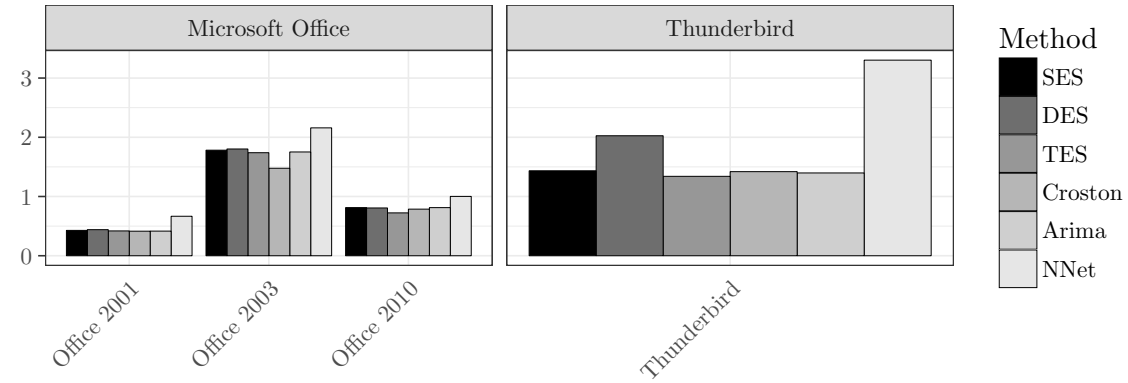

Figure B.16: Prediction Accuracy (RMSE) for Office Solutions, h=1 (month)

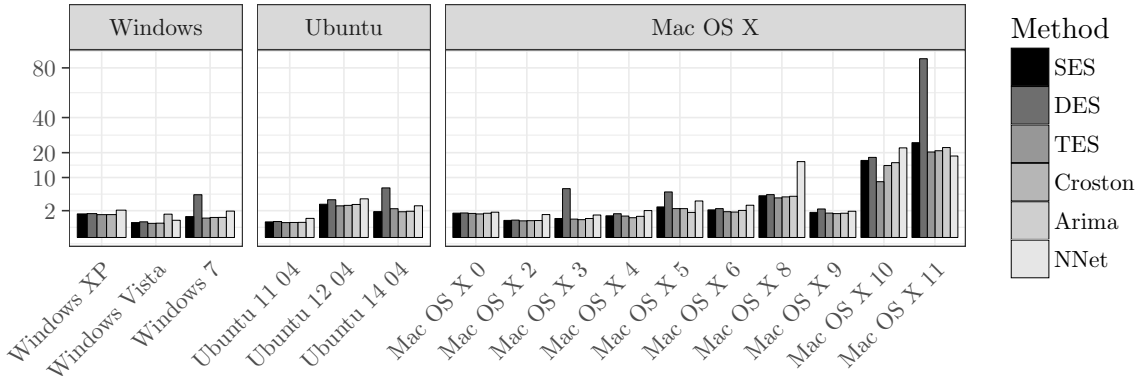

Figure B.17: Prediction Accuracy (RMSE) for Operating Systems, h=1 (month)

Appendix B.2. RMSE Results for 2 Months

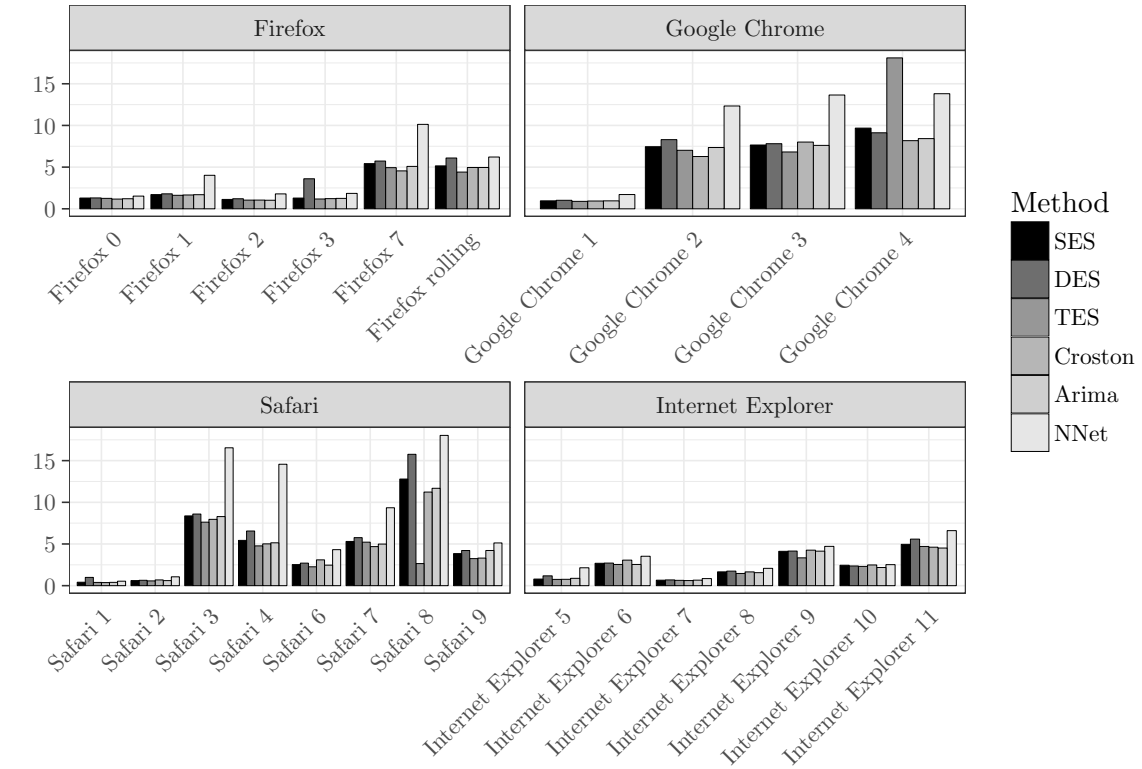

Figure B.18: Prediction Accuracy (RMSE) for Browsers, h=2 (months)

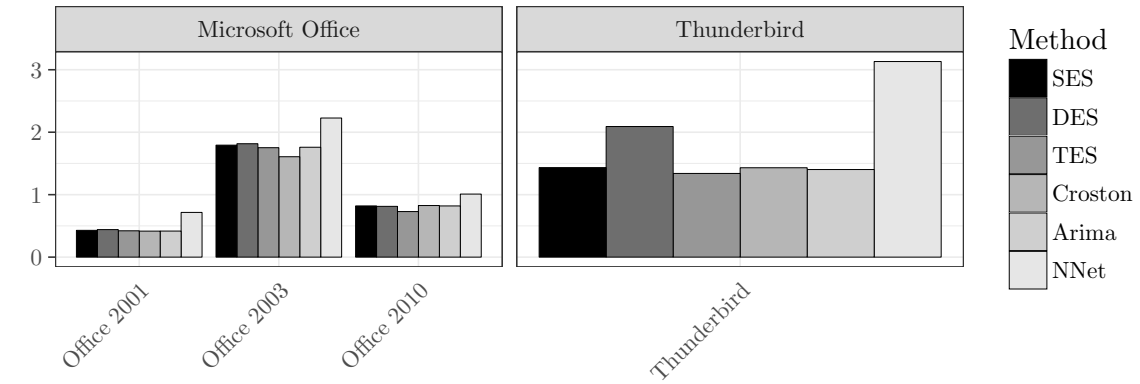

Figure B.19: Prediction Accuracy (RMSE) for Office Solutions, h=2 (months)

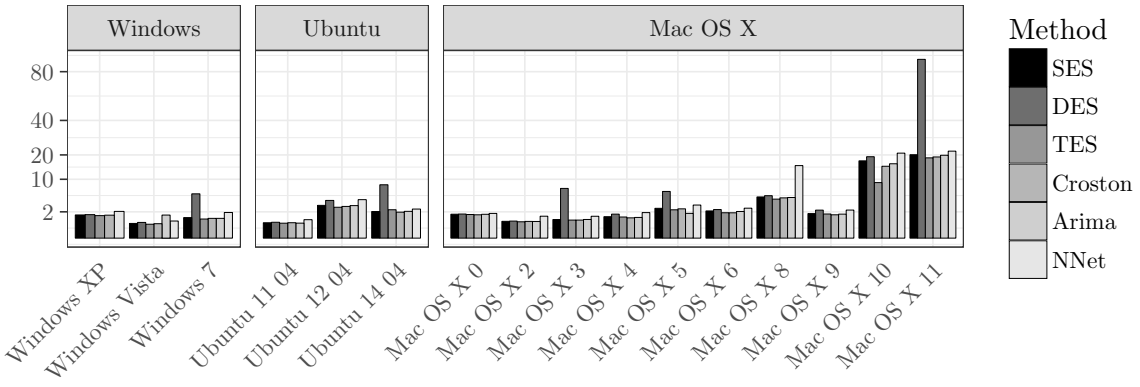

Figure B.20: Prediction Accuracy (RMSE) for Operating Systems, h=2 (months)

Appendix C. Parameters for Methods and Software / System Packages

Appendix C.1. Forecasting Results for Methods and Software / System Packages (Mean Absolute Error / Root Mean Squared Error)

Table C.4: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 1 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
Firefox Versions						
v0	(0.92 / 1.27)	(0.75 / 1.28)	(0.73 / 1.23)	(0.72 / 1.2)	(0.63 / 1.14)	(1.35 / 1.54)
v1	(1.66 / 1.7)	(1.05 / 1.79)	(0.96 / 1.62)	(1.01 / 1.68)	(0.91 / 1.65)	(3.61 / 4.1)
v2	(0.72 / 1.12)	(0.69 / 1.19)	(0.62 / 1.04)	(0.61 / 1.02)	(0.56 / 1.02)	(1.56 / 1.77)
v3	(0.93 / 1.27)	(1.95 / 3.34)	(0.7 / 1.17)	(0.74 / 1.23)	(0.64 / 1.16)	(1.64 / 1.86)
v7	(15.81 / 5.25)	(3.2 / 5.47)	(2.87 / 4.83)	(2.98 / 4.98)	(2.39 / 4.33)	(9.37 / 10.64)
v8 rolling	(14.86 / 5.09)	(3.43 / 5.88)	(2.62 / 4.4)	(2.95 / 4.92)	(2.81 / 5.1)	(4.91 / 5.57)
Google Chrome Versions						
v1	(0.58 / 0.94)	(0.75 / 1)	(0.75 / 0.88)	(0.82 / 0.94)	(0.62 / 0.95)	(1.15 / 1.78)
v2	(32.52 / 7.01)	(5.63 / 7.49)	(5.6 / 6.56)	(6.05 / 6.91)	(3.97 / 6.09)	(7.3 / 11.25)
v3	(37.48 / 7.53)	(5.77 / 7.67)	(5.75 / 6.73)	(6.56 / 7.49)	(5.33 / 8.18)	(8.75 / 13.5)
v4	(61.42 / 9.63)	(7.01 / 9.32)	(12.53 / 14.66)	(7.2 / 8.22)	(5.08 / 7.8)	(10.24 / 15.79)
Internet Explorer Versions						
v5	(0.7 / 0.79)	(0.78 / 1.1)	(0.54 / 0.75)	(0.63 / 0.88)	(0.51 / 0.75)	(1.6 / 2.13)
v6	(0.7 / 2.64)	(1.89 / 2.66)	(1.78 / 2.5)	(1.77 / 2.46)	(1.99 / 2.92)	(2.35 / 3.14)
v7	(0.7 / 0.65)	(0.48 / 0.68)	(0.45 / 0.63)	(0.47 / 0.66)	(0.41 / 0.61)	(0.67 / 0.89)
v8	(0.7 / 1.68)	(1.25 / 1.76)	(1.06 / 1.49)	(1.11 / 1.54)	(1.12 / 1.65)	(1.57 / 2.1)
v9	(0.7 / 4.07)	(2.91 / 4.11)	(2.35 / 3.29)	(2.94 / 4.08)	(2.63 / 3.86)	(3.51 / 4.69)
v10	(0.7 / 3.19)	(2.05 / 2.89)	(1.99 / 2.79)	(1.7 / 2.35)	(2.29 / 3.37)	(2.48 / 3.31)
v11	(0.7 / 4.84)	(3.81 / 5.37)	(3.33 / 4.66)	(3.25 / 4.5)	(2.86 / 4.21)	(4.9 / 6.55)

Table C.4: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 1 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
Mac OS X Versions						
v0	(0.5 / 1.65)	(0.63 / 1.67)	(0.75 / 1.61)	(0.77 / 1.64)	(0.4 / 1.54)	(1.12 / 1.78)
v2	(0.25 / 0.81)	(0.32 / 0.84)	(0.36 / 0.77)	(0.38 / 0.8)	(0.2 / 0.79)	(0.92 / 1.46)
v3	(0.31 / 1)	(2.49 / 6.63)	(0.44 / 0.94)	(0.47 / 1)	(0.23 / 0.88)	(0.89 / 1.4)
v4	(0.4 / 1.31)	(0.59 / 1.57)	(0.59 / 1.28)	(0.58 / 1.24)	(0.28 / 1.09)	(1.27 / 2.01)
v5	(0.79 / 2.6)	(2.17 / 5.77)	(1.07 / 2.32)	(0.82 / 1.75)	(0.59 / 2.32)	(2.34 / 3.7)
v6	(0.65 / 2.12)	(0.87 / 2.32)	(0.86 / 1.85)	(0.96 / 2.04)	(0.46 / 1.8)	(1.83 / 2.9)
v8	(1.48 / 4.84)	(1.92 / 5.11)	(2.02 / 4.36)	(2.21 / 4.72)	(1.18 / 4.58)	(10.11 / 16.01)
v9	(0.54 / 1.75)	(0.85 / 2.26)	(0.77 / 1.66)	(0.77 / 1.64)	(0.41 / 1.59)	(1.21 / 1.91)
v10	(5.04 / 16.5)	(6.71 / 17.85)	(4.01 / 8.66)	(7.3 / 15.56)	(3.7 / 14.4)	(14.1 / 22.34)
v11	(7.63 / 24.98)	(33.42 / 88.88)	(9.41 / 20.32)	(10.54 / 22.49)	(5.37 / 20.93)	(11.66 / 18.47)
Microsoft Office Versions						
Office 2001	(0.3 / 0.43)	(0.31 / 0.44)	(0.3 / 0.42)	(0.3 / 0.42)	(0.28 / 0.41)	(0.5 / 0.67)
Office 2003	(1.25 / 1.78)	(1.28 / 1.8)	(1.24 / 1.74)	(1.26 / 1.75)	(1 / 1.48)	(1.62 / 2.16)
Office 2010	(0.57 / 0.81)	(0.57 / 0.81)	(0.52 / 0.72)	(0.59 / 0.81)	(0.53 / 0.79)	(0.75 / 1)
Safari Versions						
v1	(0.05 / 0.4)	(0.34 / 0.89)	(0.18 / 0.38)	(0.18 / 0.39)	(0.09 / 0.35)	(0.33 / 0.53)
v2	(0.11 / 0.61)	(0.24 / 0.65)	(0.26 / 0.57)	(0.29 / 0.61)	(0.17 / 0.66)	(0.76 / 1.2)
v3	(21.19 / 8.33)	(3.22 / 8.56)	(3.51 / 7.59)	(3.87 / 8.25)	(1.86 / 7.24)	(9.73 / 15.41)
v4	(8.6 / 5.31)	(2.32 / 6.18)	(2.2 / 4.74)	(2.31 / 4.93)	(1.23 / 4.79)	(10.48 / 16.6)
v6	(1.91 / 2.5)	(1.01 / 2.68)	(1.04 / 2.25)	(1.14 / 2.43)	(0.74 / 2.89)	(2.68 / 4.25)

Table C.4: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 1 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
v7	(8.15 / 5.17)	(2.1 / 5.58)	(2.46 / 5.32)	(2.31 / 4.92)	(1.06 / 4.15)	(4.8 / 7.61)
v8	(44.23 / 12.04)	(4.76 / 12.66)	(1.22 / 2.63)	(4.78 / 10.2)	(2.55 / 9.93)	(7.78 / 12.32)
v9	(4.91 / 4.01)	(1.64 / 4.36)	(1.46 / 3.16)	(2.04 / 4.36)	(0.87 / 3.39)	(2.23 / 3.53)
Thunderbird Versions						
rolling	(0.82 / 1.43)	(1.18 / 2.03)	(0.8 / 1.34)	(0.84 / 1.4)	(0.78 / 1.42)	(2.91 / 3.3)
Ubuntu Versions						
v11.04	(0.39 / 0.67)	(0.41 / 0.71)	(0.37 / 0.62)	(0.38 / 0.64)	(0.35 / 0.63)	(0.89 / 1.02)
v12.04	(1.76 / 3.08)	(2.31 / 3.96)	(1.66 / 2.78)	(1.82 / 3.03)	(1.58 / 2.87)	(3.66 / 4.16)
v14.04	(1.07 / 1.86)	(3.99 / 6.84)	(1.37 / 2.3)	(1.14 / 1.91)	(1.02 / 1.84)	(2.46 / 2.79)
Windows Versions						
Windows XP	(1.09 / 1.54)	(1.12 / 1.59)	(1.04 / 1.45)	(1.08 / 1.52)	(0.99 / 1.45)	(1.56 / 2.08)
Windows Vista	(0.44 / 0.62)	(0.48 / 0.68)	(0.4 / 0.56)	(0.43 / 0.6)	(0.39 / 0.57)	(0.62 / 0.83)
Windows 7	(0.85 / 1.21)	(3.6 / 5.08)	(0.75 / 1.05)	(0.81 / 1.12)	(0.76 / 1.12)	(1.45 / 1.94)

Table C.5: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 2 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
Firefox Versions						
v0	(0.95 / 1.29)	(0.76 / 1.3)	(0.74 / 1.25)	(0.72 / 1.21)	(0.64 / 1.16)	(1.35 / 1.52)
v1	(1.65 / 1.7)	(1.04 / 1.79)	(0.96 / 1.62)	(1.02 / 1.69)	(0.91 / 1.66)	(3.56 / 4.02)
v2	(0.72 / 1.13)	(0.7 / 1.21)	(0.61 / 1.04)	(0.62 / 1.02)	(0.57 / 1.04)	(1.59 / 1.79)
v3	(0.95 / 1.29)	(2.09 / 3.6)	(0.69 / 1.18)	(0.75 / 1.25)	(0.67 / 1.22)	(1.64 / 1.85)
v7	(16.79 / 5.42)	(3.33 / 5.73)	(2.92 / 4.94)	(3.06 / 5.09)	(2.5 / 4.55)	(8.98 / 10.13)
v8 rolling	(15.15 / 5.15)	(3.54 / 6.09)	(2.6 / 4.4)	(2.98 / 4.95)	(2.73 / 4.96)	(5.51 / 6.21)
Google Chrome Versions						
v1	(0.6 / 0.95)	(0.77 / 1.02)	(0.78 / 0.89)	(0.84 / 0.95)	(0.6 / 0.93)	(1.32 / 1.71)
v2	(36.55 / 7.45)	(6.3 / 8.29)	(6.14 / 7.02)	(6.48 / 7.35)	(4.06 / 6.27)	(9.53 / 12.34)
v3	(38.57 / 7.65)	(5.93 / 7.81)	(5.96 / 6.82)	(6.7 / 7.61)	(5.18 / 8.01)	(10.55 / 13.65)
v4	(61.7 / 9.68)	(6.92 / 9.12)	(15.83 / 18.1)	(7.41 / 8.42)	(5.29 / 8.17)	(10.67 / 13.81)
Internet Explorer Versions						
v5	(0.71 / 0.79)	(0.84 / 1.17)	(0.54 / 0.75)	(0.64 / 0.88)	(0.51 / 0.76)	(1.58 / 2.14)
v6	(0.71 / 2.68)	(1.93 / 2.71)	(1.83 / 2.55)	(1.84 / 2.54)	(2.07 / 3.06)	(2.61 / 3.53)
v7	(0.71 / 0.66)	(0.49 / 0.69)	(0.46 / 0.64)	(0.48 / 0.66)	(0.41 / 0.61)	(0.63 / 0.85)
v8	(0.71 / 1.66)	(1.24 / 1.74)	(1.05 / 1.46)	(1.13 / 1.56)	(1.11 / 1.65)	(1.53 / 2.08)
v9	(0.71 / 4.12)	(2.96 / 4.15)	(2.4 / 3.34)	(3 / 4.14)	(2.87 / 4.26)	(3.48 / 4.71)
v10	(0.71 / 2.45)	(1.68 / 2.36)	(1.66 / 2.31)	(1.58 / 2.18)	(1.68 / 2.48)	(1.86 / 2.52)
v11	(0.71 / 4.96)	(3.98 / 5.58)	(3.38 / 4.7)	(3.27 / 4.51)	(3.11 / 4.62)	(4.87 / 6.6)

Table C.5: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 2 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
Mac OS X Versions						
v0	(0.5 / 1.65)	(0.65 / 1.68)	(0.79 / 1.61)	(0.83 / 1.64)	(0.4 / 1.57)	(0.98 / 1.76)
v2	(0.25 / 0.81)	(0.32 / 0.84)	(0.38 / 0.77)	(0.4 / 0.8)	(0.21 / 0.81)	(0.77 / 1.39)
v3	(0.3 / 1)	(2.75 / 7.14)	(0.46 / 0.94)	(0.51 / 1.01)	(0.24 / 0.94)	(0.77 / 1.38)
v4	(0.4 / 1.33)	(0.64 / 1.66)	(0.63 / 1.29)	(0.62 / 1.24)	(0.31 / 1.19)	(1.05 / 1.88)
v5	(0.78 / 2.57)	(2.43 / 6.29)	(1.13 / 2.32)	(0.89 / 1.77)	(0.64 / 2.48)	(1.76 / 3.16)
v6	(0.65 / 2.13)	(0.91 / 2.36)	(0.91 / 1.86)	(1.03 / 2.05)	(0.48 / 1.85)	(1.44 / 2.59)
v8	(1.49 / 4.9)	(2 / 5.2)	(2.14 / 4.39)	(2.4 / 4.77)	(1.2 / 4.68)	(8.47 / 15.19)
v9	(0.53 / 1.74)	(0.87 / 2.27)	(0.82 / 1.68)	(0.84 / 1.66)	(0.4 / 1.57)	(1.27 / 2.28)
v10	(5.25 / 17.24)	(7.38 / 19.14)	(4.35 / 8.9)	(8.05 / 15.97)	(3.83 / 14.91)	(11.63 / 20.85)
v11	(6.11 / 20.08)	(35.57 / 92.3)	(9.08 / 18.6)	(9.99 / 19.83)	(4.89 / 19.03)	(12.18 / 21.85)
Microsoft Office Versions						
Office 2001	(0.3 / 0.43)	(0.31 / 0.44)	(0.3 / 0.42)	(0.3 / 0.42)	(0.28 / 0.42)	(0.53 / 0.72)
Office 2003	(1.27 / 1.79)	(1.29 / 1.82)	(1.26 / 1.75)	(1.27 / 1.76)	(1.08 / 1.61)	(1.64 / 2.23)
Office 2010	(0.58 / 0.82)	(0.58 / 0.81)	(0.52 / 0.73)	(0.59 / 0.82)	(0.56 / 0.83)	(0.75 / 1.01)
Safari Versions						
v1	(0.05 / 0.41)	(0.38 / 0.99)	(0.19 / 0.39)	(0.2 / 0.39)	(0.1 / 0.37)	(0.3 / 0.54)
v2	(0.11 / 0.61)	(0.25 / 0.64)	(0.28 / 0.57)	(0.31 / 0.61)	(0.18 / 0.69)	(0.59 / 1.06)
v3	(21.31 / 8.37)	(3.31 / 8.59)	(3.72 / 7.62)	(4.18 / 8.29)	(2.05 / 7.95)	(9.22 / 16.54)
v4	(8.98 / 5.43)	(2.53 / 6.56)	(2.33 / 4.77)	(2.59 / 5.14)	(1.29 / 5.01)	(8.12 / 14.57)
v6	(1.94 / 2.52)	(1.04 / 2.7)	(1.1 / 2.26)	(1.24 / 2.46)	(0.8 / 3.1)	(2.41 / 4.32)

Table C.5: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 2 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
v7	(8.57 / 5.3)	(2.22 / 5.76)	(2.55 / 5.22)	(2.52 / 4.99)	(1.2 / 4.68)	(5.21 / 9.34)
v8	(49.8 / 12.79)	(6.07 / 15.76)	(1.29 / 2.65)	(5.89 / 11.68)	(2.89 / 11.23)	(10.05 / 18.03)
v9	(4.5 / 3.85)	(1.63 / 4.22)	(1.58 / 3.25)	(2.13 / 4.22)	(0.85 / 3.31)	(2.86 / 5.12)
Thunderbird Versions						
rolling	(0.82 / 1.43)	(1.22 / 2.09)	(0.79 / 1.34)	(0.84 / 1.4)	(0.79 / 1.43)	(2.78 / 3.13)
Ubuntu Versions						
v11.04	(0.39 / 0.68)	(0.42 / 0.73)	(0.38 / 0.64)	(0.39 / 0.65)	(0.38 / 0.69)	(0.87 / 0.98)
v12.04	(1.77 / 3.1)	(2.39 / 4.1)	(1.63 / 2.76)	(1.84 / 3.06)	(1.6 / 2.9)	(3.79 / 4.27)
v14.04	(1.17 / 2.04)	(4.78 / 8.22)	(1.37 / 2.33)	(1.24 / 2.07)	(1.07 / 1.95)	(2.16 / 2.44)
Windows Versions						
Windows XP	(1.1 / 1.55)	(1.14 / 1.59)	(1.05 / 1.46)	(1.1 / 1.54)	(1.03 / 1.52)	(1.52 / 2.06)
Windows Vista	(0.44 / 0.62)	(0.51 / 0.71)	(0.4 / 0.56)	(0.44 / 0.61)	(0.41 / 0.6)	(0.62 / 0.85)
Windows 7	(0.86 / 1.22)	(4.04 / 5.67)	(0.76 / 1.06)	(0.82 / 1.13)	(0.76 / 1.13)	(1.4 / 1.9)

Table C.6: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 3 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
Firefox Versions						
v0	(0.99 / 1.32)	(0.78 / 1.34)	(0.76 / 1.28)	(0.73 / 1.22)	(0.65 / 1.18)	(1.36 / 1.6)
v1	(1.67 / 1.71)	(1.05 / 1.81)	(0.97 / 1.63)	(1.02 / 1.7)	(0.91 / 1.66)	(3.33 / 3.91)
v2	(0.72 / 1.12)	(0.7 / 1.21)	(0.62 / 1.04)	(0.61 / 1.03)	(0.57 / 1.04)	(1.41 / 1.65)
v3	(0.95 / 1.29)	(2.25 / 3.87)	(0.7 / 1.18)	(0.75 / 1.26)	(0.69 / 1.25)	(1.65 / 1.94)
v7	(17.6 / 5.56)	(3.55 / 6.1)	(3.02 / 5.08)	(3.15 / 5.27)	(2.65 / 4.83)	(8.46 / 9.94)
v8 rolling	(15.33 / 5.19)	(3.65 / 6.27)	(2.64 / 4.43)	(2.97 / 4.98)	(2.75 / 5.01)	(5.32 / 6.25)
Google Chrome Versions						
v1	(0.62 / 0.97)	(0.79 / 1.06)	(0.77 / 0.91)	(0.83 / 0.97)	(0.62 / 0.96)	(1.39 / 1.76)
v2	(38.37 / 7.63)	(6.71 / 8.97)	(6.2 / 7.3)	(6.46 / 7.58)	(4.14 / 6.36)	(9.68 / 12.3)
v3	(40.03 / 7.79)	(5.95 / 7.94)	(5.86 / 6.89)	(6.6 / 7.74)	(5.43 / 8.34)	(10.94 / 13.9)
v4	(55.13 / 9.14)	(6.66 / 8.9)	(20.05 / 23.58)	(7.2 / 8.45)	(5.15 / 7.91)	(11.05 / 14.04)
Internet Explorer Versions						
v5	(0.71 / 0.79)	(0.89 / 1.25)	(0.54 / 0.75)	(0.64 / 0.88)	(0.51 / 0.76)	(1.98 / 2.6)
v6	(0.71 / 2.71)	(1.95 / 2.74)	(1.84 / 2.57)	(1.86 / 2.58)	(2.06 / 3.06)	(2.75 / 3.6)
v7	(0.71 / 0.66)	(0.49 / 0.69)	(0.46 / 0.64)	(0.48 / 0.67)	(0.42 / 0.62)	(0.69 / 0.9)
v8	(0.71 / 1.68)	(1.27 / 1.78)	(1.07 / 1.49)	(1.15 / 1.6)	(1.13 / 1.68)	(1.67 / 2.19)
v9	(0.71 / 4.14)	(2.97 / 4.18)	(2.41 / 3.37)	(3.01 / 4.18)	(2.93 / 4.36)	(3.68 / 4.82)
v10	(0.71 / 2.21)	(1.63 / 2.29)	(1.62 / 2.26)	(1.6 / 2.22)	(1.41 / 2.1)	(1.9 / 2.49)
v11	(0.71 / 5.09)	(4.21 / 5.9)	(3.43 / 4.78)	(3.29 / 4.56)	(3.16 / 4.7)	(4.76 / 6.24)

Table C.6: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 3 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
Mac OS X Versions						
v0	(0.5 / 1.66)	(0.66 / 1.68)	(0.81 / 1.61)	(0.83 / 1.65)	(0.41 / 1.59)	(0.92 / 1.75)
v2	(0.25 / 0.82)	(0.33 / 0.85)	(0.39 / 0.77)	(0.4 / 0.8)	(0.21 / 0.82)	(0.71 / 1.36)
v3	(0.3 / 1)	(2.96 / 7.59)	(0.47 / 0.94)	(0.51 / 1)	(0.25 / 0.96)	(0.72 / 1.38)
v4	(0.4 / 1.32)	(0.68 / 1.75)	(0.64 / 1.28)	(0.62 / 1.24)	(0.32 / 1.23)	(1.02 / 1.95)
v5	(0.77 / 2.52)	(2.68 / 6.87)	(1.16 / 2.32)	(0.9 / 1.79)	(0.63 / 2.43)	(1.55 / 2.96)
v6	(0.65 / 2.15)	(0.94 / 2.4)	(0.94 / 1.87)	(1.04 / 2.06)	(0.49 / 1.89)	(1.4 / 2.67)
v8	(1.5 / 4.95)	(2.06 / 5.27)	(2.23 / 4.44)	(2.42 / 4.81)	(1.23 / 4.76)	(7.74 / 14.74)
v9	(0.52 / 1.7)	(0.97 / 2.5)	(0.86 / 1.71)	(0.83 / 1.65)	(0.39 / 1.52)	(1.18 / 2.24)
v10	(5.42 / 17.82)	(7.91 / 20.26)	(4.41 / 8.78)	(8.25 / 16.4)	(3.96 / 15.35)	(11.39 / 21.7)
v11	(6.63 / 21.8)	(35.66 / 91.34)	(9.07 / 18.08)	(9.73 / 19.36)	(4.96 / 19.22)	(12.82 / 24.43)
Microsoft Office Versions						
Office 2001	(0.3 / 0.43)	(0.32 / 0.44)	(0.3 / 0.42)	(0.3 / 0.42)	(0.28 / 0.42)	(0.55 / 0.72)
Office 2003	(1.28 / 1.81)	(1.31 / 1.84)	(1.27 / 1.77)	(1.28 / 1.77)	(1.13 / 1.68)	(1.75 / 2.29)
Office 2010	(0.58 / 0.82)	(0.58 / 0.81)	(0.52 / 0.73)	(0.59 / 0.82)	(0.56 / 0.83)	(0.75 / 0.98)
Safari Versions						
v1	(0.05 / 0.41)	(0.43 / 1.11)	(0.2 / 0.39)	(0.2 / 0.4)	(0.1 / 0.37)	(0.28 / 0.53)
v2	(0.11 / 0.61)	(0.25 / 0.65)	(0.29 / 0.57)	(0.31 / 0.61)	(0.18 / 0.71)	(0.53 / 1.01)
v3	(21.47 / 8.41)	(3.37 / 8.63)	(3.84 / 7.66)	(4.19 / 8.33)	(2.1 / 8.14)	(8.34 / 15.89)
v4	(9.17 / 5.49)	(2.65 / 6.8)	(2.41 / 4.8)	(2.63 / 5.23)	(1.31 / 5.08)	(7.96 / 15.17)
v6	(1.98 / 2.55)	(1.07 / 2.73)	(1.13 / 2.25)	(1.25 / 2.49)	(0.84 / 3.24)	(2.16 / 4.13)

Table C.6: Forecasting Results (Mean Absolute Error / Root Mean Squared Error) for all Forecasting Methods and Software / System Packages over a 3 Month Forecasting Horizon

Method	SES	DES	TES	Arima	Croston	Nnet
v7	(8.9 / 5.41)	(2.35 / 6.02)	(2.63 / 5.24)	(2.55 / 5.06)	(1.23 / 4.77)	(5.06 / 9.65)
v8	(53.76 / 13.3)	(6.87 / 17.61)	(1.32 / 2.64)	(6.25 / 12.44)	(2.99 / 11.61)	(10.55 / 20.11)
v9	(5.75 / 4.35)	(1.75 / 4.49)	(1.72 / 3.42)	(2.26 / 4.49)	(0.86 / 3.33)	(2.32 / 4.42)
Thunderbird Versions						
rolling	(0.82 / 1.44)	(1.27 / 2.18)	(0.81 / 1.35)	(0.85 / 1.42)	(0.8 / 1.46)	(2.57 / 3.02)
Ubuntu Versions						
v11.04	(0.39 / 0.69)	(0.43 / 0.74)	(0.4 / 0.66)	(0.39 / 0.66)	(0.4 / 0.73)	(0.86 / 1.01)
v12.04	(1.79 / 3.14)	(2.47 / 4.24)	(1.64 / 2.76)	(1.84 / 3.08)	(1.63 / 2.97)	(3.69 / 4.34)
v14.04	(0.97 / 1.7)	(5.61 / 9.63)	(1.3 / 2.18)	(1.09 / 1.83)	(0.88 / 1.6)	(1.8 / 2.11)
Windows Versions						
Windows XP	(1.1 / 1.56)	(1.14 / 1.61)	(1.05 / 1.47)	(1.1 / 1.54)	(1.05 / 1.56)	(1.6 / 2.1)
Windows Vista	(0.44 / 0.63)	(0.52 / 0.73)	(0.4 / 0.56)	(0.44 / 0.61)	(0.41 / 0.61)	(0.64 / 0.83)
Windows 7	(0.86 / 1.22)	(4.47 / 6.28)	(0.76 / 1.06)	(0.82 / 1.14)	(0.77 / 1.14)	(1.47 / 1.93)

Appendix D. Descriptive Statistics

Table D.7: Descriptive Statistics

Software / System Package	Version	Time Frame	Vulnerabilities	Monthly Average
Firefox	v0	01/08/04 - 01/06/16	74	0.52
	v1	01/12/04 - 01/06/16	119	0.86
	v2	01/08/05 - 01/06/16	61	0.47
	v3	01/12/06 - 01/06/16	117	1.026
	v7	01/11/11 - 01/06/16	110	2
	v8 rolling	01/12/11 - 01/06/16	196	3.63
Google Chrome	v1	01/09/08 - 01/06/16	86	0.92
	v2	01/05/10 - 01/06/16	458	6.27
	v3	01/05/11 - 01/06/16	499	8.18
	v4	01/04/15 - 01/06/16	168	12
Internet Explorer	v5	01/03/99 - 01/06/16	157	0.76
	v6	01/04/98 - 01/06/16	373	1.71
	v7	01/12/05 - 01/06/16	31	0.25
	v8	01/05/08 - 01/06/16	99	1.02
	v9	01/06/11 - 01/06/16	288	4.8
	v10	01/03/13 - 01/06/16	77	1.97
	v11	01/12/13 - 01/06/16	152	5.07
Mac OS X	v0	01/07/01 - 01/06/16	174	4.83
	v2	01/12/02 - 01/06/16	36	0.22
	v3	01/11/03 - 01/06/16	53	0.35
	v4	01/06/05 - 01/06/16	137	0.79
	v5	01/11/07 - 01/06/16	130	1.26
	v6	01/03/10 - 01/06/16	22	0.29
	v8	01/03/13 - 01/06/16	82	2.10
	v9	01/11/13 - 01/06/16	50	1.61
	v10	01/11/14 - 01/06/16	51	0.25
	v11	01/09/15 - 01/06/16	140	15.56
MS Office	2001	01/12/99 - 01/06/16	16	0.08
	2003	01/03/06 - 01/06/16	136	1.11
	2010	01/11/10 - 01/06/16	34	0.51
Safari	v1	01/06/03 - 01/06/16	20	0.13
	v2	01/07/05 - 01/06/16	20	0.15

Table D.7: Descriptive Statistics

Software / System Package	Version	Time Frame	Vulnerabilities	Monthly Average
	v3	01/06/07 - 01/06/16	214	1.98
	v4	01/02/09 - 01/06/16	91	1.03
	v6	01/06/13 - 01/06/16	27	0.75
	v7	01/03/14 - 01/06/16	71	2.63
	v8	01/11/14 - 01/06/16	78	4.11
	v9	01/12/15 - 01/06/16	17	2.83
Thunderbird	rolling	01/08/04 - 01/06/16	116	0.82
Ubuntu	v11.04	01/09/08 - 01/06/16	17	0.18
	v12.04	01/05/12 - 01/06/16	94	1.92
	v14.04	01/05/14 - 01/06/16	54	2.16
Windows	XP	01/11/01 - 01/06/16	166	0.95
	Vista	01/02/07 - 01/06/16	40	0.36
	7	01/06/07 - 01/06/16	105	0.97

References

References

Alhazmi, O., Malaiya, Y., Ray, I., 2005. Security Vulnerabilities in Software Systems: A Quantitative Perspective. In: Jajodia, S., Wijesekera, D. (Eds.), Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security. Vol. 3654 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, August 7-10, Storrs, Connecticut, USA, pp. 281–294.

Alhazmi, O. H., Malaiya, Y. K., Ray, I., 2007. Measuring, analyzing and predicting security vulnerabilities in software systems. Computers & Security 26 (3), 219–228.

Amin-Naseri, M., Tabar, B. R., 2008. Neural Network Approach to Lumpy Demand Forecasting for Spare Parts in Process Industries. In: Gunawan, T. S. (Ed.), Proceedings of the 2008 International Conference on Computer and Communication Engineering. IEEE Computer Society, May 13-15, Kuala Lumpur, Malaysia, pp. 1378–1382.

Arora, A., Krishnan, R., Telang, R., Yang, Y., 2010. An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure. Information Systems Research 21 (1), 115–132.

Arora, A., Nandkumar, A., Telang, R., 2006. Does Information Security Attack Frequency Increase with Vulnerability Disclosure? An Empirical Analysis. Information Systems Frontiers 8 (5), 350–362.

Arora, S., Taylor, J. W., 2016. Forecasting Electricity Smart Meter Data Using Conditional Kernel Density Estimation. Omega 59, 47–59.

Chai, T., Draxler, R. R., 2014. Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? - Arguments against Avoiding RMSE in the Literature. Geoscientific Model Development 7 (3), 1247–1250.

Chaplot, V., Walter, C., Curmi, P., 2000. Improving soil hydromorphy prediction according to DEM resolution and available pedological data. Geoderma 97 (3-4), 405–422.

Chatfield, C., 2000. Time-Series Forecasting. Chapman & Hall/CRC, Boca Raton, Florida, USA.

Chatzipoulidis, A., Michalopoulos, D., Mavridis, I., 2015. Information Infrastructure Risk Prediction through Platform Vulnerability Analysis. Journal of Systems and Software 106, 28–41.

Chowdhury, I., Zulkernine, M., 2011. Using Complexity, Coupling, and Cohesion Metrics As Early Indicators of Vulnerabilities. Journal of Systems Architecture 57 (3), 294–313.

ContractorUK, May 2016. IT Contractor Guide to Data Breach and Cyber Security Insurance. URL http://www.contractoruk.com/insurance/guide_cyber_security_and_data_breach_insurance_it_contractors.html

Croston, J. D., 1972. Forecasting and Stock Control for Intermittent Demands. Journal of the Operational Research Society 23 (3), 289–303.

Der Voort, M., Dougherty, M., Watson, S., 1996. Combining Kohonen Maps with ARIMA Time Series Models to Forecast Traffic Flow. Transportation Research 4 (5), 307–318.

eWeek, May 2016. Unpatched Software and the Rising Cost of Breaches: Security Reports. URL http://www.eweek.com/security/unpatched-software-and-the-rising-cost-of-breaches-security-reports.html

Ferreiro, O., 1987. Methodologies for the Estimation of Missing Observations in Time Series. Statistics & Probability Letters 5 (1), 65–69.

FireEye, May 2016. Beyond The Bottom Line : The Real Cost Of Data Breaches. URL https://www2.fireeye.com/rs/848-DID-242/images/rpt-beyond-bottomline.pdf

Gegick, M., Rotella, P., Williams, L., 2009. Toward Non-Security Failures as a Predictor of Security Faults and Failures. In: Massacci, F., Redwine, S., Zannone, N. (Eds.), Proceedings of the First International Symposium on Engineering Secure Software and Systems. Engineering Secure Software and Systems. Springer-Verlag, Berlin, Heidelberg, February 4-6, Leuven, Belgium, pp. 135–149.

GhasemiGol, M., Ghaemi-Bafghi, A., Takabi, H., 2016. A Comprehensive Approach for Network Attack Forecasting. Computers & Security 58, 83–105.

Gospodinov, N., Gavala, A., Jiang, D., 2006. Forecasting volatility. Journal of Forecasting 25 (6), 381–400.

Gutierrez, R. S., Solis, A. O., Mukhopadhyay, S., 2008. Lumpy Demand Forecasting Using Neural Networks. International Journal of Production Economics 111 (2), 409–420.

Herbst, N. R., Huber, N., Kounev, S., Amrehn, E., 2014. Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. Concurrency and Computation: Practice and Experience 26 (12), 2053–2078.

Hinduja, S., Kooi, B., 2013. Curtailing Cyber and Information Security Vulnerabilities through Situational Crime Prevention. Security Journal 26 (4), 383–402.

Hyndman, R. J., 2017. Forecasting Functions for Time Series and Linear Models. R package version 8.0.
URL http://github.com/robjhyndman/forecast

Hyndman, R. J., Athanasopoulos, G., 2018. Forecasting: Principles and Practice. OTexts.
URL https://www.otexts.org/fpp/

Hyndman, R. J., Koehler, A. B., 2006. Another Look at Measures of Forecast Accuracy. International Journal of Forecasting 22 (4), 679–688.

Hyndman, R. J., et al., 2006. Another Look at Forecast-Accuracy Metrics for Intermittent Demand. Foresight: The International Journal of Applied Forecasting 4 (4), 43–46.

IBM Global Study, 2013. The Economics of IT Risk and Reputation: What Business Continuity and IT Security Really Mean to Your Organisation. Tech. rep., Ponemon Institute.
URL http://public.dhe.ibm.com/common/ssi/ecm/rl/en/rlw03022gben/RLW03022GBEN.PDF

Joh, H., 2011. Quantitative Analyses of Software Vulnerabilities. Ph.D. thesis, Colorado State University.

Johnson, P., Gorton, D., Lagerström, R., Ekstedt, M., 2016. Time between Vulnerability Disclosures: A Measure of Software Product Vulnerability. Computers & Security 62, 278–295.

Johnston, F., Boylan, J. E., 1996. Forecasting for Items with Intermittent Demand. Journal of the Operational Research Society 47 (1), 113–121.

Kim, J., Malaiya, Y., Ray, I., 2007. Vulnerability Discovery in Multi-Version Software Systems. In: Cukic, B., Dong, J. (Eds.), Proceedings of the Tenth IEEE High Assurance Systems Engineering Symposium. IEEE Computer Society, November 14-16, Dallas, Texas, USA, pp. 141–148.

Kim, S., Kim, H., 2016. A New Metric of Absolute Percentage Error for Intermittent Demand Forecasts. International Journal of Forecasting 32 (3), 669–679.

Kourentzes, N., 2013. Intermittent demand forecasts with neural networks. International Journal of Production Economics 143 (1), 198–206.

Last, D., 2016. Forecasting Zero-Day Vulnerabilities. In: Trien, J. P., Prowell, S. J., Goodall, J. R. (Eds.), Proceedings of the Eleventh Annual Cyber and Information Security Research Conference. Association for Computing Machinery, April 5 - 7, Oak Ridge, Tennessee, USA, pp. 1–4, Article No. 13.

Li, G., Shi, J., 2010. On Comparing Three Artificial Neural Networks for Wind Speed Forecasting. Applied Energy 87 (7), 2313–2320.

Li, J., Heap, A. D., 2011. A Review of Comparative Studies of Spatial Interpolation Methods in Environmental Sciences: Performance and Impact Factors. Ecological Informatics 6 (3-4), 228–241.

Martin, R. A., 2001. Managing Vulnerabilities in Networked Systems. Computer 34 (11), 32–38.

MITRE Corporation, 2017a. CVE - Common Vulnerabilities and Exposures: Download CVE. URL https://cve.mitre.org/data/downloads/

MITRE Corporation, 2017b. CVE - Common Vulnerabilities and Exposures: Request a CVE ID. URL https://cve.mitre.org/cve/request_id.html

MITRE Corporation, 2017c. CVE - Common Vulnerabilities and Exposures: Terminology. URL http://cve.mitre.org/about/terminology.html

Mozilla, November 2017. Release Notes for v 57.0. URL https://www.mozilla.org/en-US/firefox/57.0/releasenotes/

Neuhaus, S., Zimmermann, T., Holler, C., Zeller, A., 2007. Predicting Vulnerable Software Components. In: Ning, P. (Ed.), Proceedings of the Fourteenth ACM Conference on Computer and Communications Security. Association for Computing Machinery, October 29-November 2, Alexandria, Virginia, USA, pp. 529–540.

Nguyen, V. H., Tran, L. M. S., 2010. Predicting Vulnerable Software Components with Dependency Graphs. In: Scandariato, R., Williams, L. (Eds.), Proceedings of the Sixth International Workshop on Security Measurements and Metrics. Association for Computing Machinery, September 16-17, Bolzano, Italy, pp. 1–8, Article No. 3.

Nikolopoulos, K., Buxton, S., Khammash, M., Stern, P., 2016. Forecasting Branded and Generic Pharmaceuticals. International Journal of Forecasting 32 (2), 344–357.

Ogcu Kaya, G., Demirel, O. F., 2015. Parameter Optimization of Intermittent Demand Forecasting by Using Spreadsheet. Kybernetes 44 (4), 576–587.

Ozment, A., 2005. The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting. In: Camp, L. J. (Ed.), Proceedings of the Fourth Workshop on the Economics of Information Security. Harvard University, June 2-3, Cambridge, Massachusetts, USA, pp. 1–21.

Ozment, A., 2006. Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models. In: Gollmann, D., Massacci, F., Yautsiukhin, A. (Eds.), Proceedings of the First

Workshop on Quality of Protection. Vol. 23 of Advances in Information Security. Springer Boston Massachusetts, September 19-22, Como, Italy, pp. 1–13.

Ozment, A., Schechter, S. E., 2006. Milk or Wine: Does Software Security Improve with Age? In: Keromytis, A. D. (Ed.), Proceedings of the Fifteenth Conference on USENIX Security Symposium - Volume 15. USENIX Association Berkeley, California, USA, July 31-August 4, Vancouver, British Columbia, Canada, pp. 93–104.

Petropoulos, F., Kourentzes, N., 2015. Forecast combinations for intermittent demand. Journal of the Operational Research Society 66 (6), 914–924.

Petropoulos, F., Kourentzes, N., Nikolopoulos, K., 2016. Another Look at Estimators for Intermittent Demand. International Journal of Production Economics 181 (Part A), 154–161.

Ponemon Institute, June 2016. IBM — Ponemon Cost of Data Breach 2016.
URL http://www-03.ibm.com/security/infographics/data-breach/

Rescorla, E., 2005. Is Finding Security Holes a Good Idea? IEEE Security & Privacy 3 (1), 14–19.

Roumani, Y., Nwankpa, J. K., Roumani, Y. F., 2015. Time Series Modeling of Vulnerabilities. Computers & Security 51, 32–40.

Scandariato, R., Walden, J., Hovsepyan, A., Joosen, W., 2014. Predicting Vulnerable Software Components via Text Mining. IEEE Transactions on Software Engineering 40 (10), 993–1006.

Schryen, G., 2009. Security of Open Source and Closed Source Software: An Empirical Comparison of Published Vulnerabilities. In: Kendall, K. E., Varshney, U. (Eds.), Proceedings of the Fifteenth Americas Conference on Information Systems. Association for Information Systems, August 6 - 9, San Francisco, California, USA.

Schryen, G., 2011. Is Open Source Security a Myth? What Do Vulnerability and Patch Data Say? Communications of the ACM 54 (5), 130–139.

Shenstone, L., Hyndman, R. J., 2005. Stochastic Models Underlying Croston's Method for Intermittent Demand Forecasting. Journal of Forecasting 24 (6), 389–402.

Shin, Y., Meneely, A., Williams, L., Osborne, J., 2011. Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. IEEE Transactions on Software Engineering 37 (6), 772–787.

Shin, Y., Williams, L., 2008. An Empirical Model to Predict Security Vulnerabilities Using Code Complexity Metrics. In: Rombach, D. (Ed.), Proceedings of the Second ACM-IEEE International

Symposium on Empirical Software Engineering and Measurement. Association for Computing Machinery, October 9-10, Kaiserslautern, Germany, pp. 315–317.

Shin, Y., Williams, L., 2013. Can Traditional Fault Prediction Models Be Used for Vulnerability Prediction? Empirical Software Engineering 18 (1), 25–59.

Singh, U. K., Joshi, C., Gaud, N., 2016. Information Security Assessment by Quantifying Risk Level of Network Vulnerabilities. International Journal of Computer Applications 156 (2), 37–44.

Smith, B., Williams, L., 2011. Using SQL Hotspots in a Prioritization Heuristic for Detecting All Types of Web Application Vulnerabilities. In: O'Conner, L. (Ed.), Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation. IEEE Computer Society, March 21-25, Berlin, Germany, pp. 220–229.

Software Testing NEWS, October 2016. Software Code is Regularly Released before Security Testing New Survey Finds.
URL http://www.softwaretestingnews.co.uk/software-code-regularly-released-security-testing-new-survey-finds/

Syntetos, A., Boylan, J., 1999. Correcting the Bias in Forecasts of Intermittent Demand. In: Gerald, D. E. (Ed.), Proceedings of the Nineteenth International Symposium on Forecasting. June 26-30, Washington, D.C., USA.

Syntetos, A. A., Babai, M. Z., Gardner, E. S., 2015. Forecasting Intermittent Inventory Demands: Simple Parametric Methods vs. Bootstrapping. Journal of Business Research 68 (8), 1746–1752.

Syntetos, A. A., Boylan, J. E., 2005. The Accuracy of Intermittent Demand Estimates. International Journal of Forecasting 21 (2), 303–314.

Taylor, J. W., Snyder, R. D., 2012. Forecasting Intraday Time Series with Multiple Seasonal Cycles Using Parsimonious Seasonal Exponential Smoothing. Omega 40 (6), 748–757.

Telang, R., Wattal, S., 2007. An Empirical Analysis of the Impact of Software Vulnerability Announcements on Firm Stock Price. IEEE Transactions on Software Engineering 33 (8), 544–557.

Venter, H., Eloff, J. H., 2004. Vulnerability Forecasting - A Conceptual Model. Computers & Security 23 (6), 489–497.

Veracode, 2016. Bug Bounty Programs Are Not a Quick-Fix. Tech. rep., Veracode.
URL https://www.veracode.com/sites/default/files/Resources/Whitepapers/bug-bounty-programs-are-not-a-quick-fix.pdf

Walden, J., Stuckman, J., Scandariato, R., 2014. Predicting Vulnerable Components: Software Metrics vs Text Mining. In: O'Conner, L. (Ed.), Proceedings of the Twenty-Fifth IEEE International Symposium on Software Reliability Engineering. IEEE Computer Society, November 3-6, Naples, Italy, pp. 23–33.

Wang, J. A., Zhang, F., Xia, M., 2008. Temporal Metrics for Software Vulnerabilities. In: Sheldon, F. T., Mili, A. (Eds.), Proceedings of the Fourth Annual Workshop on Cyber Security and Information Intelligence Research. Association for Computing Machinery, May 12-14, Oak Ridge, Tennessee, USA, pp. 1–3.

Wang, X., Smith-Miles, K., Hyndman, R., 2009. Rule Induction for Forecasting Method Selection: Meta-learning the Characteristics of Univariate Time Series. Neurocomputing 72 (10), 2581–2594.

Willemain, T. R., Smart, C. N., Schwarz, H. F., 2004. A New Approach to Forecasting Intermittent Demand for Service Parts Inventories. International Journal of forecasting 20 (3), 375–387.

Willemain, T. R., Smart, C. N., Shockor, J. H., DeSautels, P. A., 1994. Forecasting Intermittent Demand in Manufacturing: A Comparative Evaluation of Croston's Method. International Journal of Forecasting 10 (4), 529–538.

Willmott, C. J., 1982. Some Comments on the Evaluation of Model Performance. Bulletin of the American Meteorological Society 63 (11), 1309–1313.

Willmott, C. J., Ackleson, S. G., Davis, R. E., Feddema, J. J., Klink, K. M., Legates, D. R., O'Donnell, J., Rowe, C. M., 1985. Statistics for the Evaluation and Comparison of Models. Journal of Geophysical Research: Oceans 90 (C5), 8995–9005.

Willmott, C. J., Matsuura, K., 2005. Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance. Climate Research 30 (1), 79–82.

Younis, A., Malaiya, Y. K., Ray, I., 2016. Assessing Vulnerability Exploitability Risk Using Software Properties. Software Quality Journal 24 (1), 159–202.

Zhang, S., Caragea, D., Ou, X., 2011. An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities. In: Hameurlain, A., Liddle, S. W., Schewe, K.-D., Zhou, X. (Eds.), Proceedings of the Twenty-Second International Conference on Database and Expert Systems Applications. Vol. 6860 of Lecture Notes in Computer Science. August 29 - September 2, Toulouse, France, pp. 217–231.

Zhao, W., Wang, J., Lu, H., 2014. Combining Forecasts of Electricity Consumption in China with Time-Varying Weights Updated by a High-Order Markov Chain Model. Omega 45, 80–91.