Forecasting the Time Series of eCPMs in Online Advertising

by

Huu-My NGUYEN

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN INFORMATION TECHNOLOGY ENGINEERING
M.A.Sc.

MONTREAL, APRIL 22, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Pascal Giard, Thesis supervisor
Department of Electrical Engineering, École de Technologie Supérieure

Mr. Jean-Marc Lina, President of the board of examiners
Department of Electrical Engineering, École de Technologie Supérieure

Mr. Georges Kaddoum, Member of the jury
Department of Electrical Engineering, École de Technologie Supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON APRIL 19, 2021

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

FOREWORD

This thesis representing 30 out of 45 credits of the Master's degree in Information Technology, was presented at the Ecole de Technologie Superieure. The supervisor of this report is professor Pascal Giard. This thesis is part of the application at our industrial partner M32 Connect. The application is to help web publishers manage the ad inventories more effectively and increase their ad space selling revenue.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor Professor Pascal Giard for all the thorough comments and supports throughout my Master study. I have learned from prof. Giard not only how to carry a research work but also how to communicate effectively. This dissertation would not be possible without his persistent help and guidance.

Secondly, I would like to express my appreciation to our industrial partner M32 Connect for providing me the opportunity to work on this interesting topic. My sincere thanks are to James Tweedie, Mo Kahlain, Claude Cajolet, Ananta Purohit, and Edoardo A. Dewhurst for all the knowledge in the field of online advertising and your useful comments in our discussions.

Thirdly, I want to thank my labmates Marwa, Tannaz, Ilshat, and Amir for making my Master journey memorable with many sweet memories.

Finally, I would like to give my deepest thanks to my wife and my family for their love and support over the years.

Prévision de séries temporelles de l'eCPM d'espaces publicitaires en ligne

Huu-My NGUYEN

RÉSUMÉ

La publicité en ligne est devenue la principale source de revenus pour de nombreux éditeurs de sites web. Avec le développement du «real-time bidding (RTB)», les éditeurs sont désormais en mesure de vendre leurs espaces publicitaires en temps réel. Le prix est déterminé par la demande du marché en temps réel. Dans cette thèse, nous avons tenté d'aider les éditeurs à prédire les prix de ventes attendues sur le marché de la publicité en ligne au cours des 30 prochains jours en utilisant les données historiques des deux dernières années. Tout d'abord, nous explorons l'utilisation du modèle «Auto Regressive Integrated Moving Average (ARIMA)» pour ajuster les séries temporelles des «effective cost per mille (eCPM)» historiques et faire la prédiction. Ensuite, nous illustrons la distribution de eCPM sur une période et développons un indicateur de confiance reflétant la volatilité du marché. Le processus d'entraînement et de prévision est ensuite intégré dans le pipeline de données du partenaire industriel pour évaluation en production.

Mots-clés: publicité en ligne, enchères en temps réel, série temporelle, arima

Forecasting the Time Series of eCPMs in Online Advertising

Huu-My NGUYEN

ABSTRACT

Online advertising has become the main channel of revenue for many web publishers. With the development of real-time bidding (RTB), publishers now are able to sell their advertising space in real-time, where the price is determined by the demand of the market at a time. In this thesis, we made an attempt to help publishers forecast the expected effective cost per mille (eCPM) of their ads in an RTB market in the next 30 days using the historical data of eCPM in the past 2 years. First, we explore the use of an Auto Regressive Integrated Moving Average (ARIMA) model to fit the time series of historical eCPM and make the forecast. Second, we examine the distribution of eCPM over a period and develop a confidence indicator which suggests the market volatility. The training and forecasting process is then integrated into our industrial partner data pipeline for evaluation in a production environment.

Keywords: online advertising, real-time bidding, time series, arima

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

RTB	Real-time Bidding
ML	Machine Learning
CPM	Cost Per Mille
eCPM	Effective Cost Per Mille
SSP	Supply Side Platform
DSP	Demand Side Platform
DMP	Data Mangement Platform
ARIMA	Auto Regressive Integrated Moving Average
RMSE	Root Mean Square Error
ADF	Augmented Dickey-Fuller
ACF	AutoCorrelation Function
PCF	Partial Autocorrelation Function
STD	Standard Deviation
CLI	Command-Line Interface

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

w	Smoothing window width
p	Order of autogressive component
d	Order of differencing
q	Order of moving average component
$	Canadian dollar (eCPM value)

INTRODUCTION

Online advertising has become a booming industry in recent years. In the online advertising market, the advertisers, who provide the advertisements, play the role of buyers and the web publishers, who own a website and its content, play the role of sellers. The selling process can be settled in two ways: direct and programmatic. In a direct process, the publisher works directly with advertisers to sell their advertising space on their websites. They negotiate the price, ad size, ad position, date, and how long the ad will be shown. The programmatic process is the process of selling and buying ads via a real-time bidding (RTB) auction where advertisers place the bids simultaneously and the highest bidder wins the inventory slot. By analyzing user's browsing history (cookies) as well as ad placement information, RTB is capable of delivering the best-matched ads to targeted audiences (Yuan, Wang, Li & Qin, 2014).

One common pricing model used in selling ads is cost per mille (CPM) which is the price advertisers pay for every 1,000 impressions. When measuring the average CPM for all website traffic volumes, the term effective cost per mille (eCPM) is used. The eCPM is a way to evaluate the performance of the ad inventory. Therefore, many publishers want to know their eCPM in advance so that they sell their ads at an appropriate price. However, this is a challenging task that requires advice from experts in the field.

In this project, we worked with our industrial partner M32 Connect [1]. The company helps digital publishers understand, control, and optimize their data and the various monetization sources and formats. In this thesis, we will leverage the machine learning techniques to build the tool to predict the expected eCPM in the next 30 days based on the historical eCPM time series. The ultimate goal is to help publishers make an informed decision when selling their ads inventories. Moreover, we develop a confidence indicator that indicates how likely the actual eCPM will be around the predicted value.

[1] https://m32connect.com

This thesis consists of three main chapters. Chapter 1 presents an overview of the real-time bidding market in online advertising and relevant works from both sides of the market. Chapter 2 describes our approach of modeling for eCPM and market volatility. Chapter 3 presents how we evaluate the model and the forecasting performance of different models.

CHAPTER 1

BACKGROUND

1.1 The Business of Real-time Bidding for Online Advertisement

Publishers can sell their ad slots through direct contacts (non-programmatic) or by the means of real-time auctions (programmatic). The most popular mechanism of trading ads in programmatic marketing is RTB. In RTB, one can distinguish between the supply side who provide the ad space, the demand side who is interested in purchasing the ad space, the market organizers, and supporting companies. The key players in RTB can be depicted as follows (Yuan *et al.*, 2014):

- The advertiser is the buyer of an ad impression. In RTB auction, advertisers bid for ad impressions according to their budget, marketing objectives, and strategy.
- The publisher is the owner of an online website and the seller of ad space.
- Ad platform is an exchange market that matches buyers and sellers for each ad impression.
- Demand Side Platform (DSP) is an agency platform that helps advertisers optimize their ad management and buying strategies.
- Supply Side Platform (SSP) is an agency platform that helps publishers manage their ad inventories and selling strategies.
- Data Management Platform (DMP) collects, stores, and analyzes Internet users' information. DMP helps DSP and advertisers target the audiences of their interest.

Figure 1.1 depicts the typical process of ad delivering in RTB. The process includes the following steps:

1. The process begins with an internet user who visits the publisher's website.
2. If there is an ad slot available on the website, a request will be sent to SSP and ad platform along with information of the user (cookie) and ad slots (ad size, position, etc).
3. DSP is informed of the available ad slots. The DSP then parses the information in the ad request, queries necessary information of this user from the DMP, and selects the matched advertisers.

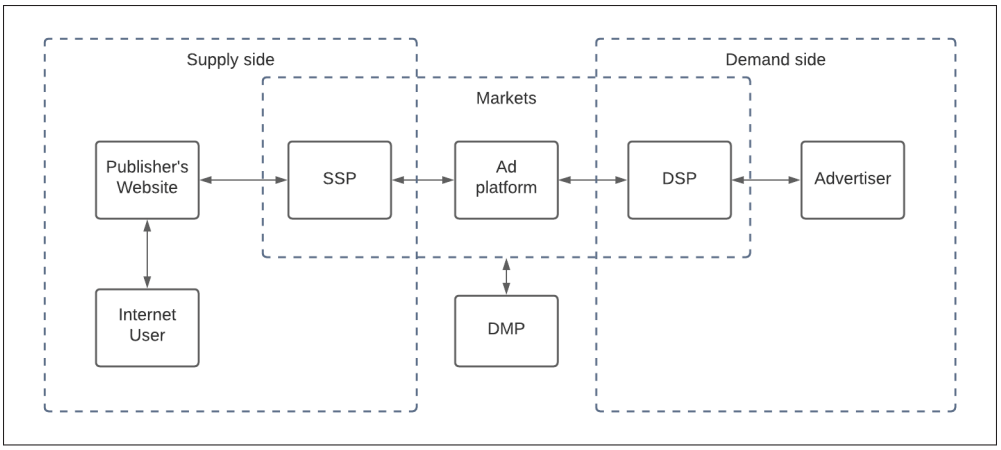


Figure 1.1 The process of ad delivery in RTB

4. Ad platform starts an auction and determines the winner with the highest bid. If the winner's bid is lower than the publisher's reserve price, the ad slots will be left empty. Otherwise, the winner's ad is displayed on the publisher's website.

There are two common auction mechanisms to determine the winner in online advertising: second-price auction and first-price auction. In both auction types, an advertiser with the highest bid is the winner. However, in a second-price auction, the winning bidder only pays the second-highest bid plus $0.01. While in a first-price auction, the winner has to pay exactly what they bid (Kulesza, 2019). Figure 1.2 illustrates these mechanisms with a concrete example. In this example, bidder B wins the auction with the highest bid of $2.8. In the second-price auction, bidder B will pay $2.51 to have his ad displayed on the publisher's website, whereas, in the first-price auction, he has to pay $2.8.

1.2 Related works

Since there are two key players in a RTB market: advertisers and publishers, many works have been done from both sides to achieve their goal in the market. Specifically, the goal of advertisers is to deliver their ads to the right users with minimal cost, whereas the publishers' goal is to maximize their selling ads space revenue. Even though our work is solely on the publisher side, studying the advertisers' behaviors helps us understand the dynamics of this market and thus,

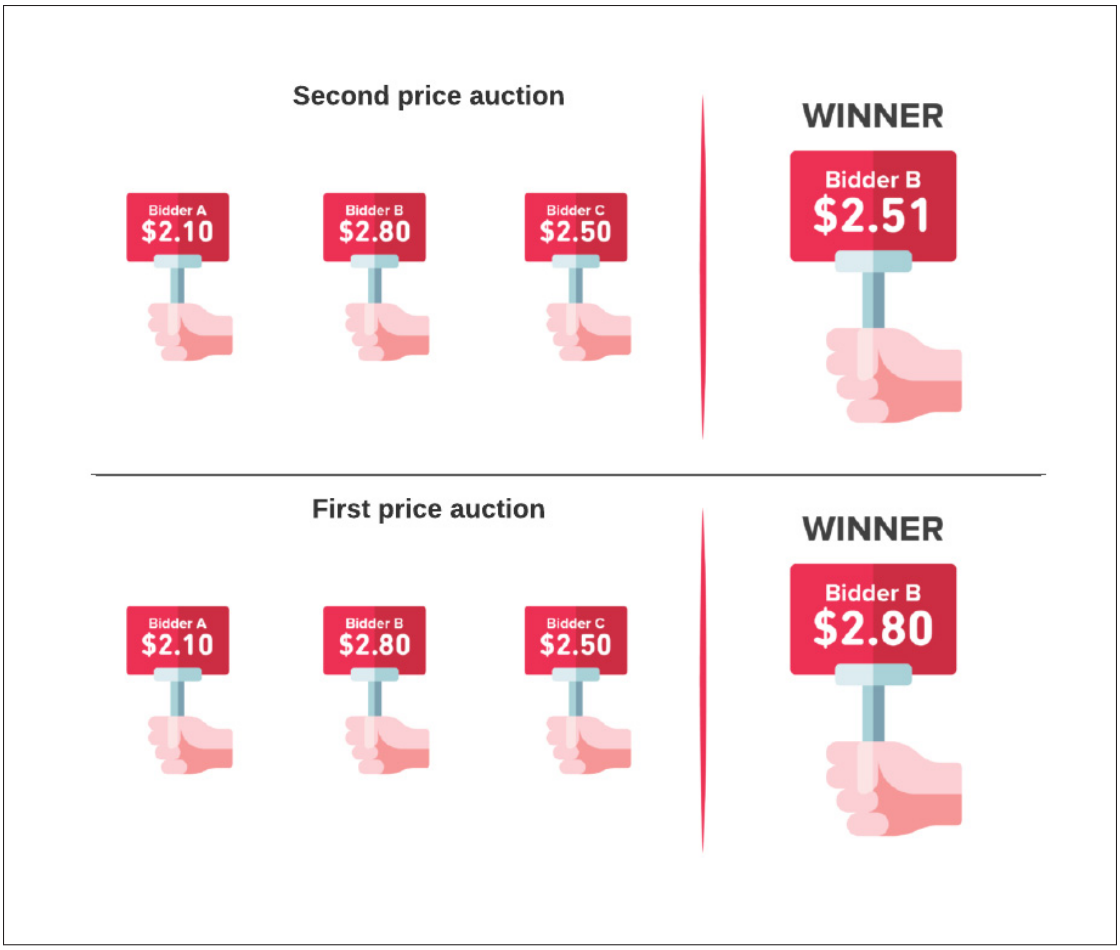


Figure 1.2 Second price auction versus first price auction
Taken from Kulesza (2019)

play accordingly. Below we summarize the main problems and attempt to solve them from each side of the market.

1.2.1 Advertisers' perspective

In the view of advertisers, the goal is to spend their campaign budget effectively to achieve high profits. Meaning that they seek ads that result in high positive user responses, e.g. clicks or conversions, but within the budget constraints. As a result, there are three main challenges that an advertiser needs to solve. First, they need to estimate the utility of an impression such as click-through rate with the consideration of the user, publisher, and other context information.

The second challenge is to predict the market price of an ad. The last challenge is to apply a proper bidding strategy that helps them win as many effective impressions as possible (Ren et al., 2018).

The utility metrics (click-through rate, conversion rate, etc) can be estimated by using machine learning models. Richardson, Dominowska & Ragno (2007) formulate the click-through rate using a logistic regression model while other models such as gradient boosting regression trees or Bayesian probit regression have been used in the works of Graepel, Candela, Borchert & Herbrich (2010) or He et al. (2014).

To predict the impression cost (CPM), one popular approach is to forecast the distribution of the prices offered by the market which is often referred to as landscape forecasting. Wu, Yeh & Chen (2015) consider a mixture model of both linear regressions on observed data and censored regression on bids with a censored winning price to solve the problem. Ren, Qin, Zheng, Yang, Zhang & Yu (2019) proposed an approach that combines deep learning for probability distribution forecasting and survival analysis for censorship handling.

Lastly, based on the estimated click-through rate and the impression cost, an optimal bidding function which considers the budget constraint is used to maximize the profit at the advertiser's side. Various bidding strategies have been proposed in Lin, Chuang, Wu & Chen (2016) and Liu, Yue, Qiu & Li (2020).

1.2.2 Publishers' perspective

From publishers' perspective, their goal is to maximize the revenue from selling the advertising space in both the traditional market (direct sales) and RTB market. The main mechanism that helps publishers control the price of an ad in RTB market is setting the reserve price aka floor price. Setting a higher reserve price might help drive higher CPM in RTB market. However, too high a reserve price might result in a loss of revenue due to lesser impressions being sold. Thus, finding an optimal reserve price has drawn attention from many researchers in the past few years.

Xie, Lee & Wang (2017) provided an efficient method of improving publisher revenue by adjusting the reserve price for only high-value ad inventories. First, they use a classifier to identify the inventories with top prices. Then, a cluster of classifiers is used to predict the price separations between the top two bids in the second price auction and a reserve price is calculated based on the difference between the two bids. Chahuara, Grislain, Jauvion & Renders (2017) propose a real-time solution to adjust the reserve price for each auction by estimate the first and second bid distribution using a simple regression model.

However, since mid-2019, the industry announced a transition from second-price auction to first-price auction for its ad exchange which has a strong impact on the strategy for reserve price optimization (Bigler, 2019). As a result, most of the prior works that rely on the first and second bids cannot be applied effectively for the new auction mechanism. Wodecki (2020) is one of the few works that attempt to solve this problem since the change in auction mechanism. In the proposed work, the author used time-series algorithms to make a short-term (3 days) forecast for the coverage parameter and then set the reserve price based on this predicted coverage parameter with the help of their agency consultants.

In our work, due to the nature of the available data, we take a semi-automatic approach to solve this problem. First, we make a forecast for the eCPM of an asset in the next 30 days solely based on the series of eCPM in the past. The approach we used to forecast eCPM is similar to Wodecki (2020) that different time series algorithms are evaluated to select the best one. However, our period of interest is 30 days instead of 3 days. The period of 30 days is chosen based on the business need of the customer as it gives enough time in advance for them to react while other periods like 7 days or 14 days are too short. Then based on the forecast eCPM, customers and experts from M32 Connect will have some insights into what the market could become in the next 30 days.

In the next chapter, we will describe our approach for forecasting the expected eCPM and the volatility of the market in the next 30 days.

CHAPTER 2

METHODOLOGY

2.1 Data preparation

In this project, we work on the data set provided by the M32 ad platform for a Media PublisherA which is a popular website in Canada. The actual name of the publisher is censored due to the confidential constraint. The data is collected over a 2-year period from 2018-11-01 to 2021-01-10. The data contains aggregated impressions and publisher revenue grouped by multiple levels such as report date, ad size, advertiser name, demand channel name, etc. There are over 50 data dimensions in the dataset. However, in this project, we only consider following features: date report, ad size, ad type, revenue raw, and impression raw. The details of these features are described in table 2.1. We calculate the daily eCPM by getting total revenue divided by the total impressions on that day.

Table 2.1 Feature dimensions

Feature name	**Description**
Date report	From 2018-11-01 to 2021-01-10
Ad size	For the scope of this research experiment, we only focus on 4 ad sizes: 300x250, 300x600, 728x90, and 970x250
Ad type	We only consider ads sold via programmatic channels. There are different types of programmatic sales. In this report, we combined and calculated the average eCPM for all programmatic channels.
Revenue raw	Total revenue on report date by ad size and ad type
Impression raw	Total impresson on report date by ad size and ad type
Daily eCPM	(Total Revenue raw) / (Total Impression raw)

Figure 2.1 shows the eCPM for 4 ad sizes for PublisherA website from 2018-11 to 2021-01. To protect the data for our partner, the y-axis of the figure shows 4 levels of eCPM: low, medium, high, and very high instead of the real values in dollars.

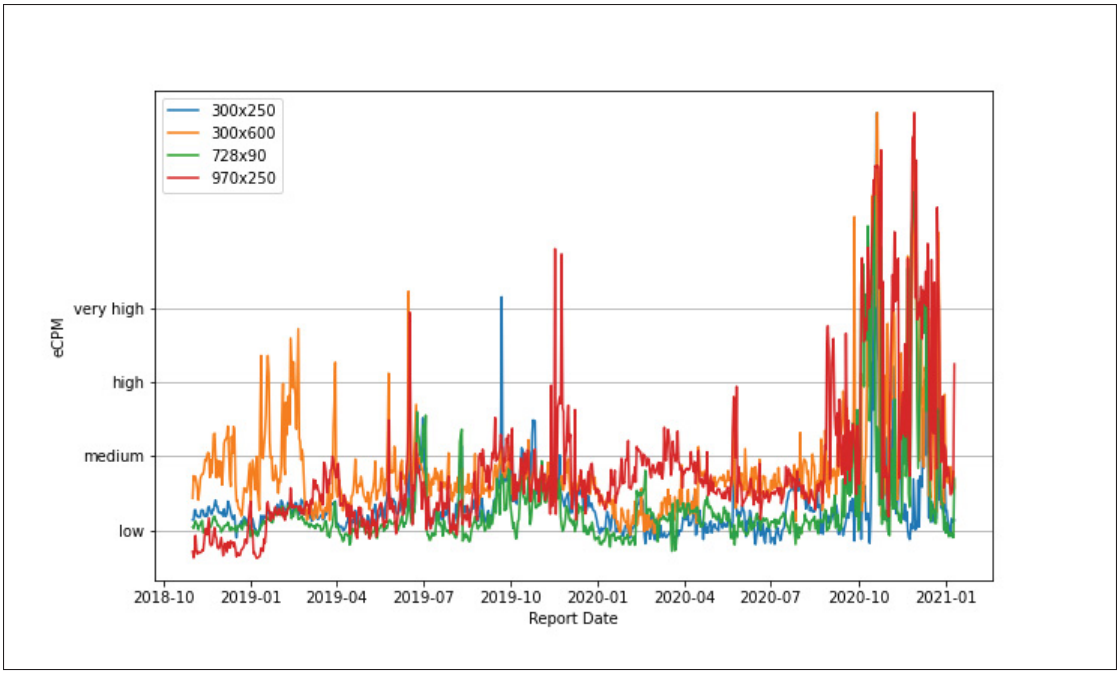


Figure 2.1 PublisherA daily raw eCPM

2.2 Predicting the average eCPM

In this section, we describe the proposed approach to predict the expected eCPM in the next 30 days based on the historical series of eCPM in the past 2 years. We also develop a confidence indicator that indicates how likely the actual eCPM will be around the predicted value. The goal of our approach is to provide the best forecasted eCPM that is as close to the actual eCPM as possible. However, during high volatile periods, it is unlikely to achieve this goal. In this case, the confidence indicator should reflect the fact that the market is unstable and thus, the publishers can lower their expectations when interpreting the forecast results.

Since our work aims to predict the expected eCPM in the next 30 days and the raw data contains many noises. We proposed a three-step procedure to forecast the expected eCPM in the next 30 days as follows:

- Step 1: Smooth training data with a window w
- Step 2: Create new series with relevant data points

- Step 3: Fit an Auto Regressive Integrated Moving Average (ARIMA) model on the new series to make the next forecast

In the next sections, we will provide more details over these steps.

2.2.1 Smoothing training data

First, a moving average method is applied to remove the spikes from the raw series $\mathcal{Y}$. It is also worth noting that these spikes are not faulty data but stemming from some unexpected events that cannot be predicted. In this project, we focus on the general trend of the eCPM rather than these unexpected events. A smoothed series $\mathcal{S}$ with window w can be calculated using equation below (Hyndman & Athanasopoulos, 2018b):

$$s_t = \frac{1}{w} \sum_{i=-w+1}^{0} y_{t+i} \tag{2.1}$$

Where s_t is the value at index t in the smoothed series $\mathcal{S}$ and y_t is the value at index t in the original series. Meaning that the estimated eCPM at time t is obtained by averaging the previous w values of the raw series $\mathcal{Y}$. Figure 2.2 shows the eCPM for 4 different ad sizes from PublisherA after applying the smoothing method with $w = 30$. Figure 2.3 illustrates the effects of various windows size w. In general, the larger the window size is, the smoother the average series become but less realistic to the original data. In the proposed approach, we treat w as a parameter in our model and perform a validation on a range of window sizes to find the best value for w. The details of validation results will be discussed in chapter 3.

2.2.2 Creating new series with relevant data points

The smoothed series $\mathcal{S}$ from section 2.2.1 is a eCPM series with daily index. However, in our project, we are interested in predicting the expected eCPM 30 days ahead. That means, if we use $\mathcal{S}$ directly as a training series, to obtain the forecast at date $t + 30$, we have to perform 30 single-step predictions. As a result, the prediction at date $t + 30$ is not expected to be reliable

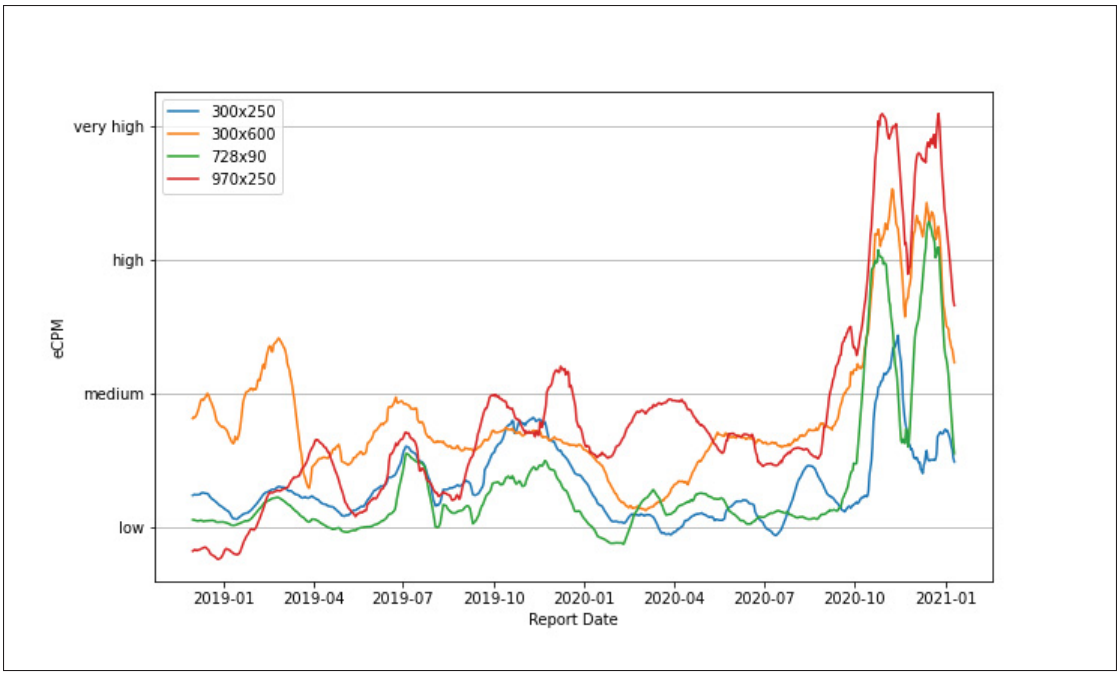


Figure 2.2 PublisherA daily eCPM after applying average smoothing window $w = 30$ days

since it relies on other predictions from date $t + 1$ to date $t + 29$. To achieve a better forecast, we create a new series $\mathcal{G}$ from series $\mathcal{S}$ by selecting the observations at each step of 30 days:

$$\mathcal{G}(i) = \mathcal{S}(30i) \tag{2.2}$$

where i is the index of the observation in the series. Studying the series $\mathcal{G}$ will help us capture the changes and patterns happening every 30 days period which is the goal of this project. Figure 2.4 illustrates the new series $\mathcal{G}$ for ad size 300x600 in red dots.

After generating the series $\mathcal{G}$ with only the selected observations of interest, we will fit an ARIMA model and make forecast the for next observation. The process of fitting an ARIMA model is described in section 2.2.3.

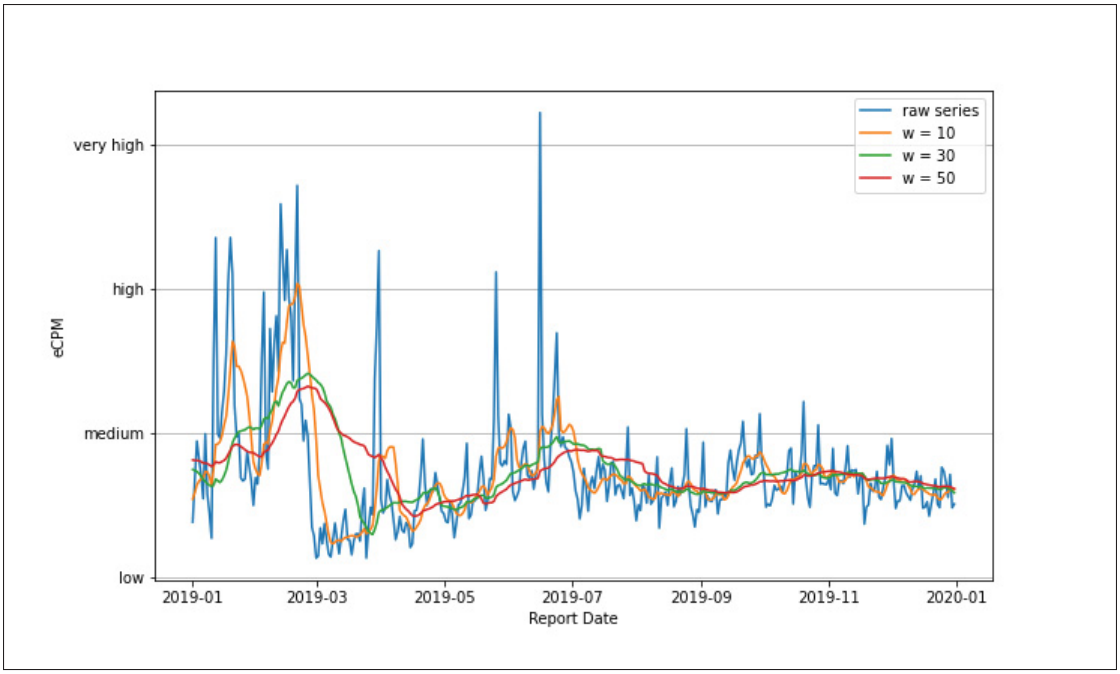


Figure 2.3 Smoothing effects of different windows for PublisherA ad size 300x600 in 2019

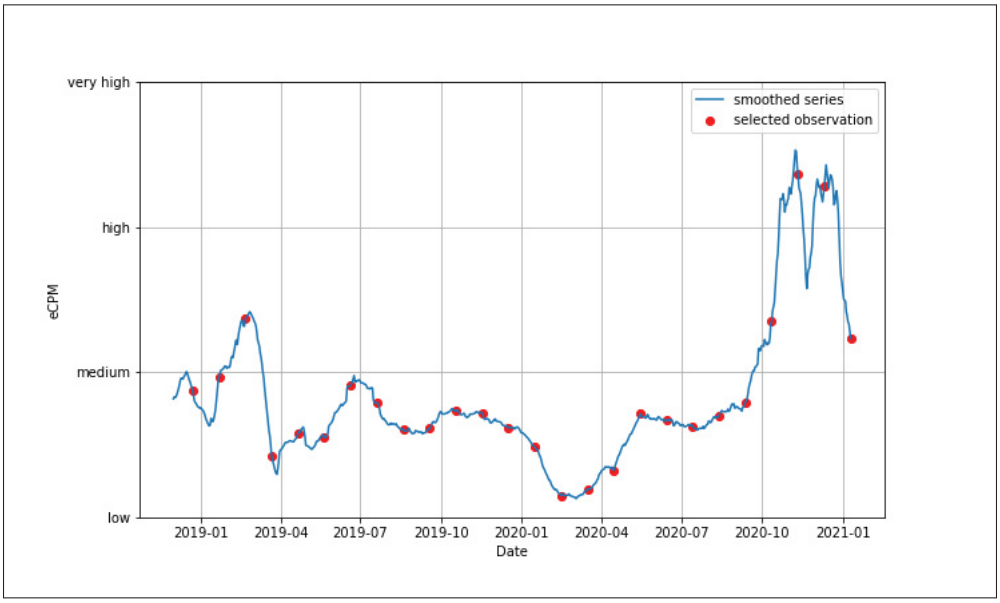


Figure 2.4 Selected observations from smoothed series ($w = 30$) for ad size 300x600

2.2.3 The ARIMA model and its components

An ARIMA model is a class of statistical models used for analyzing and forecasting time series data. An ARIMA model includes three components: autoregressive model, moving average model, and differencing operation (Hyndman & Athanasopoulos, 2018a). The following sections will explain each component of an ARIMA model in detail.

2.2.3.1 Autoregressive models

In an autoregressive model, we forecast the variable using a linear combination of past values of the variable itself (Hyndman & Athanasopoulos, 2018a). An autoregressive model of order p for time series $\mathcal{G}$ can be written as:

$$g_t = c + \phi_1 g_{t-1} + \phi_2 g_{t-2} + \cdots + \phi_p g_{t-p} + \varepsilon_t \tag{2.3}$$

where g_i is the observation at time index i of the series $\mathcal{G}$, c and ϕ_i are parameters that will be estimated to minimize the forecasting errors, and $\varepsilon_t \overset{iid}{\sim} N(0, \sigma_\varepsilon^2)$ is the error term. Usually, maximum likelihood estimation is used to estimate parameters c and ϕ_i where the parameters are estimated to maximize the likelihood function given the observed data. An autoregressive model of order p is often refered to as an AR(p) model.

2.2.3.2 Moving average models

A moving average model uses past forecasted errors to make a prediction (Hyndman & Athanasopoulos, 2018a). A moving average model of order q for time series $\mathcal{G}$ can be written as:

$$g_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \tag{2.4}$$

where g_t is the observation at time index t of the series $\mathcal{G}$, c and θ_i are the estimated parameters, and $\varepsilon_t \overset{iid}{\sim} N(0, \sigma_\varepsilon^2)$ is the error term. A moving average model of order q is often refered as a MA(q) model.

2.2.3.3 Stationarity and differencing

Stationarity means that the statistical properties of a process generating a time series do not depend on the time the series is observed (Palachy, 2019). Time series with trends, or with seasonality is not stationary because the values at different times are dependent on the trend and seasonality. To make a non-stationary time series $\mathcal{G}$ stationary, one can compute the differences between consecutive observations:

$$\Delta g_t = g_t - g_{t-1} \tag{2.5}$$

This operation is usually refered to as differencing. The differencing operation can be applied multiple times. A differencing of order d can be defined as:

$$\Delta^d g_t = \Delta(\Delta^{d-1} g_t) \tag{2.6}$$

2.2.3.4 ARIMA models

ARIMA is a family of models that captures temporal structures in time series data. The model has been applied widely in many forecasting problems and has proved its effectiveness as shown in Gilbert (2005), and Espinola, Nogales & Conejo (2003). ARIMA is an acronym that stands for Auto-Regressive Integrated Moving Average which means that the models comprise 3 components: an autoregressive part, a moving average part, and the use of differencing of raw observations in order to make the time series stationary. The full ARIMA model for time series $\mathcal{G}$ can be written as:

$$g'_t = c + \phi_1 g'_{t-1} + \cdots + \phi_p g'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{2.7}$$

where g' is the new series after applying the differencing transformation to eliminate trend and seasonal structures. As can be noticed in (2.7), the model is a linear combination of past p observations (2.3) and past q forecasted errors (2.4).

The model shown in (2.7) is often refered to as ARIMA(p, d, q) where:

- p is the order of the autoregressive part,
- d is the number of differencing operation applied,
- q is the order of the moving average part.

2.2.4 Identification of the ARIMA parameters suitable for our time series of eCPMs

In this section, we describe the process of identifying the orders of 3 components of ARIMA models for time series $\mathcal{G}$ from section 2.2.2. First, we determined the order of differencing, d, that makes the series stationary. To determine whether a series is stationary or not, we use a series of Augmented Dickey-Fuller (ADF) test (Dickey & Fuller, 1979). The ADF test is a statistical test that tests for a presence of a unit root in a time series. The null hypothesis of the test is that the time series can be represented by a unit root which means it is not stationary. The alternative hypothesis is that the time series is stationary. The result of our ADF test for time series $\mathcal{G}$ is shown in table 2.2. The p-values from the test for ad sizes are all above the significance level of 1%. Thus, we are unable to reject the null hypothesis that the time series is not stationary. This suggests further differencing operations are required. We performed the differencing on the time series and did the ADF test again. As shown in table 2.3, the p-value for ADF test this time is below the significance level of 1%. Therefore, we select $d = 1$ as the order of differencing in our ARIMA model.

Table 2.2 ADF test result for smoothed eCPM series

Ad size	Test result	p-value
300x600	-2.71	0.072
300x250	-3.38	0.011
728x90	-2.72	0.070
970x250	-2.17	0.217

Table 2.3 ADF test result for smoothed eCPM series with first order differencing

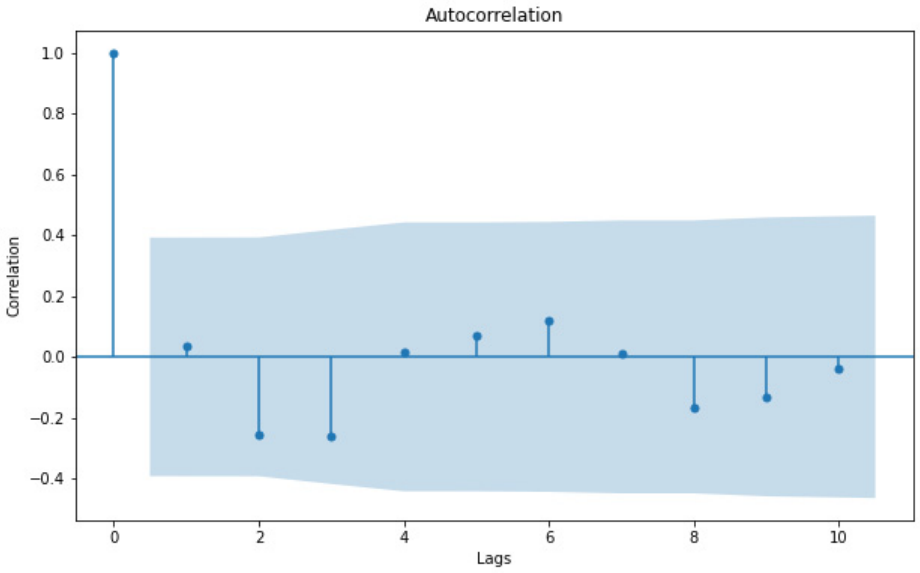
Ad size	Test result	p-value
300x600	-3.74	0.003
300x250	-5.26	0.000006
728x90	-4.04	0.001
970x250	-3.97	0.001

Next, to determine the order p and q of our ARIMA model, we examine the AutoCorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots of the differenced series. In brief, the ACF plot for series $\mathcal{G}$ shows the autocorrelations between g_t and g_{t-k} while the PACF measures the relationship between g_t and g_{t-k} after removing the effects of any lags between g_t and g_{t-k}.

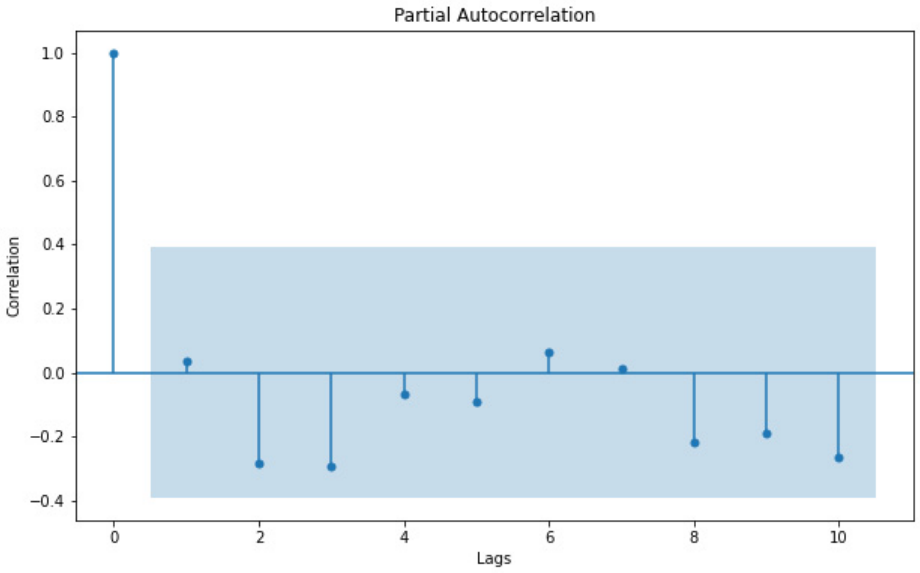
Figure 2.5 shows the ACF and PACF plots for the first order differencing series for ad size 300x600. The blue band in the plots indicates the 95% confidence interval when the estimated correlation is zero. In Figure 2.5, the correlations for any lags except lag 0 which is itself are within the blue band. This suggests that there is no statistically significant correlation between the series and its lagged values. Thus, we conclude that our series neither has an autoregressive component (AR) nor moving average components (MA). Meaning that the most appropriate model for the series $\mathcal{G}$ is ARIMA(0, 1, 0) which is a random walk model. Since there are only 26 observations in our 30-day interval series $\mathcal{G}$, it is not possible to come up with a statistically significant model. To confirm the selected order is the best one, we also evaluate different orders of ARIMA models and compare their results in chapter 3.

2.2.5 Other models

Besides ARIMA model, we also explore other simple forecasting methods as a comparison for the effectiveness of our selected methods. These simple forecasting methods are extremely simple yet surprising effective in many applications (Hyndman & Athanasopoulos, 2018a). In



a) ACF



b) PACF

Figure 2.5 ACF and PACF plots of differencing eCPM series for ad size 300x600

this thesis report, we will compare our ARIMA approach against the average method and the seasonal naive method.

2.2.5.1 Average method

For the average method, we simply forecast the eCPM at the next 30th days to be equal to the average of all observations in the past.

$$y_{t+30} = \frac{1}{N} \sum_{i=-N}^{0} y_{t+i} \tag{2.8}$$

where y_{t+30} is the forecast eCPM at date $t + 30$ of the raw series $\mathcal{Y}$ and N is the number of observations available prior to t.

2.2.5.2 Seasonal naive method

For the seasonal naive forecasting method, the forecasted value will be equal to the average of the same period last year.

$$y_{t+30} = \bar{y}_{t-335} \tag{2.9}$$

where $\bar{y}_{t-335}$ is the 30-day average of eCPM from the same date last year.

This section has described various models that we used to forecast the expected eCPM in the next 30 days. In the next section, we will discuss the confidence indicator which gives publisher insights about the volatility of the market in the future.

2.3 Confidence indicator

From our exploratory analysis, we discovered two important observations about the distribution of eCPM:

- First, the daily eCPM does not distribute evenly throughout the year. There are stable periods where the daily eCPM closely scatter around the average value for that period. Whereas, in

highly volatile periods, the prices are widely dispersed from the average price. As a result, when we make a forecast for the period that experiences high volatility of eCPM, there is a low probability that the actual eCPM would happen around this forecast value. Figure 2.6 illustrates this observation. In Figure 2.6, the green dots correspond to the actual eCPM from the raw series $\mathcal{Y}$ and the envelops are kernel density estimation of the underlying distribution. We can see that for the month of August 2020, our daily eCPM is stable and we have high confidence that actual eCPM will be between low and medium level. However, for highly volatile periods like October of 2020, the actual eCPM values are distributed between low and very high level, and the probability for the actual eCPM to be close to the mean is lower comparing to the stable period.

- Second, our initial hypothesis is that there is an annual seasonal pattern for volatility. Meaning that for the same period of the year, we would experience the same volatility. However, from our analysis, this hypothesis is not always correct. For example, in Figure 2.7, we can see that for ad size 300x600, in 2019, there is a wide dispersion of eCPM from Jan to Mar. However, for that period of 2020, the daily eCPM is relatively stable. In contrast, the eCPM experience high volatility for the period from Oct to Dec 2020 but stay steady for that period of 2019. On the other hand, in Figure 2.8, the eCPM for ad size 300x250 shows some sign of annual seasonality. The eCPM is relatively stable for the months from January to April for both 2019 and 2020 and then starts increasing towards the end of the year. Figure 2.9 and Figure 2.10 compare the distribution of eCPM for ad size 728x90 and 970x250 respectively. In both cases, the discrepancy between eCPM distribution in 2019 and 2020 can be observed.

These observations motivate us to develop a confidence indicator that would inform our publishers about the volatility of RTB market for the predicted period.

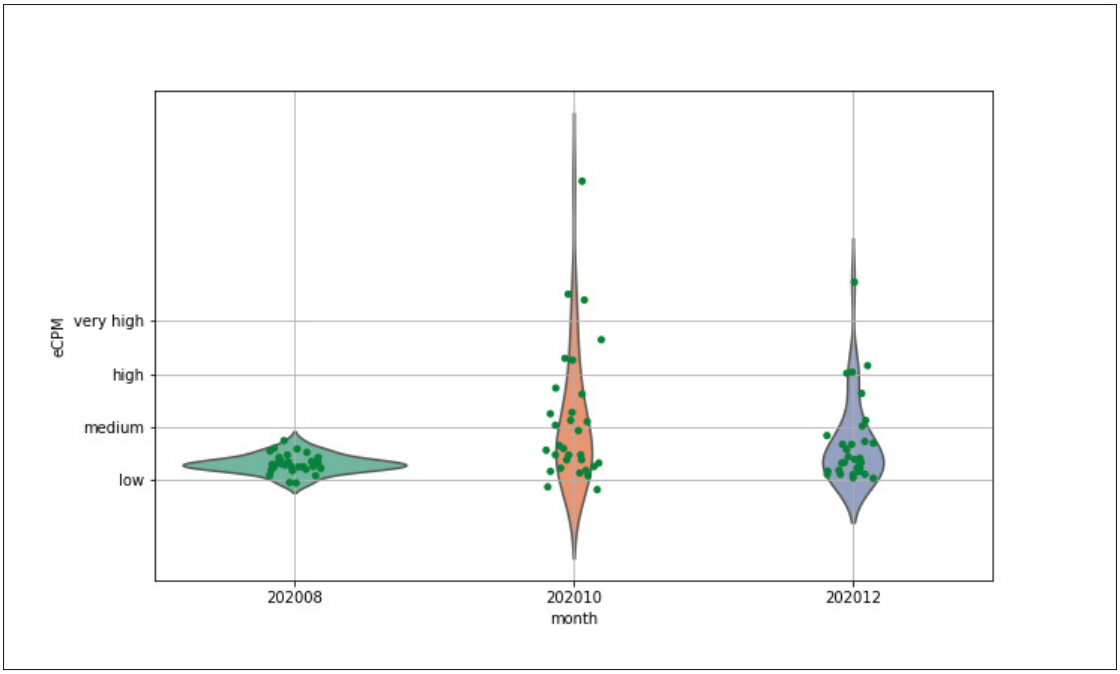


Figure 2.6 PublisherA daily eCPM distribution by month for 300x250 ad size

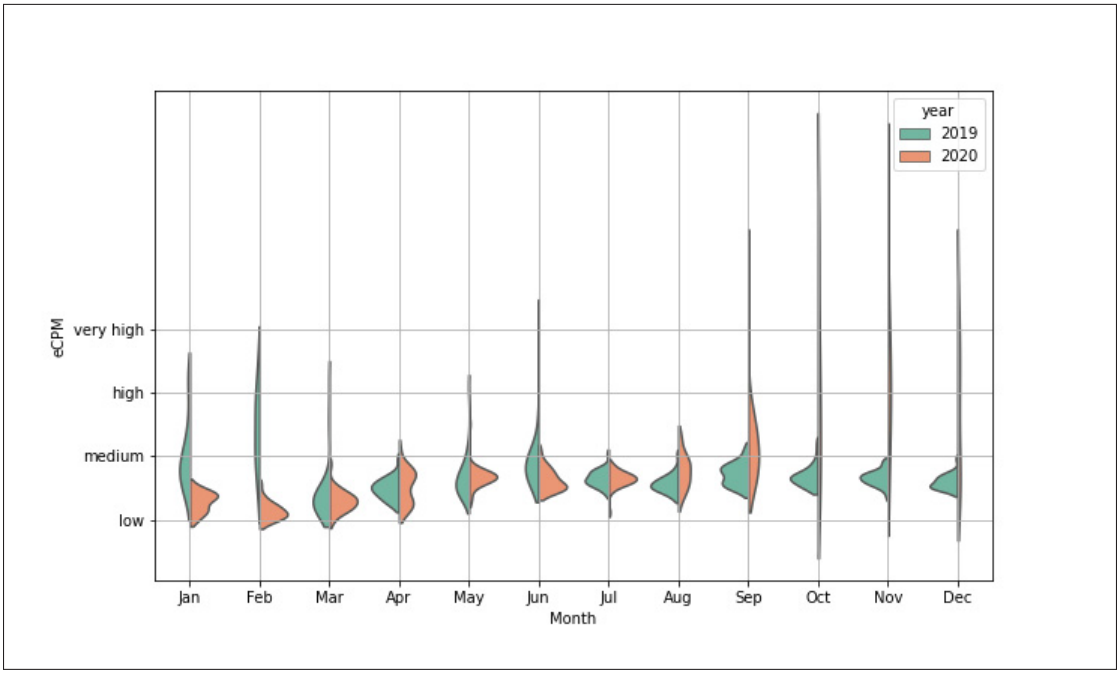


Figure 2.7 PublisherA daily eCPM distribution by month 2019 versus 2020 for 300x600 ad size

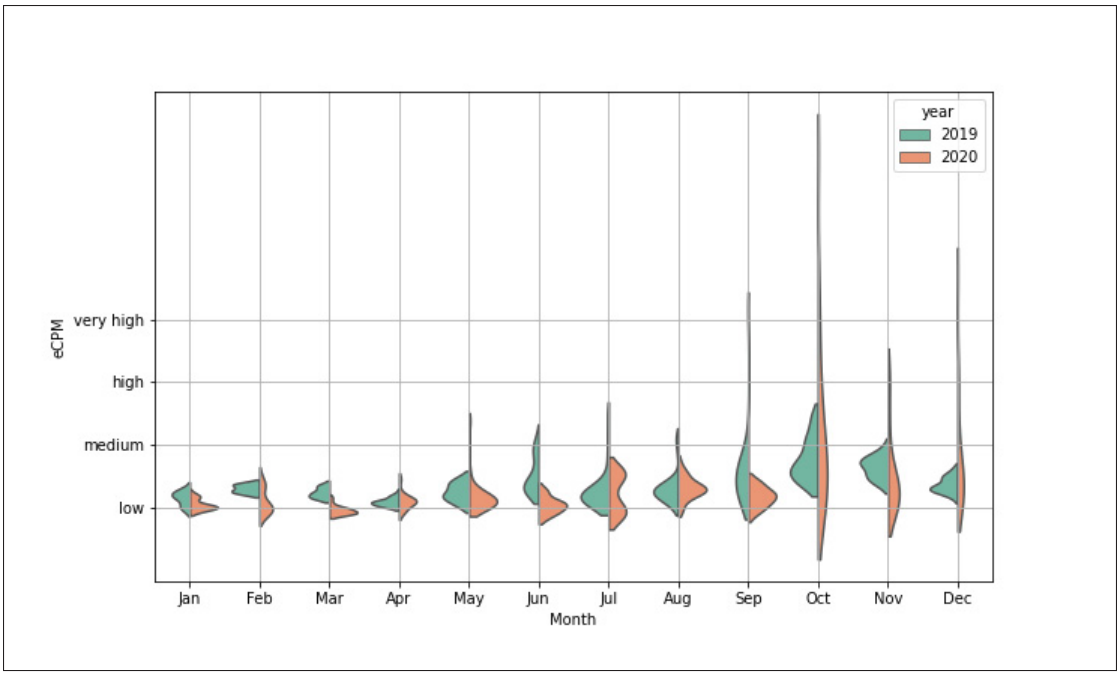


Figure 2.8 PublisherA daily eCPM distribution by month 2019 versus 2020 for 300x250 ad size

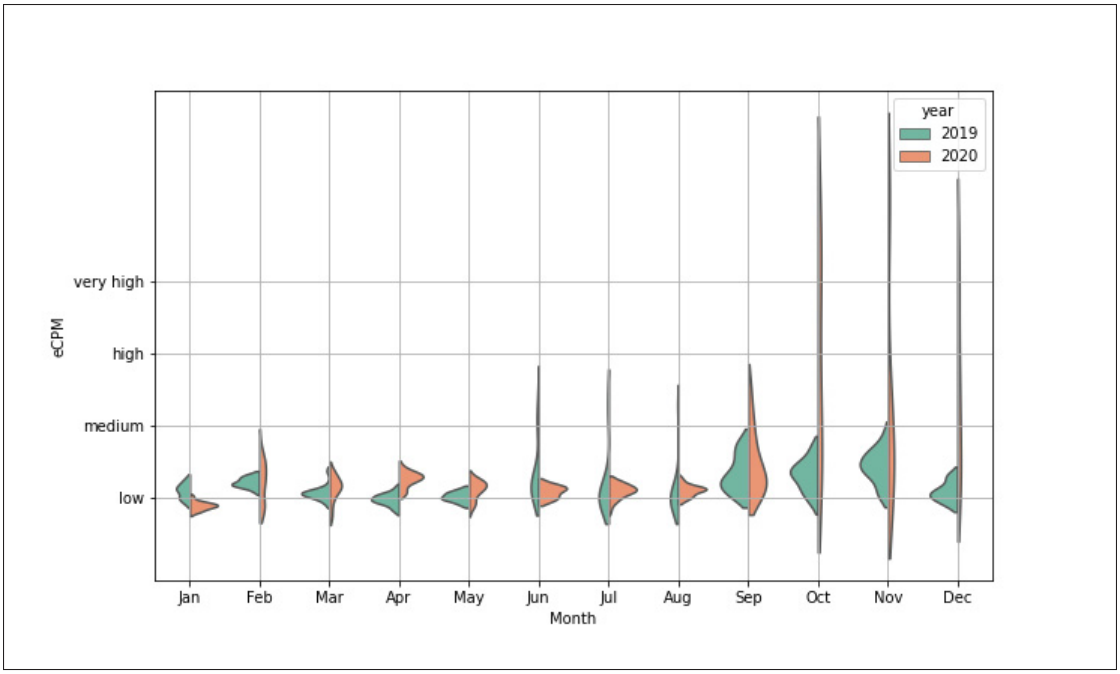


Figure 2.9 PublisherA daily eCPM distribution by month 2019 versus 2020 for 728x90 ad size

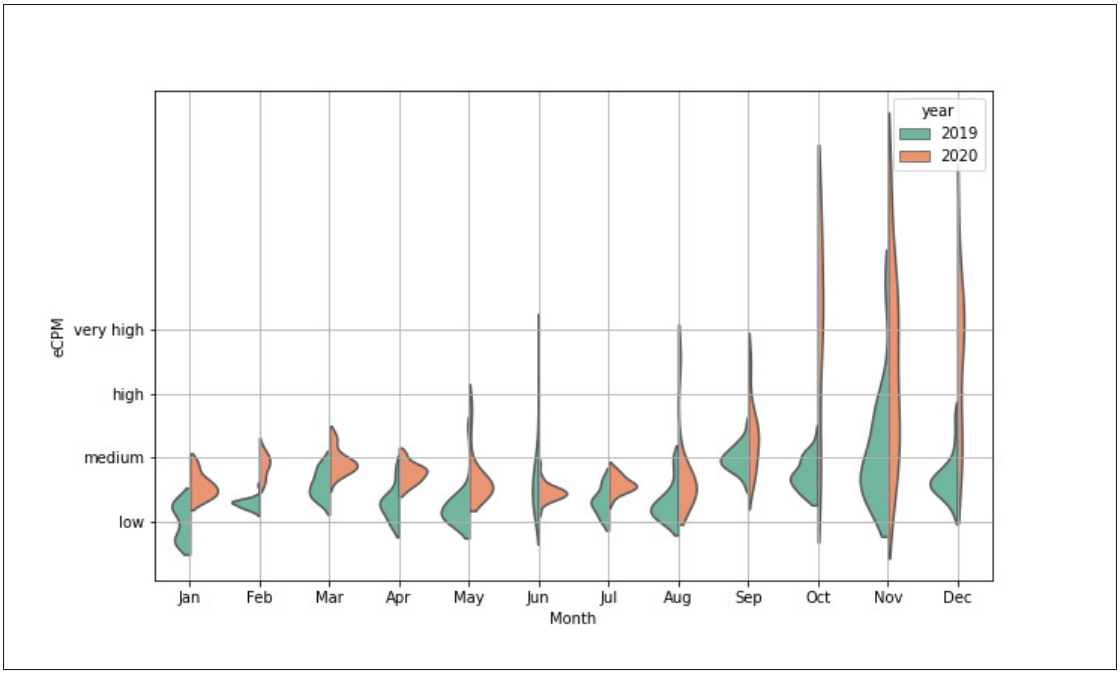


Figure 2.10 PublisherA daily eCPM distribution by month 2019 versus 2020 for 970x250 ad size

2.3.1 Modelling eCPM volalitity

We use sample standard deviation as metric to measure eCPM volatility for a 30-day period:

$$std = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \bar{y})^2}{N - 1}} \tag{2.10}$$

where N = 30 is the number of daily eCPM in the observed interval, $\bar{y}$ is the average eCPM over the observed interval and y_i is the actual value in the series. Standard deviation measures the spread of data distribution. A low value of standard deviation indicates low volatility in the market and vice versa. In a normal distribution, most of the eCPM distributes around the mean and we can expect approximately 68% of the eCPM should lie between the mean ±1 standard deviation.

Since we do not know the standard deviation for a period in the future, we make a prediction based on the historical series of the past standard deviations. Figure 2.11 shows the series of standard deviations for different ad sizes from Jan 2019 to Jan 2020. Each value in the series is the sample deviation of eCPM in the last 30 days. The process of fitting an ARIMA model to forecast the standard deviation for the next 30 days is performed similarly to the process of forecasting the mean eCPM in section 2.2.3 which can be summarized as follow:

- Calculate a series of standard deviations with the interval of 30 days.
- Analyze the series to find the appropriate orders for the ARIMA model.
- Evaluate the model on a validation data set to confirm the selected order and compare with other forecasting methods

Details of the evaluation process will be discussed in chapter 3. The best performing model is saved and used to make a forecast for a 30-day period standard deviation in the future.

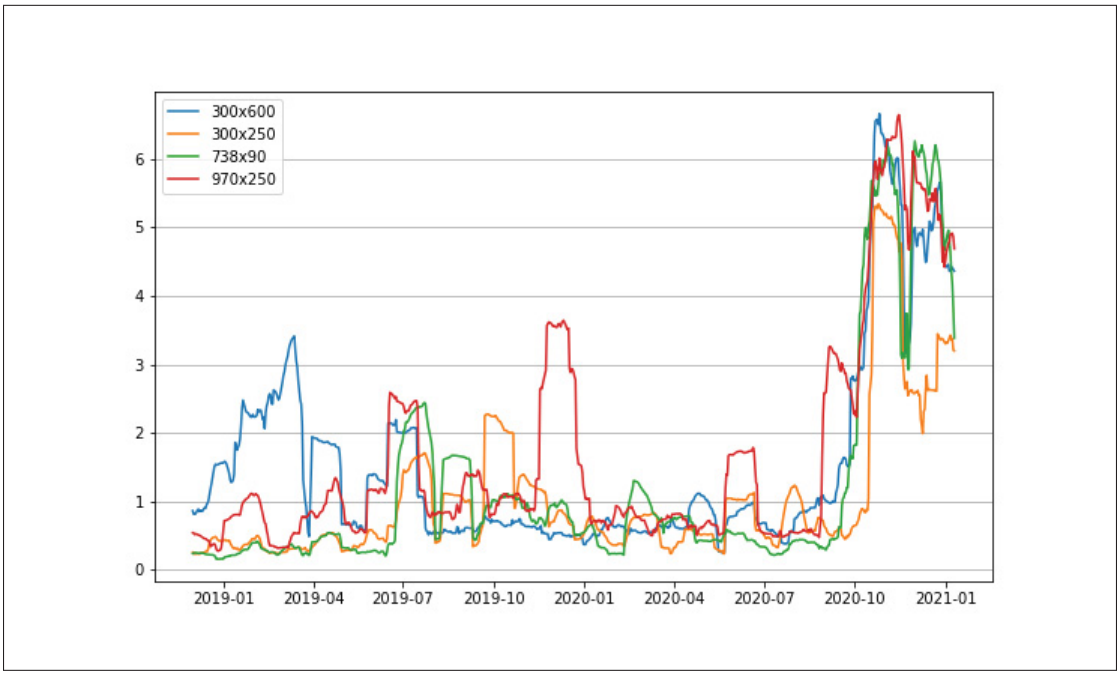


Figure 2.11 Time series of standard deviations for different ad sizes

2.3.2 Confidence indicator based on predicted volatility

Even though standard deviation provides a good implication about market volatility, it is hard to understand directly by our end users. To provide a more intuitive way to present the volatility, we develop a confidence indicator which is a different presentation on top of the standard deviation metric. The indicator has three levels of confidence: high, medium, and low. Table 2.4 shows the mapping between the standard deviation metric and three different confidence levels. These thresholds are chosen based on analyzing the visualization of data and consulting with our industrial partners at M32 Connect. In this work, we study the data from one publisher only. Thus, these thresholds may not be suitable for all ad sizes nor publishers. The task of establishing the thresholds for all ad sizes or publishers is left for future work.

Table 2.4 Confidence level based on standard deviation levels

Standard deviation level	**Confidence level**
< 0.5	High
>= 0.5 and <= 1	Medium
> 1	Low

In this chapter, we have described our approach of forecasting the expected eCPM in the next 30 days by fitting an ARIMA model on the time series of historical eCPM. In addition, we examined the distribution of eCPM over each 30 days period and use standard deviation (STD) to measure the market volatility. A confidence indicator is developed based on the forecasting STD metric to inform publishers about market volatility in the upcoming period. In the next chapter, we will describe the methodology we used to obtain the forecast results and to select the best models.

CHAPTER 3

RESULTS

3.1 Model evaluation

To evaluate the model, we split the collected data into a training set and a test set. The training set includes 533 observations from 2018-11-01 to 2020-04-16. The test set includes 240 observations from 2020-05-16 to 2021-01-10. Since we use a walk-forward validation procedure to validate the models, the training set will increase by one after each iteration. This train-test split gives us enough observations in the test set to have a stable validation score. The walk-forward validation procedure is illustrated in Figure 3.1. Specifically, it includes the following steps (Brownlee, 2017):

1. First, the model is trained with an initial training set. In our case, the initial training set contains data from 2018-11-01 to 2020-04-16.
2. The forecast eCPM for the next 30 days is made using the fitted model.
3. The forecast value is stored to evaluate against the actual eCPM value.
4. The training data set is expanded to include the next observation eCPM.
5. The process is repeated until all the forecast for the validation period is finished.

Note that there is a gap of 30 days between our initial training set and the first validation date. This is because our prediction window is 30 days. To emulate the real situation, we do not include the observations in this 30-day window into the training set. The walk-forward validation procedure also helps us mimic the production scenario where we attempt to make the best prediction with all the available data at date t.

To evaluate the forecast accuracy, we used Root Mean Square Error (RMSE) metric to measure the prediction quality:

$$\textbf{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{3.1}$$

where y_i and $\hat{y}_i$ are the observed value and the forecast value, respectively.

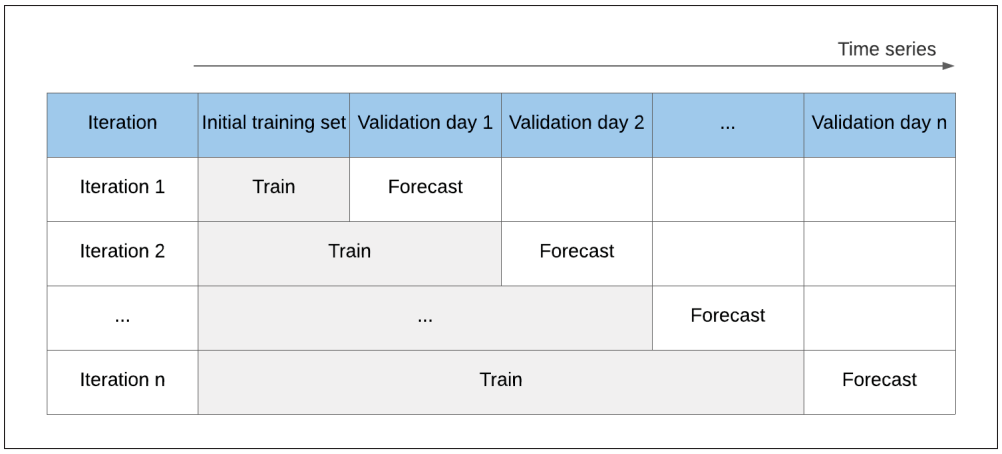


Figure 3.1 Validation with walk forward forecast strategy

In addition to RMSE, we also calculate the Mean Absolute Percentage Error (MAPE) metric in the evaluation process:

$$\textbf{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{3.2}$$

Compared to RMSE, MAPE normalizes the error relatively to the eCPM value and is less sensitive to large errors. We chose RMSE as our metric to measure model accuracy since our goal is to forecast the mean eCPM and thus, large errors are undesirable. For example, for the same MAPE of 50%, the forecast eCPM of $10 and the actual eCPM of $20 would have a strong impact on the revenue compared to the forecast of $1 and the actual eCPM of $2. Thus, the MAPE in the report is to serve as a reference only since it gives us an idea of the magnitude of the errors in relation to their actual values.

3.2 Evaluation results

To select the best model for each ad size, we run the evaluation process with different parameters of the ARIMA model and smoothing windows. We also conduct comparisons against other baseline approaches which are the average method and the naive seasonal method.

First, we study the errors when forecasting the expected eCPM for the next 30 days. The results are shown in Table 3.1 for four different ad sizes. For brevity, Table 3.1 only shows the results for

ARIMA with order of p and q between 0 and 1. In the experiments, we also ran the evaluation process with a higher order of p and q. However, these ARIMA models do not bring more accurate forecast results and are thus not included in this thesis. Based on the RMSE metric, we can see that to achieve the best forecasting performance, different models are required for different ad sizes. For 300x600 and 970x250 ad sizes, the best performing model is ARIMA model with order (0, 1, 0) using a smoothing window of 20 and 30, respectively. This means that the smoothing series of eCPM for ad size 300x600 and 970x250 is a random walk where the future change is unpredictable and the best forecast is equal to the last observation in the smoothing series. Whereas the best model for 728x90 ad size is ARIMA with order (1, 1, 0) with a smoothing window of 40 and the seasonal naive is the best fit for 300x250 ad size. The MAPE results show that our forecasting errors are still relatively high compared to the actual eCPM values where error can go up to around 50%.

The evaluation results reflect the facts that AR model with longer past in the model does not bring more accurate prediction. Since there are no significant correlations between the forecasted values and observations that happen more than 30 days ago. Thus, the maximum order of p in our selected models is 1.

Table 3.1 RMSE and MAPE metrics for forecasting the mean eCPM where arima$(p, d, q)_w$ stands for ARIMA model with order (p, d, q) and smoothing window w

	300x600		**300x250**		**728x90**		**970x250**	
Model	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
arima(0,1,0)_20	**4.109**	36.1	3.219	55.8	4.883	65.9	4.793	40.8
arima(1,1,0)_20	4.126	36.5	3.153	56.8	4.840	66.1	5.188	43.4
arima(1,1,1)_20	4.242	36.4	3.030	49.0	4.991	62.1	5.285	43.2
arima(0,1,0)_30	4.144	35.3	3.080	51.3	4.408	55.3	**4.805**	39.4
arima(1,1,0)_30	4.154	35.7	3.045	51.8	4.491	59.2	5.004	41.3
arima(1,1,1)_30	4.129	34.4	3.017	49.0	4.454	52.6	5.078	41.3
arima(0,1,0)_40	4.208	35.5	2.982	47.9	4.321	51.1	4.936	39.0
arima(1,1,0)_40	4.198	36.1	3.012	49.8	**4.231**	52.4	5.014	40.4
arima(1,1,1)_40	4.173	34.8	2.991	49.1	4.233	48.3	5.076	40.4
average	4.792	27.8	2.843	36.9	4.945	34.4	6.416	33.7
seasonal naive	4.851	31.0	**2.63**	51.2	4.688	36.4	5.841	36.2

In our validation process, we performed an exhaustive sweep of the smoothing window size w from 1, which is no smoothing at all, to 100. Figure 3.2, Figure 3.3 and Figure 3.4 illustrate the effect of window sizes on the forecasting results for ARIMA(0, 1, 0), ARIMA(1, 1, 0) and ARIMA(1, 1, 1), respectively, for ad size 300x600. In the figures, the forecasting results are shown in comparison against the actual raw data and the smoothed series. Overall, the effect of different window sizes is marginal when the eCPM is stable like the period from 2020-06 to 2020-09. However, when the eCPM is trending quickly up or down, like the period from 2020-10 to 2021-01, the effect of different window sizes becomes more significant. Larger window sizes are slower to adapt to the change. A similar pattern can be observed for ad size 300x250 (Figure 3.5, 3.6, and 3.7), ad size 728x90 (Figure 3.8, 3.9, and 3.10) and ad size 970x250 (Figure 3.11, 3.12, and 3.13).

Figure 3.14 compares the forecast results of the best ARIMA model against the average and seasonal naive approach. As can be seen in the Figure, the ARIMA model provide better forecasts compared to the two naive approaches. Figure 3.16 and Figure 3.17 show a similar trend for ad size 970x250 and 728x90, respectively. In contrast, the eCPM time series of ad size 300x250 show a different pattern where the seasonal naive approach provide a better fit compared to the ARIMA model.

In summary, the models using small window sizes tend to be too sensitive to changes while large windows are slow to respond to the changes. Depend on the characteristics of the series, different smoothing window sizes are required to achieve the best forecasting performance. As shown in Table 3.1, the selected window sizes are 20, 30, and 40 corresponding to ad size 300x600, 970x250, and 728x90.

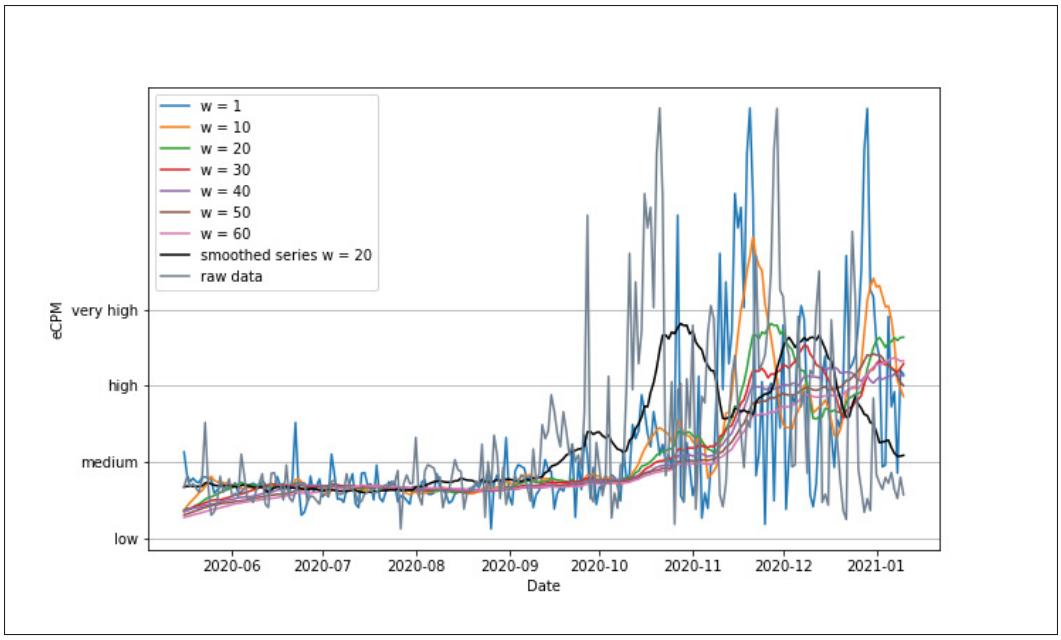


Figure 3.2 Forecast results using ARIMA(0, 1, 0) model with different smoothing windows for 300x600 ad size

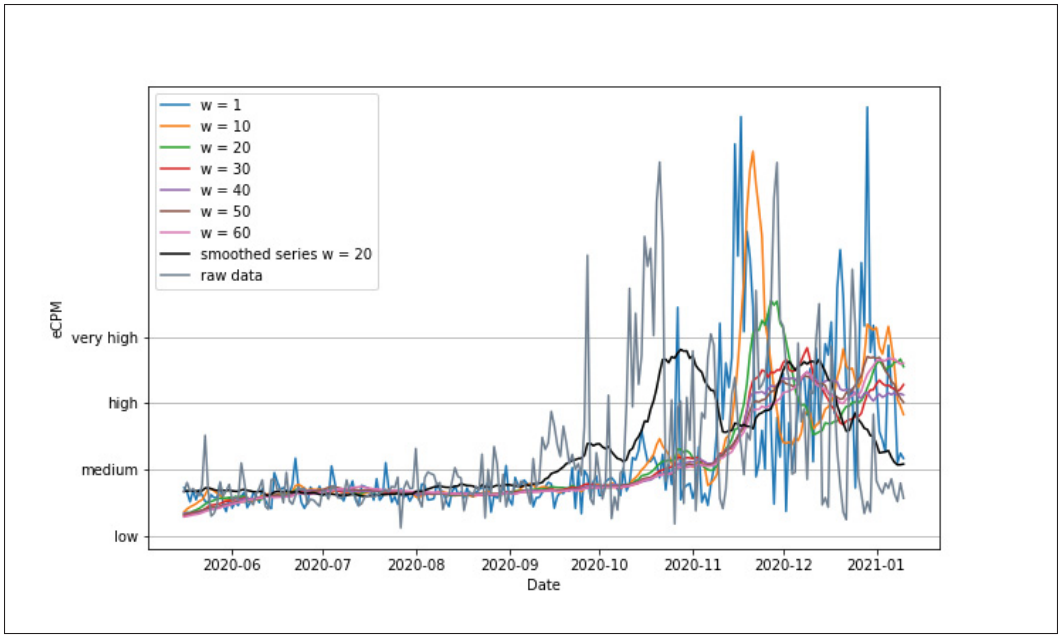


Figure 3.3 Forecast results using ARIMA(1, 1, 0) model with different smoothing windows for 300x600 ad size

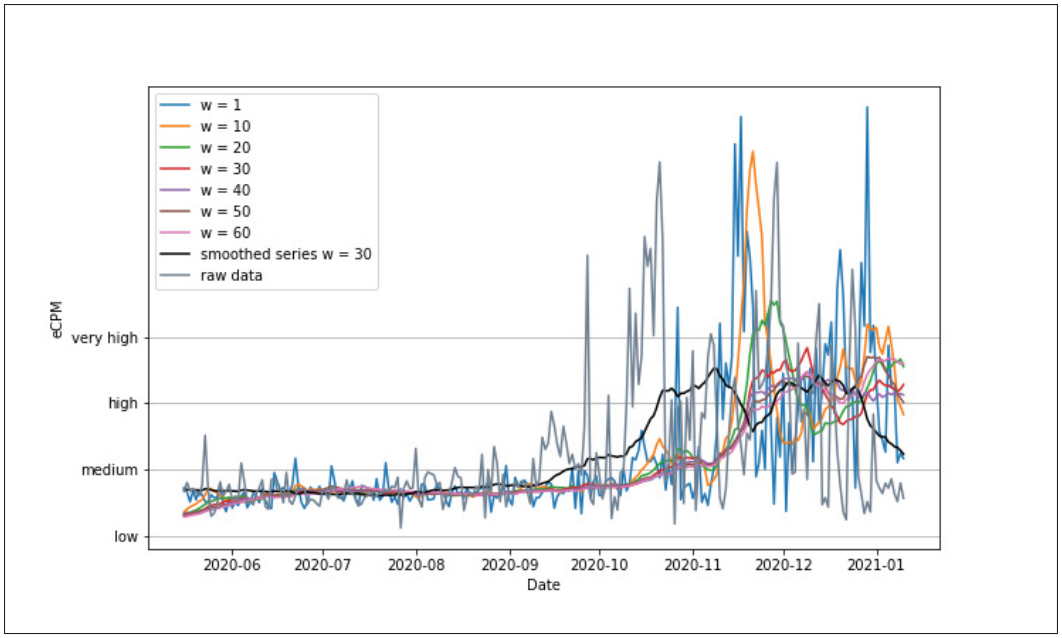


Figure 3.4 Forecast results using ARIMA(1, 1, 1) model with different smoothing windows for 300x600 ad size

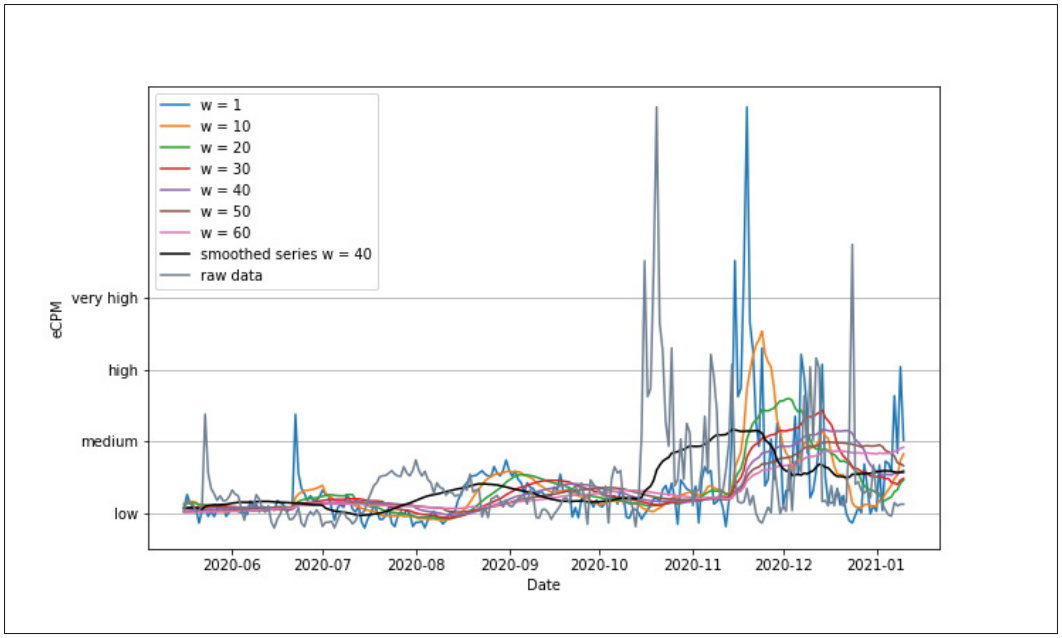


Figure 3.5 Forecast results using ARIMA(0, 1, 0) model with different smoothing windows for 300x250 ad size

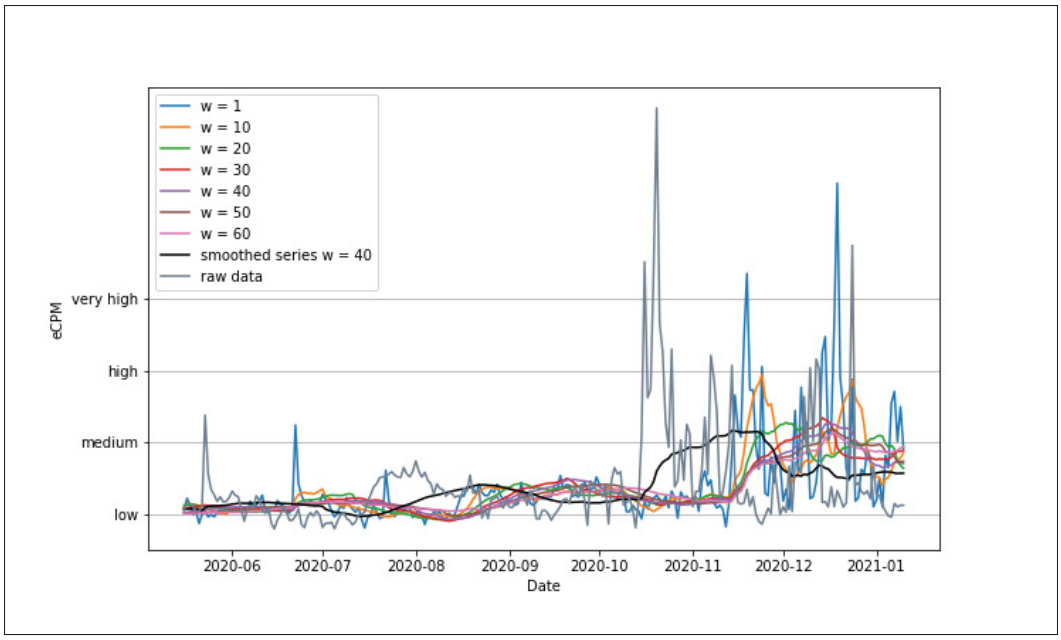


Figure 3.6 Forecast results using ARIMA(1, 1, 0) model with different smoothing windows for 300x250 ad size

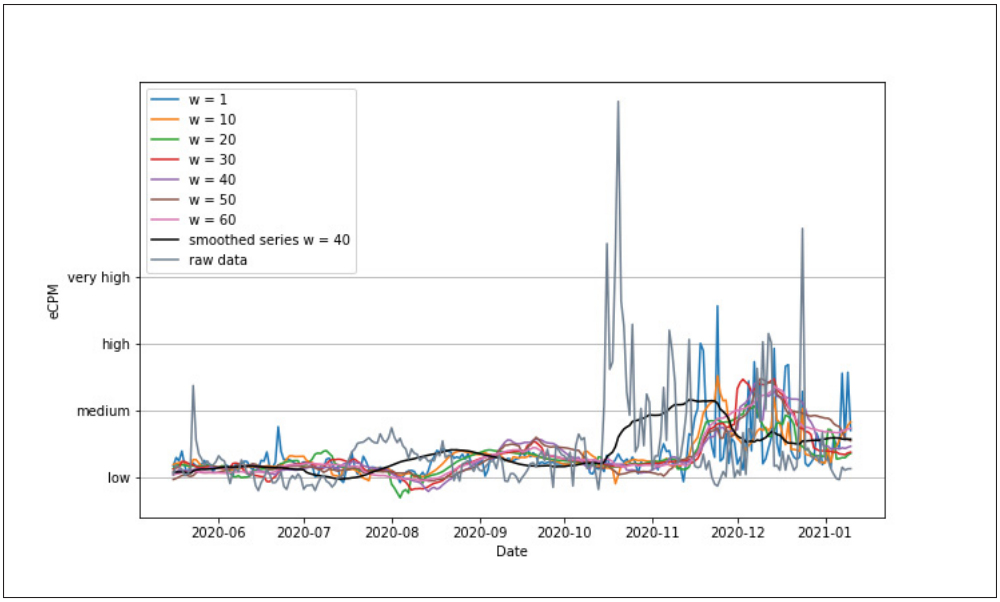


Figure 3.7 Forecast results using ARIMA(1, 1, 1) model with different smoothing windows for 300x250 ad size

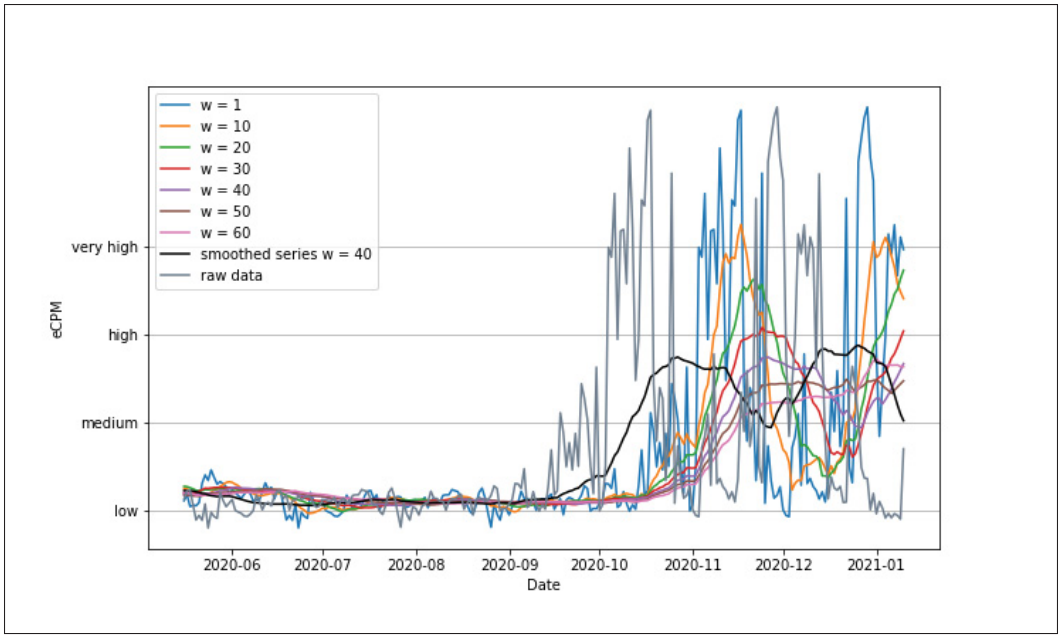


Figure 3.8 Forecast results using ARIMA(0, 1, 0) model with different smoothing windows for 728x90 ad size

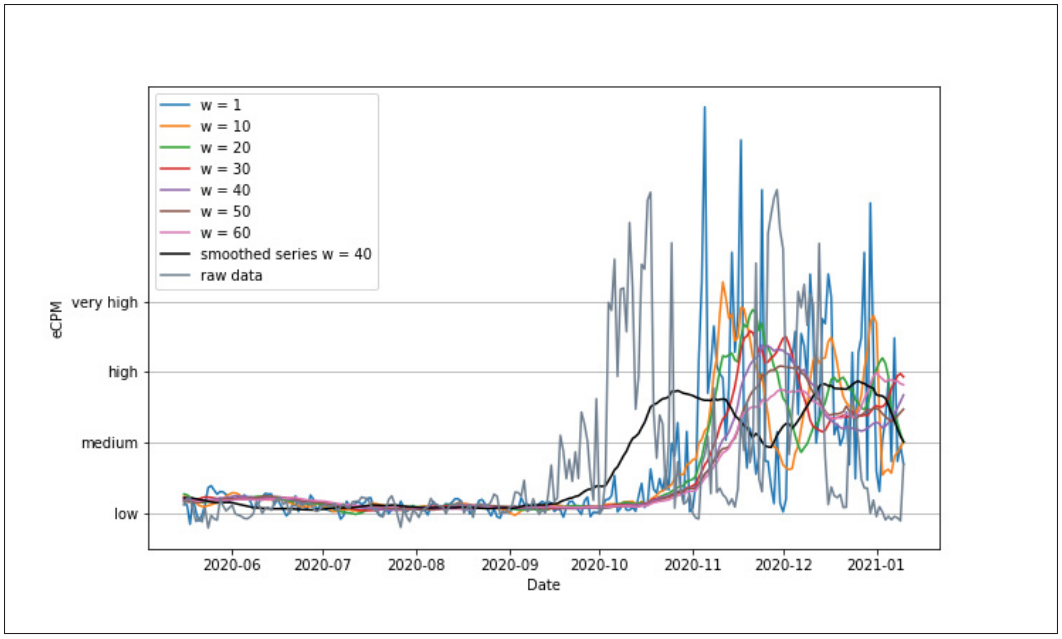


Figure 3.9 Forecast results using ARIMA(1, 1, 0) model with different smoothing windows for 728x90 ad size

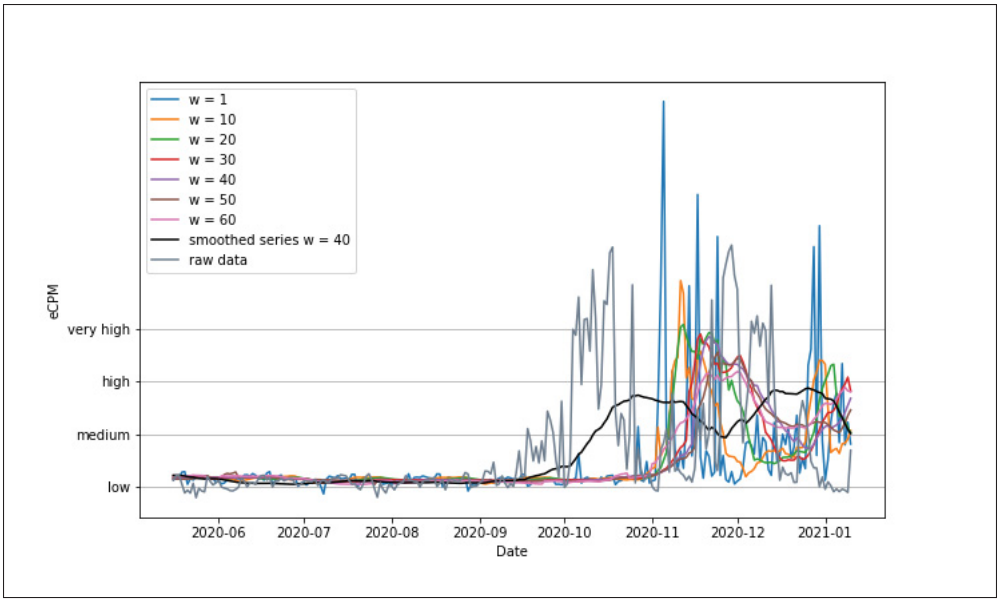


Figure 3.10 Forecast results using ARIMA(1, 1, 1) model with different smoothing windows for 728x90 ad size

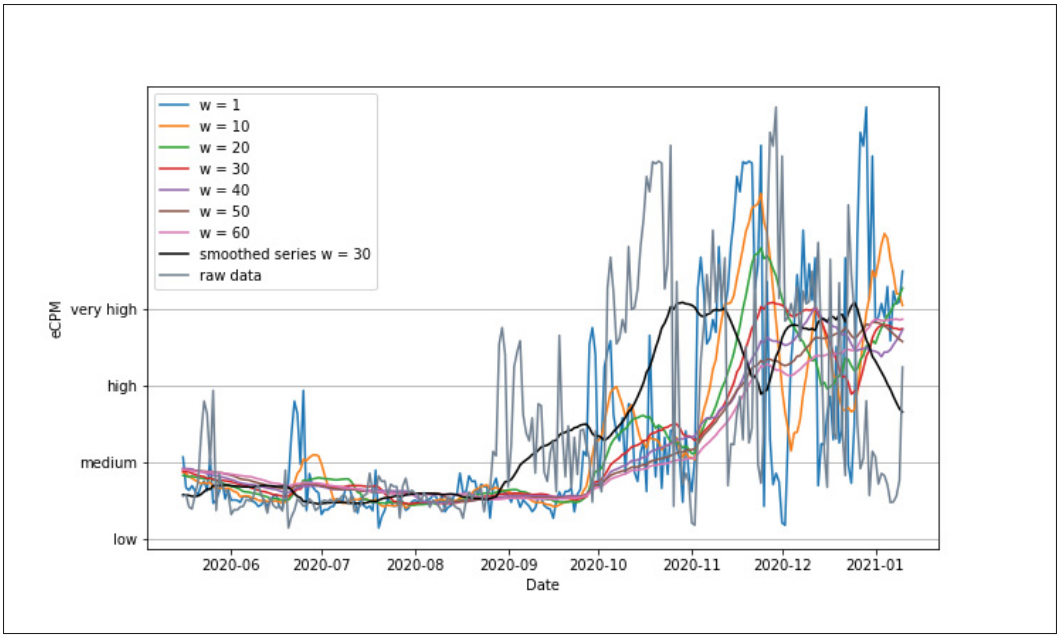


Figure 3.11 Forecast results using ARIMA(0, 1, 0) model with different smoothing windows for 970x250 ad size

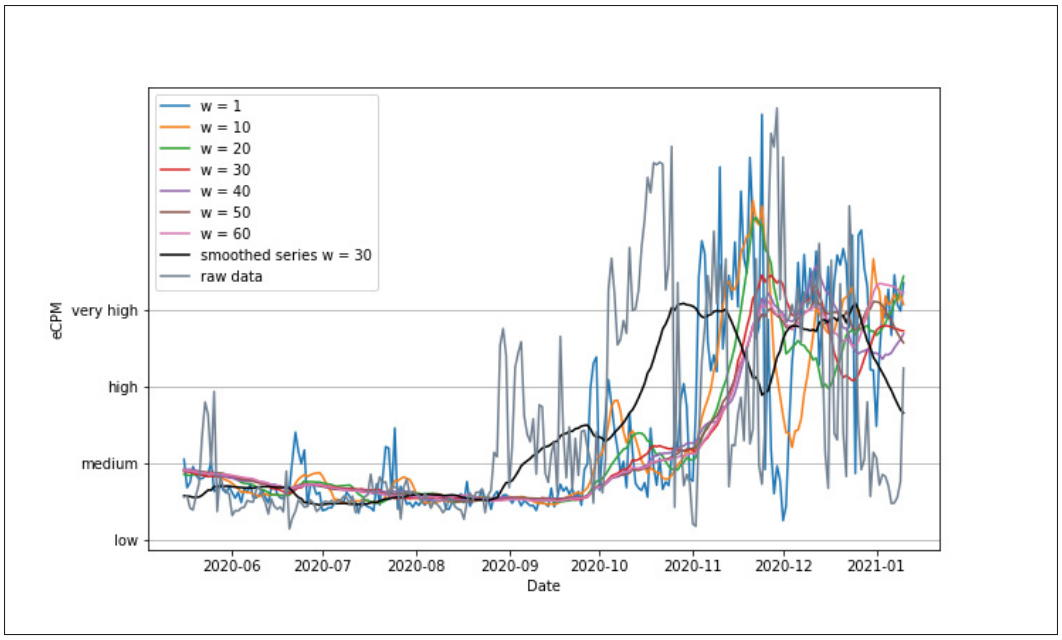


Figure 3.12 Forecast results using ARIMA(1, 1, 0) model with different smoothing windows for 970x250 ad size

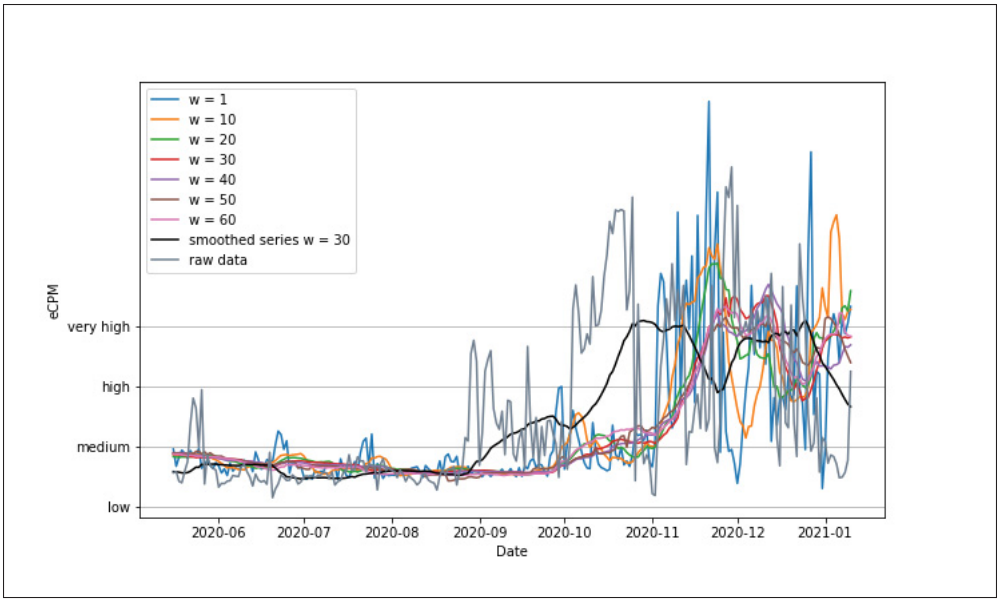


Figure 3.13 Forecast results using ARIMA(1, 1, 1) model with different smoothing windows for 970x250 ad size

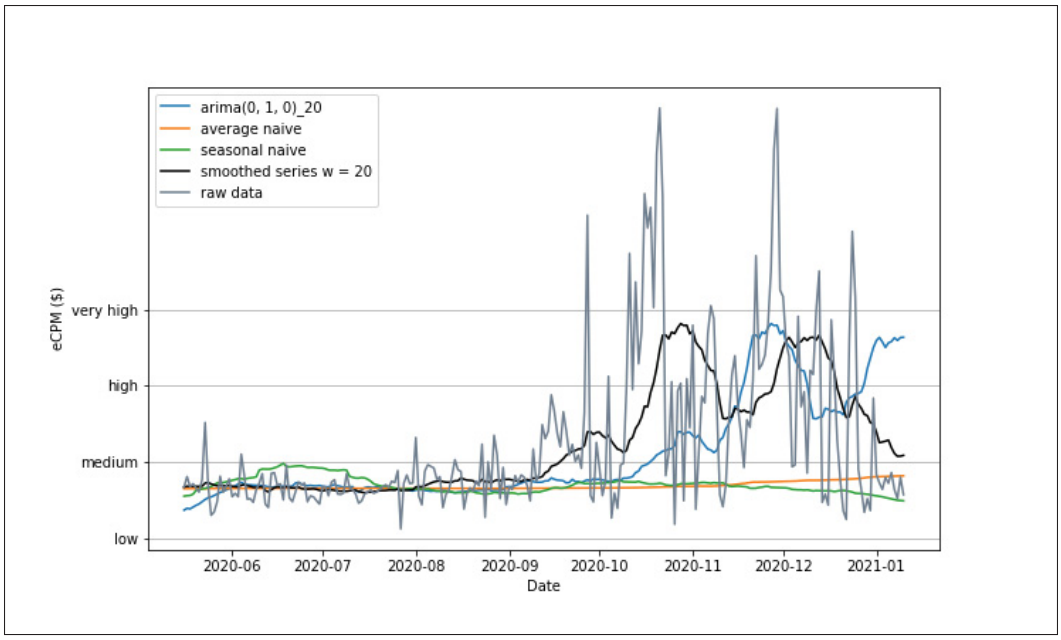


Figure 3.14 Forecast results using best ARIMA model compared with average and seasonal approach for 300x600 ad size

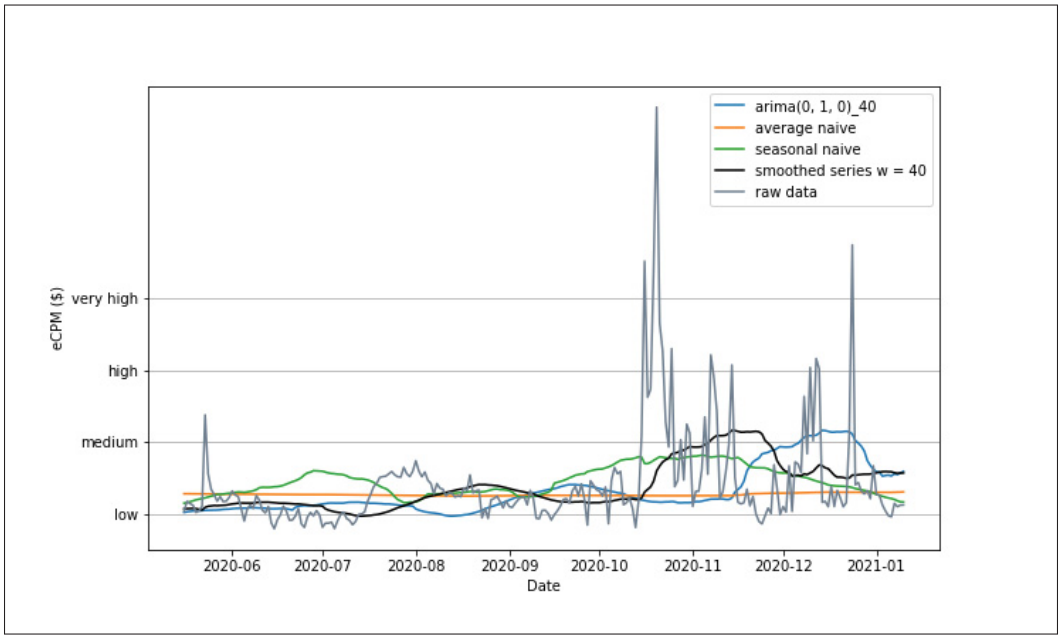


Figure 3.15 Forecast results using best ARIMA model compared with average and seasonal approach for 300x250 ad size

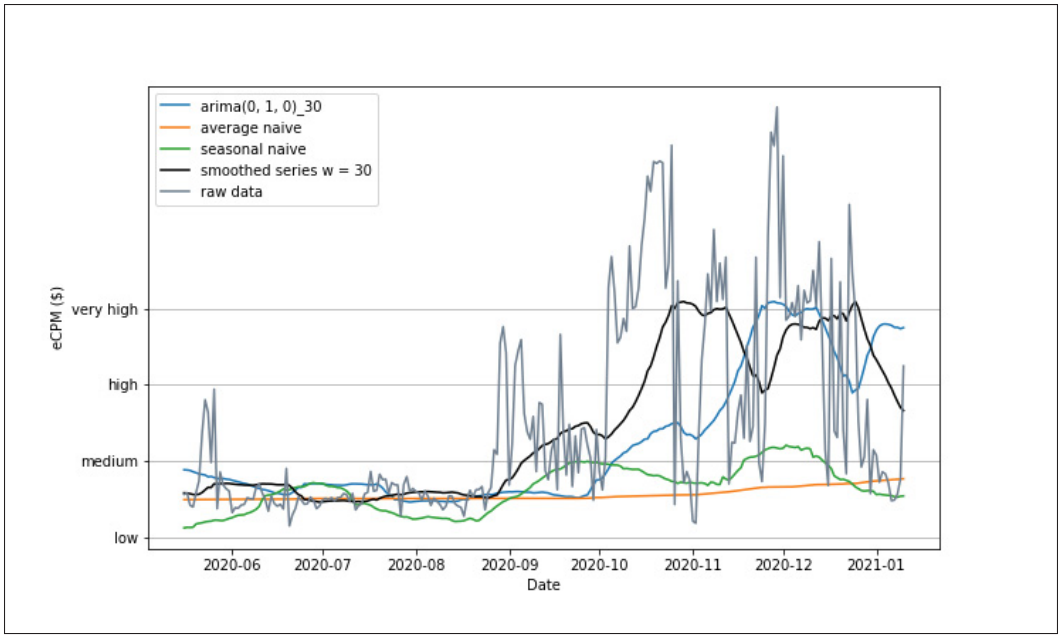


Figure 3.16 Forecast results using best ARIMA model compared with average and seasonal approach for 970x250 ad size

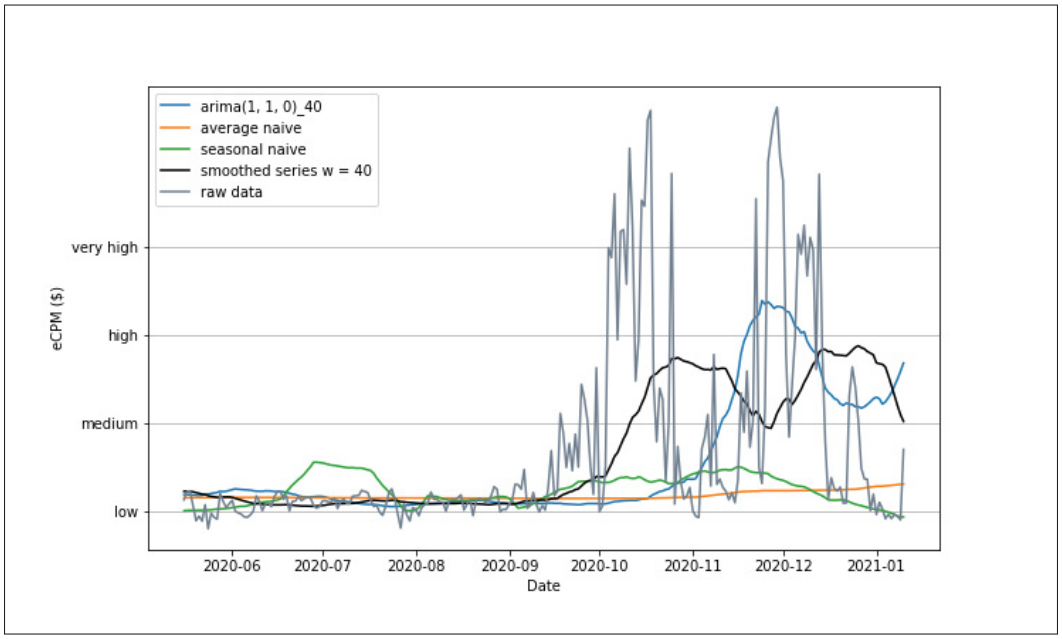


Figure 3.17 Forecast results using best ARIMA model compared with average and seasonal approach for 728x90 ad size

Second, we performed the evaluation for the standard deviation forecast. Table 3.2 shows the results for ARIMA model with different orders compared with the average method and seasonal method. The ARIMA gives the best forecasting performance in term of RMSE for 3 ad size 300x600, 728x90 and 970x250. For ad size 300x250, the seasonal naive model gives the best forecasting performance.

Table 3.2 RMSE and MAPE metrics for forecasting standard deviation

	300x600		**300x250**		**728x90**		**970x250**	
Model	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
arima(1,0,1)	1.917	49.8	1.663	44.2	2.198	45.9	1.925	59.0
arima(0,1,0)	**1.626**	50.2	1.826	68.3	**1.773**	45.1	**1.519**	58.4
arima(1,1,0)	1.886	46.4	1.696	53.2	1.944	44.7	1.640	61.8
arima(1,1,1)	2.007	48.7	1.688	48.8	2.105	52.4	1.887	62.0
average	2.520	68.1	1.846	52.3	2.884	83.6	2.770	76.2
seasonal	2.935	102.7	**1.565**	67.1	2.934	159.9	2.636	102.9

Figure 3.18 shows the forecasting results using ARIMA(0, 1, 0) with smoothing window of 20. The confidence indicator is displayed by different colors based on the thresholds in table 2.4. Overall, the model is able to forecast the general trend but fails to catch up with the sudden change from September 2020 to October 2020.

A similar pattern can be observed in figure 3.20 and figure 3.21 for the forecasting eCPM using ARIMA(0, 1, 0) with smoothing window of 40 and 30 respectively. The models give an under-forecast for the period 2020-09 and 2020-10. The confidence indicator is also unable to forecast the sudden change in volatility.

Figure 3.19 shows the forecasting eCPM using the seasonal naive approach for ad size 300x250. The result shows evidence of annual seasonality for this ad size where the eCPM increase for the period from October 2020 to November 2020 which is similar to last year.

In summary, the confidence indicator can suggest the high volatility and low volatility of the market for most of the periods. During low volatile periods, the publishers can expect the actual eCPM to be close to the forecast value. While, in high volatile periods, it is less likely that the

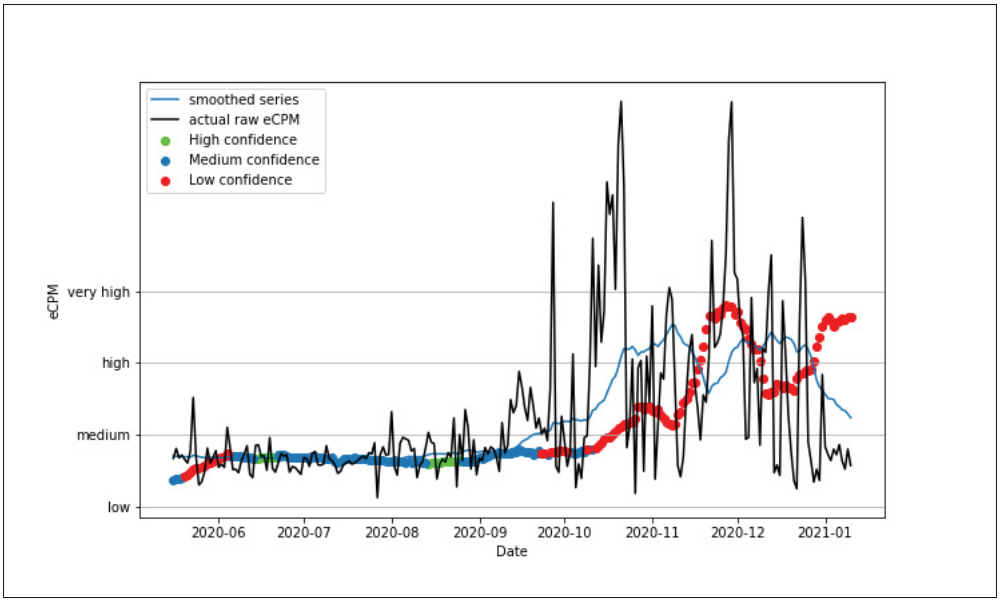


Figure 3.18 Forecast eCPM using ARIMA(0, 1, 0)_20 with confidence indicator for 300x600 ad size

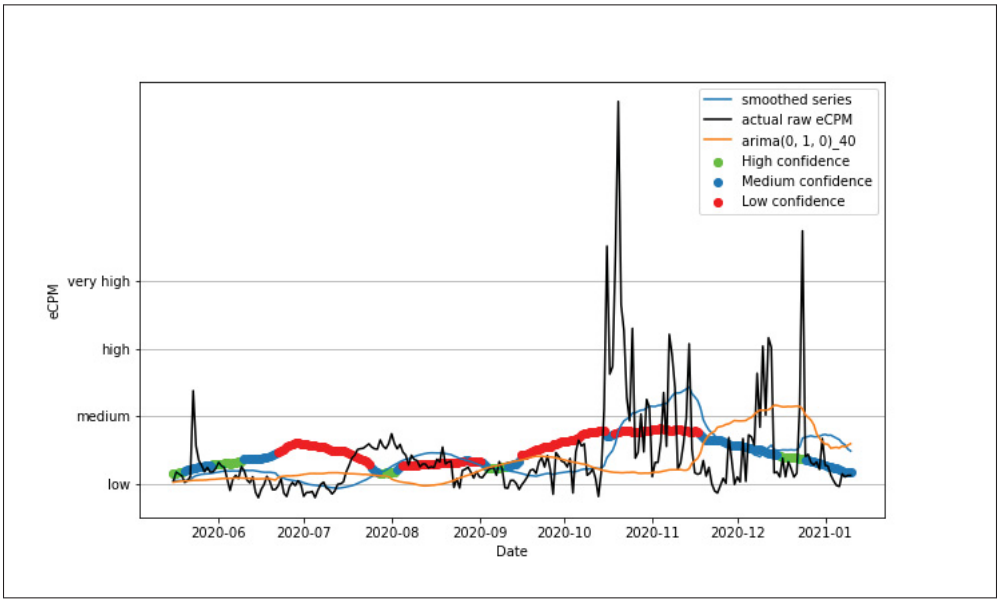


Figure 3.19 Forecast eCPM using seasonal naive model with confidence indicator for 300x250 ad size. The seasonal naive method gives better forecast results compared with the ARIMA(0, 1, 0)_40 for ad size 300x250

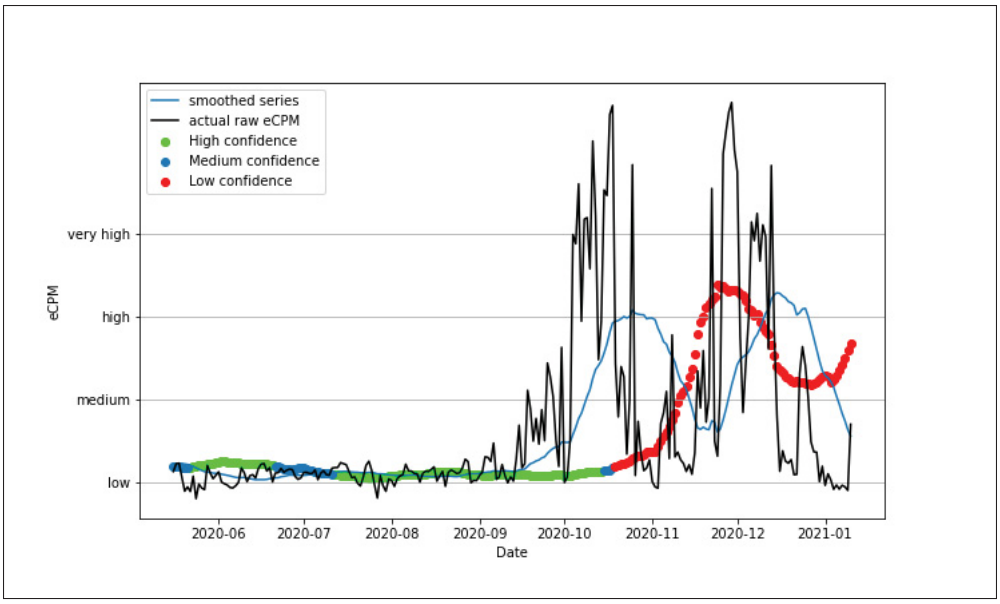


Figure 3.20 Forecast eCPM using ARIMA(1, 1, 0)_40 with confidence indicator for 728x90 ad size

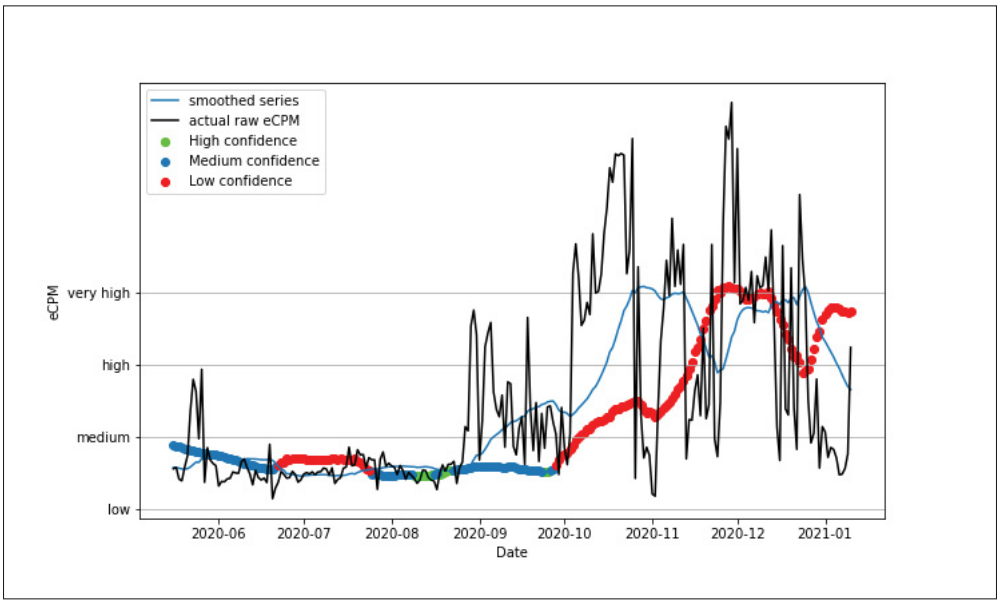


Figure 3.21 Forecast eCPM using ARIMA(0, 1, 0)_30 with confidence indicator for 970x250 ad size

actual eCPM will be around forecast value. When there is a sudden change in eCPM, the models fail to detect the volatility as well as a correct prediction for the mean eCPM. In our study, the

available data is from the end of 2018 to 2020 and the training data for the evaluation process is less than 2 year period. Since this timespan is relatively short considering we make a prediction for the next 30 days, it is hard to have a significant model. Due to the pandemic in 2020, we witness an unprecedented shift from traditional markets to online markets, the surge in eCPM in late 2020 may be an outlier when we consider the broader timespan. Thus, it is challenging to give a correct forecast for this period.

Since there is room for improvement and the re-evaluation process is required when we gather more data, we implemented a framework that allows us easily update the model and deploy it to production. The high-level topology of how the training and forecast process work in production at M32 Connect are described in the next section.

3.3 Moving towards production

In this section, we describe how the process of training and forecasting is integrated into the production pipeline at M32. Figure 3.22 shows an overview of the process. The main components in the pipeline are:

- **Data orchestration** is responsible for pulling data from the ad platform. The data include the total impressions, revenues and, other metrics which serve different purposes at M32. This information is then saved in the data warehouse. This script is scheduled to run daily by a **scheduler**.
- A **training** script queries the data from the data warehouse and starts the training process to find optimal parameters for the selected model from the evaluation process in section 3.2. Then, a forecast for expected eCPM in the next 30 days and its confidence level is made. The forecast result is saved into a table in the data warehouse.
- A **dashboard** is a customer-facing application where all information about ad revenue and forecast are shown in a comprehensive visualization. This application sends queries to the data warehouse to get relevant data including our forecast results and displays them to the clients.

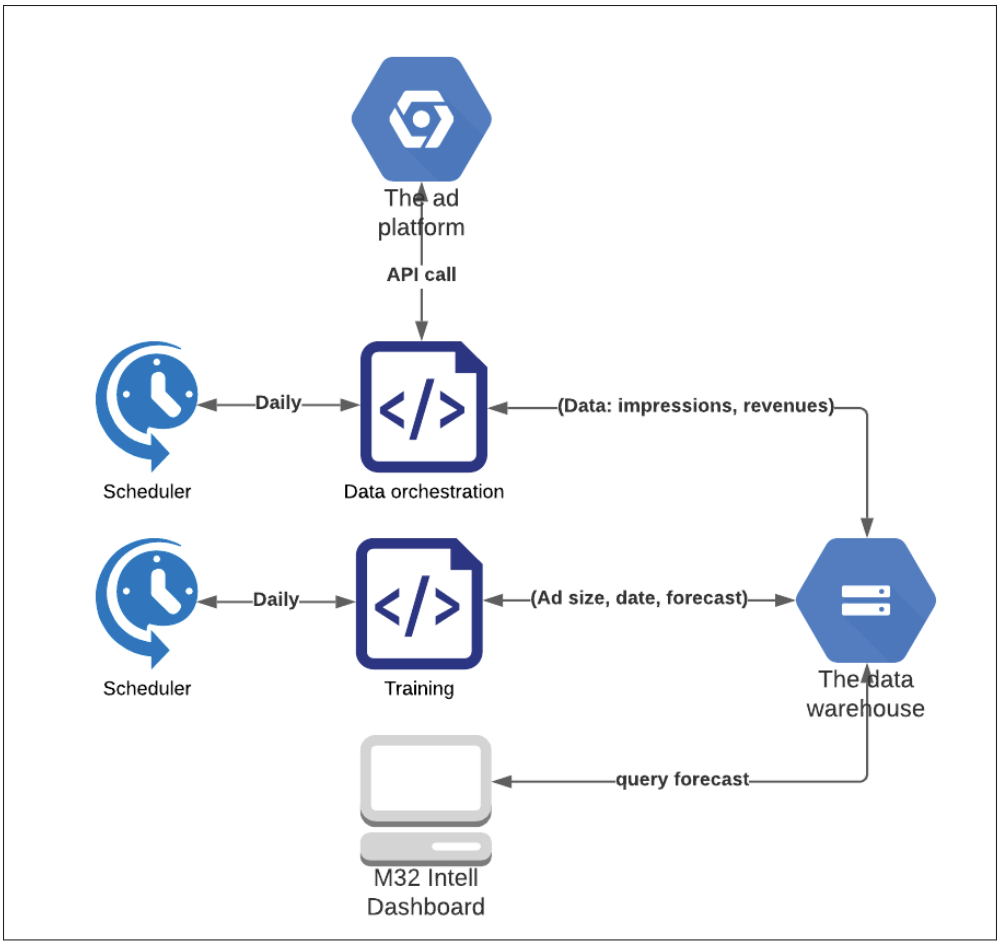


Figure 3.22 Forecasting process on production at M32 Connect

The training script houses most of our works regarding training and forecasting machine learning models. More specifically, it has the following features to allow future changes and extensions:

- The script is a python command-line interface (CLI) program that accepts a report date and a list of publishers as input parameters. The report date is usually the date on which the script is executed. However, it can be configured to be the date in the past for backtesting purposes. Only eCPM data prior to the configured date is pulled for training. At the time of writing, the program is in dry run mode and the list of publishers is configured to be PublisherA and PublisherB.

- The best model for each ad size selected from the evaluation process is saved in a JavaScript Object Notation (JSON) configuration file. In brief, the configuration file contains multiple entries where each entry is the set of order p, d, q and the window size w for each ad size.
- The ARIMA model is implemented in a separate python module. It implements two interfaces: a training method and a forecasting method. The training method fits the model on the historical data while the forecasting method returns the forecast eCPM and the confidence level for the next 30 days. For other methods than ARIMA, they need to follow the same interfaces.
- After the training and forecasting, we save the following information to the data warehouse: publisher website, ad size, report date, forecast date, forecast eCPM, and confidence level.

Details in the shape of pseucode and Python scripting are shown in Appendix I. In the next section, we will summarize our work and give recommendations for further improvements.

CONCLUSION AND RECOMMENDATIONS

In this thesis, we attempted to forecast the expected eCPM in the next 30 days for various ad sizes from PublisherA using the historical series of eCPM in the past 2 years. The study shows that for most ad sizes (300x600, 728x90, and 970x250), there is no statistically significant correlation and clear pattern in the series. Thus, the best forecast for these series is using the average of most recent eCPM values. In contrast, the ad size 300x250 shows an annual seasonal pattern where a forecast using an eCPM from the same date last year shows improvement over a random walk model.

Next, we developed a confidence indicator that suggests market volatility over a period of 30 days. The confidence indicator helps publishers adjust their expectations when interpreting the forecast data. When the indicator shows high confidence, publishers can expect the market is stable and the future eCPM will be around the forecast eCPM. In contrast, when the indicator shows low confidence, the publishers can expect high volatility in the market and there can be large errors in the forecast value.

In this study, the proposed models still have large errors and could not predict the sudden change in eCPM. To improve the models further, we suggest the following actions:

- Re-assess the models as we receive data. As more data arrive, there can be new correlations or seasonality with higher statistical significance. The period for reassessment can be six months. More frequent assessments should not be necessary since the new data is not enough to make a statistical impact on the model.
- Study the eCPM from a group of publishers in the same category. The eCPM of publishers from the same category tend to experience the same trend. A sudden change in eCPM from one publisher can be a leading indicator to predict the future eCPM for other publishers in the same group.

APPENDIX I

IMPLEMENTATION

1. Training algorithm

Pseudo code of the training program is shown in algorithm I-1. The program accepts a report date and a list of publishers as input parameters. The forecast results is saved in the data warehouse.

Algorithm-A I-1 Training program

```
Input: Report date d and list of publishers 𝒫
Result: Insert new record to database with report date, publisher, ad size, forecast date,
        forecast eCPM, confidence level

/* these threshold are based on the evaluation results  */
HIGH_THRESOLD ← 0.5
MEDIUM_THRESOLD ← 1
ad_sizes ← ['300x600', '300x250', '728x90', '970x250']
foreach site in publishers 𝒫 do
    foreach size in ad_sizes do
        /* read configuration for each site and size       */
        mean_order ← read_mean_configuration(site, size)
        std_order ← read_std_configuration(site, size)
        /* query eCPM data up to report date for specific ad
           size for publisher's site                       */
        train_data ← query_data_from_data_warehouse(site, size, report_date)
        /* initialize model with pre-configured order      */
        model ← ArimaModel(mean_order, std_order)
        model→train()
        forecast_date, predict_eCPM, predict_std ← model→forecast()
        if predict_std < HIGH_THRESHOLD then
            confidence_level ← 'high'
        else
            if predict_std < MEDIUM_THRESHOLD then
                confidence_level ← 'medium'
            else
                confidence_level ← 'low'
            end if
        end if
        /* save forecast results to data warehouse         */
        save_forecast(site, size, report_date, forecast_date, predict_eCPM, predict_std,
         confidence_level)
    end foreach
end foreach
```

2. ARIMA model

Implementation of ARIMA model is shown in Source Code I.1. Estimation of the parameters of ARIMA model is handled by Statsmodels[1] Python package.

```
import statsmodels.api as sm
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pandas as pd

class ArimaModel:
  def __init__(self, data, smoothing_window=30, order=(1, 0, 0),
               seasonal_order=None, std_order=(0, 1, 0),
               std_seasonal_order=None):
    """
    data: pandas dataframe (index: date, columns: eCPM)
    """
    self.data = data
    self.smoothing_window = smoothing_window
    self.model_fit = None
    self.std_model_fit = None
    self.target_period = 30
    self.order = order
    self.seasonal_order = seasonal_order
    self.std_order = std_order
    self.std_seasonal_order = std_seasonal_order
    self.avg = self.data.rolling(
        window=self.smoothing_window,
        center=False).mean().dropna().reset_index()
    self.std = self.data.rolling(
        window=self.target_period,
        center=False).std().dropna().reset_index()

  def train(self):
    # prepare data,
    # create series with step of self.target_period
    length = len(self.avg)
    idx = [
        i for i in self.avg.reset_index().index
        if (i - length + 1) % self.target_period == 0
    ]
    std_length = len(self.std)
    std_idx = [
        i for i in self.std.reset_index().index
        if (i - std_length + 1) % self.target_period == 0
    ]
    avg = self.avg.iloc[idx].reset_index()['eCPM']
    std = self.std.iloc[std_idx].reset_index()['eCPM']
    model = SARIMAX(
        avg,
        order=self.order,
        seasonal_order=self.seasonal_order)
    std_model = SARIMAX(
        std,
        order=self.std_order,
        seasonal_order=self.std_seasonal_order)
    self.model_fit = model.fit(disp=0)
    self.std_model_fit = std_model.fit(disp=0)
```

[1] https://www.statsmodels.org/stable/index.html

```
    def forecast(self):
      # make 1 prediction ahead
      predict = self.model_fit.forecast(1)
      std_predict = self.std_model_fit.forecast(1)
      target_date = max(
          self.data.index) + pd.Timedelta(
              days=self.target_period)
      return (target_date.strftime('%Y-%m-%d'),
              predict.iloc[0], std_predict.iloc[0])
```

Source Code I.1: `models/arima_model.py`

BIBLIOGRAPHY

Bigler, J. (2019, September, 5). Rolling out first price auctions to Google Ad Manager partners. Consulted at https://www.blog.google/products/admanager/rolling-out-first-price-auctions-google-ad-manager-partners/.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (ed. 5). John Wiley and Sons Inc. Consulted at https://www.wiley.com/en-ca/Time+Series+Analysis:+Forecasting+and+Control,+5th+Edition-p-9781118675021.

Brownlee, J. (2017). *Introduction to Time Series Forecasting With Python* (ed. 1.3).

Chahuara, P., Grislain, N., Jauvion, G. & Renders, J.-M. (2017). Real-Time Optimization of Web Publisher RTB Revenues. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi: 10.1145/3097983.3098150.

Dickey, D. A. & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, 74(366), 427–431. Consulted at http://www.jstor.org/stable/2286348.

Espinola, J. C. R., Nogales, F. J. & Conejo, A. J. (2003). ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3), 1014-1020. doi: 10.1109/TPWRS.2002.804943.

Gilbert, K. (2005). An ARIMA supply chain model. *Management Science*, 51, 305-310. doi: 10.1287/mnsc.1040.0308.

Graepel, T., Candela, J. Q., Borchert, T. & Herbrich, R. (2010). Web-Scale Bayesian Click-through Rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, (ICML'10), 13–20.

He, X. et al. (2014). Practical Lessons from Predicting Clicks on Ads at Facebook. *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, (ADKDD'14), 1–9. doi: 10.1145/2648584.2648589.

Hyndman, R. J. & Athanasopoulos, G. (2018a). *ARIMA models* (ed. 2). Melbourne, Australia: OTexts. Consulted at https://otexts.com/fpp2/.

Hyndman, R. J. & Athanasopoulos, G. (2018b). *Moving averages* (ed. 2). Melbourne, Australia: OTexts. Consulted at https://otexts.com/fpp2/.

Kulesza, P. (2019, July, 2). First-Price vs. Second-Price Auction – What Are the Main Differences? Consulted at https://yieldbird.com/first-price-vs-second-price-auction/.

Lin, C.-C., Chuang, K.-T., Wu, W. C.-H. & Chen, M.-S. (2016). Combining Powers of Two Predictors in Optimizing Real-Time Bidding Strategy under Constrained Budget. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, (CIKM '16), 2143–2148. doi: 10.1145/2983323.2983656.

Liu, M., Yue, W., Qiu, L. & Li, J. (2020). An Effective Budget Management Framework for Real-Time Bidding in Online Advertising. *IEEE Access*, 8, 131107-131118. doi: 10.1109/ACCESS.2020.2970463.

Palachy, S. (2019, April, 8). Stationarity in time series analysis. Consulted at https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322.

Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W. & Yu, Y. (2019). Deep Landscape Forecasting for Real-time Bidding Advertising. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, abs/1905.03028. Consulted at http://arxiv.org/abs/1905.03028.

Ren, K. et al. (2018). Bidding Machine: Learning to Bid for Directly Optimizing Profits in Display Advertising. *IEEE Transactions on Knowledge and Data Engineering*, abs/1803.02194. Consulted at http://arxiv.org/abs/1803.02194.

Richardson, M., Dominowska, E. & Ragno, R. (2007). Predicting Clicks: Estimating the Click-through Rate for New Ads. *Proceedings of the 16th International Conference on World Wide Web*, (WWW '07), 521–530. doi: 10.1145/1242572.1242643.

Wodecki, A. (2020). The Reserve Price Optimization for Publishers on Real-Time Bidding Online Marketplaces with Time-Series Forecasting. *Foundations of Management*, 12.

Wu, W. C.-H., Yeh, M.-Y. & Chen, M.-S. (2015). Predicting Winning Price in Real Time Bidding with Censored Data. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (KDD '15), 1305–1314. doi: 10.1145/2783258.2783276.

Xie, Z., Lee, K.-C. & Wang, L. (2017). Optimal Reserve Price for Online Ads Trading Based on Inventory Identification. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (ADKDD'17). doi: 10.1145/3124749.3124760.

Yuan, S., Abidin, A. Z., Sloan, M. & Wang, J. (2012). Internet Advertising: An Interplay among Advertisers, Online Publishers, Ad Exchanges and Web Users. *CoRR*, abs/1206.1754. Consulted at http://arxiv.org/abs/1206.1754.

Yuan, Y., Wang, F., Li, J. & Qin, R. (2014). A survey on real time bidding advertising. *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics*, pp. 418-423. doi: 10.1109/SOLI.2014.6960761.