Learning Framework for eCPM Prediction in Online Advertisement

by

Marwa Tageldien Thabet ABDALLAH

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

MONTREAL, FEBRUARY 11, 2022

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 "FEBRUARY 1ST, 2022"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

FOREWORD

My master's project was a great journey that was full of several challenges. This project takes 30 credits out of my 45 master's credits at *École de technologie supérieure*. All of this wonderful time was under the supervision of my great supervisor professor Pascal Giard. He makes these hard times easier than expected. On the other hand, another great support to completing this project came from the Industrial Partner since this thesis is part of the Industrial Partner application. They supported us with useful information that helped us in trying to maximize the revenues of the web publishers.

ACKNOWLEDGEMENTS

I would like to thank my master's supervisor professor Pascal Giard that he gave me the chance to be one of his team members. He is always supporting me and all the team members in everything he could possibly do, whether at our work or our lives. Without his support, I would not have been able to move forward with my master's thesis.

I also would like to thank the Industrial Partner for making me one of their project team members during my master's. I would like to thank them for their support and help with the knowledge and the information that they have to complete our work. Without their efforts, we would not have been able to accomplish many things in this project. Deepest thanks to my lab-mates for their support and the great time that we spend together.

All thanks, and love, to my dear husband. Without his constant support, I would not have been able to move on in my career and in my life. Thanks to my lovely husband, my son, and my family for the great support.

Cadre d'apprentissage pour la prédiction de l'eCPM dans la publicité en ligne

Marwa Tageldien Thabet ABDALLAH

RÉSUMÉ

De nos jours, la publicité en ligne devient l'outil le plus populaire pour acheter et vendre tous les produits. Surtout après l'apparition de COVID-19, la publicité en ligne est devenue un grand défi pour tous les annonceurs et éditeurs de sites Web afin d'augmenter leurs revenus. Tous les annonceurs web cherchent à acheter des emplacements publicitaires vides à bas prix. En même temps, tous les éditeurs de sites Web cherchent à vendre leurs espaces publicitaires et à obtenir des revenus élevés. La vente des espaces publicitaires des éditeurs se fait en temps réel sur une plateforme de Real-Time Bidding (RTB). Il n'y a donc pas de prix spécifique pour la vente des espaces publicitaires des éditeurs. Le prix de l'espace publicitaire est spécifié en fonction de la demande du marché à un moment donné. Les éditeurs souhaitent connaître le prix optimal de l'espace publicitaire qui leur permettra d'obtenir les meilleurs revenus. Dans ce mémoire, nous essayons d'aider les éditeurs à augmenter leurs revenus en essayant de prévoir le prix moyen des espaces publicitaires. Nous essayons de prévoir le Effective Cost per Mille (eCPM) moyen autour d'une certaine date pour les 7, 14 et 30 prochains jours à partir d'aujourd'hui en nous basant sur l'historique du eCPM. Nous avons utilisé différents modèles pour tenter d'atteindre cet objectif. Dans cette thèse, nous avons utilisé les modèles K-Nearest Neighbors (KNN), Multi Layer Perceptron (MLP), et Long Short-Term Memory (LSTM) sur quatre jeux de données différents en essayant d'atteindre la meilleure précision du modèle.

Mots-clés: Publicité en ligne, enchères en temps réel, eCPM, KNN, MLP, LSTM

Learning Framework for eCPM Prediction in Online Advertisement

Marwa Tageldien Thabet ABDALLAH

ABSTRACT

Nowadays, online advertising becomes the most popular tool for buying and selling all products. Especially after the appearance of COVID-19, online advertising has become a big challenge for all web advertisers and web publishers to increase their revenues. All web advertisers seek to buy empty ad slots at low prices. At the same time, all web publishers seek to sell their ads spaces and achieve high revenues. Selling publishers ad space happens in real-time through Real-Time Bidding (RTB) platform. So, there is no specific price for selling publisher's ads spaces. The ad space price is specified based on the market demand at a time. The publishers wish if they can know the optimal ad space price that achieves the highest revenue. In this thesis, we are trying to help publishers to increase their revenues by trying to predict the average Effective Cost per Mille (eCPM) price for ads spaces. We are trying to forecast the average eCPM around certain date for the next 7, 14, and 30 days from today based on the eCPM history. We used different models trying to achieve this goal. In this thesis, we used K-Nearest Neighbors (KNN), Multi Layer Perceptron (MLP), and Long Short-Term Memory (LSTM) models on four different datasets trying to reach the best model accuracy.

Keywords: Online advertisement, real-time bidding, eCPM, KNN, MLP, LSTM

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AdaGrad Gradient Descent Adaptive Gradient Algorithm.

ANN Artificial Neural Network.

AR Autoregressive Model.

ARIMA Autoregressive Integrated Moving Average.

ARMA Autoregressive Moving Average Model.

CNN Convolutional Neural Networks.

CPM Cost Per Mille.

DBSCAN Density-Based Spatial Clustering of Applications with Noise.

DFFNN Deep Feed Forward Neural Networks.

DMP Data Management Platform.

DSP Demand-Side Platform.

eCPM Effective Cost per Mille.

GRU Gated Recurrent Unit.

KNN K-Nearest Neighbors.

LOF Local Outlier Factor.

LSTM Long Short-Term Memory.

MA Moving Average Model.

MLP Multi Layer Perceptron.

MSE Mean Square Error.

Multivariate ARIMA Multivariate Autoregressive Integrated Moving Average.

NMSE Normalized Mean Square Error.

PSO Particle Swarm Optimization.

PSO-ETC-RAP PSO based explore-then-commit algorithm with risk-aware pricing.

ReLU Rectified Linear Activation Function.

Resnet Residual Neural Network.

RF Random Forests.

RMSProp Root Mean Square Propagation.

RNN Recurrent Neural Network.

RNNs Recurrent Neural Networks.

RTB Real-Time Bidding.

SSP Supply-Side Platform.

STSC Subsequence Time-Series Clustering.

SVM Support Vector Machines.

Tanh Hyperbolic Tangent Activation Function.

TS-PF-RAP Thompson sampling with particle filter and risk-aware pricing.

TS-PF-TB Thompson sampling with particle filter and only modeling the top bid.

VAR Vector Auto-Regression.

INTRODUCTION

Online advertising become the most popular way of trading recently. From people side, buying products online becomes easier and faster. From web advertisers and web publishers, online trading becomes more challenging to increase their revenues. Web advertisers are the owners of the products. They want to show their products on an empty ad slot on the website. While the web publishers are the owners of the websites. They want to sell their empty ad slots to web advertisers. Both of them, advertisers and publishers, are want to increase their revenues.

There are two ways to sell publishers' inventories: direct process and programmatic process. In the direct process, publishers deal directly with advertisers. They discuss all the ad proprieties and set a suitable price for the ad inventory. In the programmatic process, publishers and advertisers do not communicate directly. Advertisers and Publishers use platforms to facilitate the buying and selling process. Advertisers and publishers communicate using Demand-Side Platform (DSP) and Supply-Side Platform (SSP) platforms respectively through the Ad Platform. The Ad Platform facilitates the communication between them. The buying and selling process happens through the RTB platform. RTB is a protocol that facilitates the buying ad selling process between advertisers and publishers. In this process, the publishers specify the reserve price, the suitable price for the ad inventory. Then, the advertisers make bids for the empty ad inventory and the winner advertiser is selected through holding an auction.

The advertisers set a price to buy the empty ad inventory and the winner advertiser pays this price for every thousand impressions. This price is called Cost Per Mille (CPM). The advertisers use CPM to estimate the cost of their campaigns. For publishers, they use the eCPM to calculate the effective revenue for every thousand impressions. So, the publishers seek to forecast the future average eCPM based on the past average eCPM to increase their revenues.

In this thesis, our main concern is the publisher's goals. So, in this thesis, we are seeking to predict the average eCPM around certain date for the next 7, 14, and 30 days based on the history

of the eCPM time-series. To achieve this goal, we used different models, machine learning and neural network models, trying to reach the best model accuracy. The models that we used are: KNN, MLP, and LSTM models. We compared the results after testing the LSTM and the MLP models on the four datasets and selected the suitable one that achieved the minimum error. Then we compared the selected model with the KNN model. After all the comparisons and analysis we concluded that the LSTM model is the best model on 300×250, 300×600, and 728×90 datasets. The MLP is the best model on the 970×250 dataset.

This thesis consists of five chapters. We will discuss the content of the thesis as follows: We discuss the background of the online advertising process and the background of each model that we used in chapter 1. Also, we added some of the prior-work that related to our work in chapter 2. While in chapter 3, we explain the methodology that we used to achieve our goal. All the experiments are described in chapter 4. Finally, the experiments results are shown and analyzed in chapter 5.

CHAPTER 1

BACKGROUND

1.1 Online Advertisement

Nowadays, the number of internet users around the world increases rapidly (Ha, 2008). Consequently, especially under COVID-19 restrictions, online trading gains more popularity among all people. All publishers in online advertisement seek to sell their inventories at high prices to increase their revenues. On the other hand, the advertisers seek to buy the inventories of publishers with the lowest price. The publisher is the owner of the website, and the advertiser is the owner of the product that wants to advertise it using the empty publisher ad slot.

For the publishers, there are two selling processes: direct process and programmatic process. In the direct process, the publishers sell their inventories by themselves. They deal directly with advertisers. In the programmatic process, the publishers search for a partner or a platform to offer their inventories to an Ad Platform. The Ad Platform is a place where the advertiser can transact on the publisher's inventory. It is a platform that connects SSP with DSP. SSP is a platform that manages the empty ad slot inventory offered to advertisers, it is on the publisher side. While DSP is a platform that manages the buyers that want to bid for the inventory, it is on the advertiser side.

One of the widely used mechanisms to buy the inventories of publishers is the RTB mechanism. In the following subsection, we will provide an overview of RTB.

1.1.1 Real Time Bidding (RTB)

RTB is a protocol that facilitates the communication between the publisher, through the SSP, and the advertiser, through the DSP, using the Ad Platform. It is a process where buyers or bidders can make a bid for an empty ad slot.

As shown in Figure 1.1, once a user visits the website that has an empty ad slot, the Data Management Platform (DMP) can identify the interests and characteristics of this user through the cookie. Then, the publisher sends a request to the Ad Platform via SSP indicating that an impression can potentially be displayed in this particular ad slot. At the same time, the Ad Platform sends the user information to the eligible DSPs. These DSPs ask DMP and know the user interests then start the auction. The advertisers that are connected to DSPs send bids requests to the Ad Platform indicating that they are willing to bid for this impression. Then the RTB decides which advertiser can participate in the auction based on the reserve price and the money that the winner advertiser will pay based on the auction type as will be discussed in the coming subsections. This process is called RTB process.

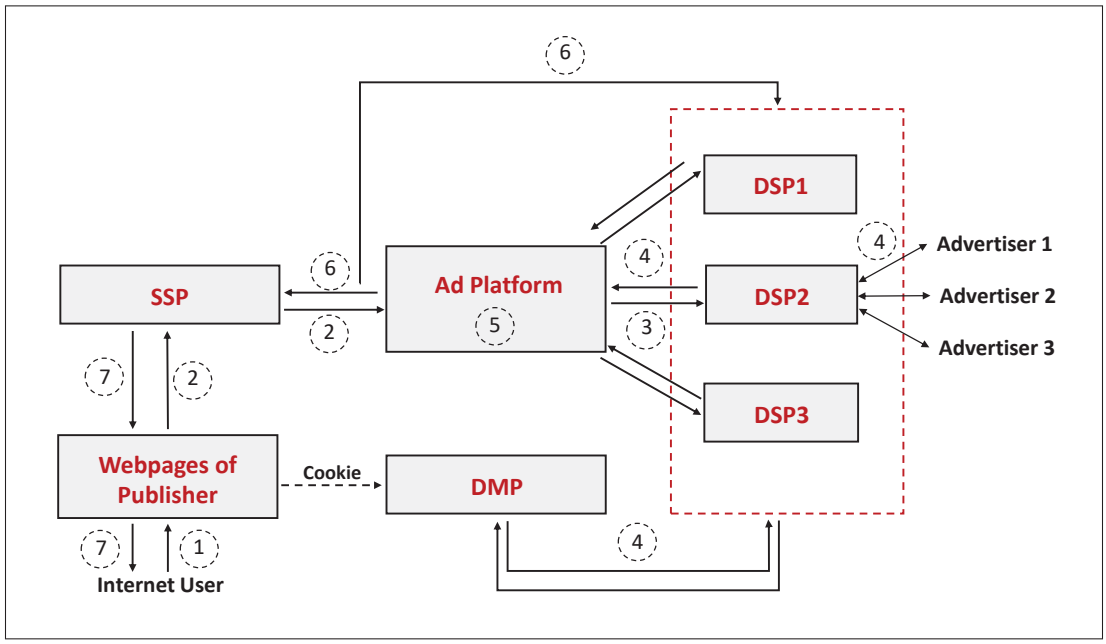

Figure 1.1 The business process of RTB ad delivery
Adapted from Yuan *et al.* (2014b)

The duration of the RTB process, from opening the website by the user till declare the winner advertiser, takes from 10 to 100 milliseconds (Yuan *et al.*, 2014b).

1.1.2 Reserve Price

The reserve price or the **floor price** is the minimum price that publishers will accept from bidders to buy their inventories (Hayes, 2021). If the bids of the bidders were higher or equal to the reserve price, they will participate in the auction, otherwise, these bidders will be discarded (Rosenkranz & Schmitz, 2007). So, the reserve price should be balanced, not being very low since this will reduce the revenues of publishers and not being very high which prevent many advertisers from participating in the auction. Consequently, this will decrease the publisher's revenues.

As shown in figure 1.2, increasing the floor price until a certain point will achieve high revenues but if the floor price increased more than this certain point the revenues will be decreased. So, all publishers need to optimize their floor price to maximize their revenues.

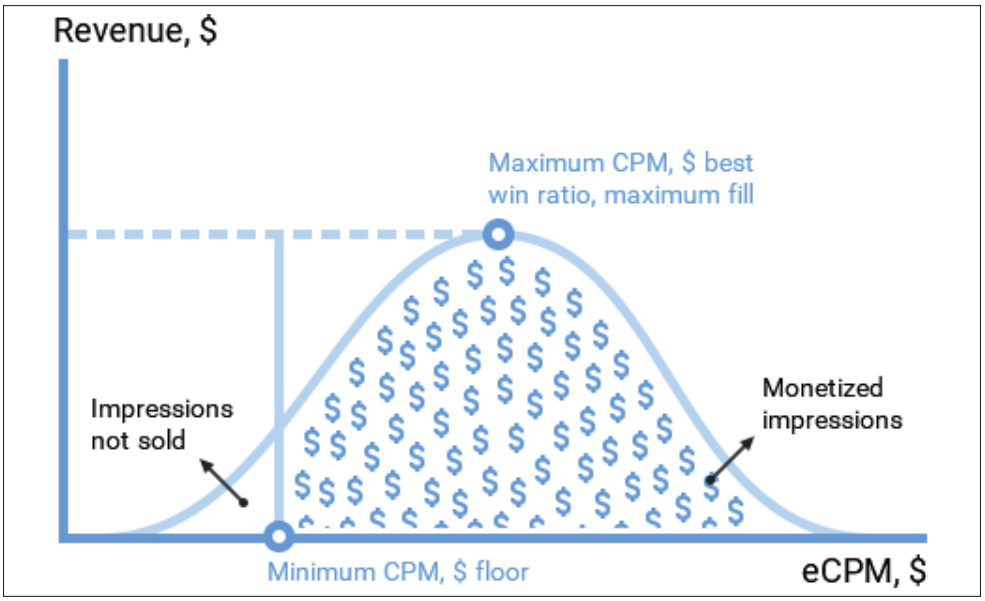

Figure 1.2 The relation between the floor price and the revenue
Taken from steldSSP Company (2021)

1.1.3 Online Auctions

Online auctions are the auctions that held to sell publishers' inventories (the empty ad stots) to advertisers. There are two types of online auctions first-price auction and second-price auction.

- **First-price auction:** in this auction the advertiser with the highest bid will win the auction. In this case, the winner advertiser will pay exactly what he bid.

  As shown in Figure 1.3 on the left side, *advertiser C* bids the highest price so, he wins the auction and pays exactly what he bid. In this case, *advertiser C* will pay 4$ for the inventory.
- **Second-price auction:** in this auction the winner advertiser, the advertiser with the highest bid, will pay the second highest bid not what he bid. The winner advertiser will pay (the second highest bid + 0.01) (INMOBI, 2021, October 15).

  As shown in Figure 1.3 in the right side, *Advertiser C* bids the highest price 4$, so he wins this auction. nevertheless, in this case *Advertiser C* will pay (the second highest bid + 0.01) which is *Advertiser B* bid + 0.01. So, *advertiser C* will pay (3$ + 0.01) = 3.01$ for the inventory.

Figure 1.3 First-price Auction vs. Second-price Auction
Taken from ROXOT (2021)

1.1.4 CPM VS. eCPM

CPM is the price the advertisers pay for every thousand impressions of a certain ad.

Equation 1.1 is the equation of computing CPM:

$$CPM = \frac{Costo f theCampaign}{NumberofTotalImpressions} \times 1000 \tag{1.1}$$

For example, if the advertiser budget for a campaign is 50$ and the advertiser ad generates 10,000 impression, then the advertiser will pay 5$ for the 1000 impression ((50 / 10,000) × 1000 = 5$).

eCPM is used by publishers to determine the revenue achieved from a thousand impressions of an ad campaign.

Equation 1.2 is the equation of computing eCPM:

$$eCPM = \frac{EstimatedEarnings}{NumberofTotalImpression} \times 1000 \tag{1.2}$$

For example, if an ad campaign generated 100$ revenue after receiving 10,000 impressions, the eCPM would be 10$ ((100 / 10,000) × 1000 = 10$). This means that the publisher can earn 10$ per 1000 impressions.

In this work, we are going to predict the average eCPM around certain date.

1.2 Long Short-Term Memory (LSTM)

LSTM is a Recurrent Neural Network (RNN) architecture that used in the field of deep learning. It is introduced in 1997 (Hochreiter & Schmidhuber, 1997). LSTM is designed to solve the problems that exist in RNN architecture since RNN was suffering from short-term dependencies and vanishing gradient problem (Phi, 2021). LSTM is capable of learns long-term dependencies

between samples (Greff, Srivastava, Koutník, Steunebrink & Schmidhuber, 2016). It is has gates that control the information flow.

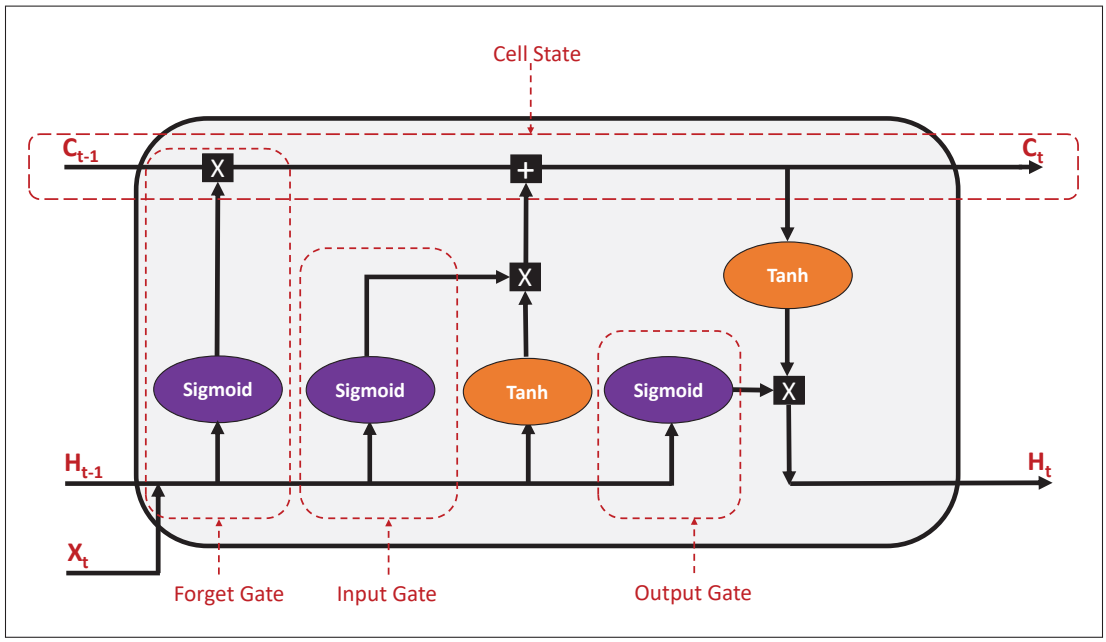

Figure 1.4 LSTM Cell

As shown in Figure 1.4, the common LSTM unit consists of a cell state and three gates (input gate, output gate, and forget gate).

- **Cell state** is like a memory because is it transfer the relevant information through the processing sequence. The relevant information is decided by the three gates.
- **Forget gate** decides what is the relevant information to keep from the previous steps. It contains the information from the previous hidden state (H_{t-1}) and the current input (X_t).
- **Input gate** decides the relevant information that will be added from the current step (H_{t-1} and X_t) then updates the cell state.
- **Output gate** determines the next hidden state. The output from the output gate is the new hidden state (H_t) that will be passed to the next time step.

In our work, we will use the vanilla-LSTM model. This model predicts the future average eCPM information based on the history of the past eCPM information. Based on the dimensions of the input data, we can classify the vanilla-LSTM model into univariate and multivariate

vanilla-LSTM models. Based on the datasets that we have in our work, we have only one feature. So, we will employ a univariate vanilla-LSTM model as will be explained explicitly in the coming chapters.

1.3 Multi-Layer Perceptron (MLP)

MLP is a field of Artificial Neural Network (ANN) that used in solving several problems such as pattern recognition and classification (Noriega, 2005). It is designed to approximate any continuous function and trying to solve the non-linear separable problems. As shown in Figure 1.5, MLP architecture consists of one input layer, at least one or more hidden layers, and one output layer. Each layer consists of number of interconnected neurons or nodes. These nodes are connected by weights and output signals. The neurons are trained using the backpropagation algorithm (Rumelhart, Hinton & Williams, 1985) since it is the most straightforward algorithm that is used to train the MLP.

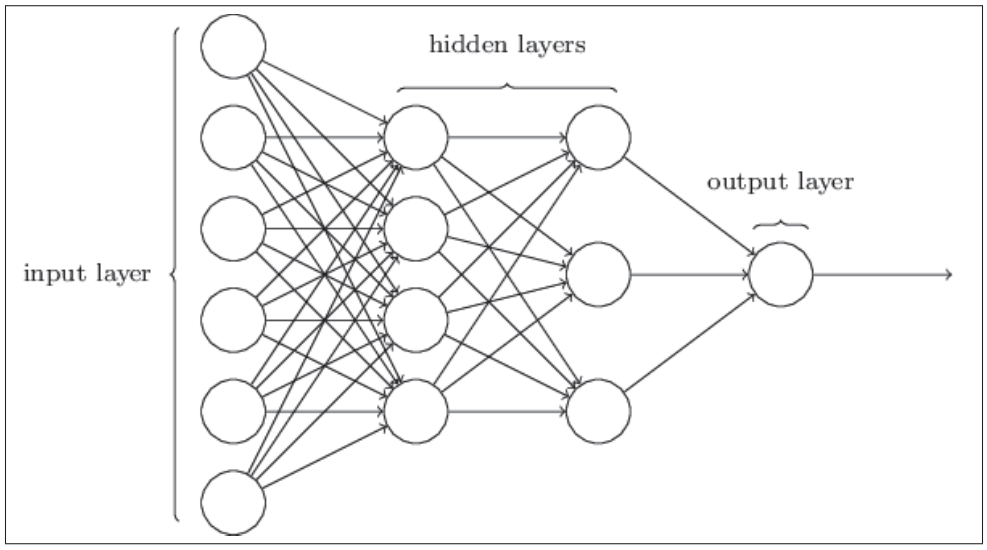

Figure 1.5 MLP Architecture
Taken from Karim (2021)

In addition, MLP is a fully connected architecture since each node is connected to all nodes in the next and previous layer (Gardner & Dorling, 1998). Also, MLP is a feedforward ANN

because the data flow in a forward direction from the input layer to the hidden layers and finally to the output layer (no feedback connections).

1.4 K-Nearest Neighbour (KNN)

KNN is a supervised machine learning algorithm that is used to solve classification and regression problems (Kramer, 2013). This algorithm is based on the idea of similarity. It works on finding the distances between a query and all the data points in the dataset. There are several methods for calculating the distance between data points. The most popular method is the **Euclidean Distance** (1.3). This method finds the distance between two different data points in a multidimensional space where p and q are two data points in euclidean n-space (n) and (q_i, p_i) are the euclidean vectors.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1.3}$$

Selecting the number of the nearest Neighbors to the data point is based on the K value. For example, if the K value equals 5, it will look for 5 nearest Neighbors to that data point. we can choose the suitable value of K by running the model several times and choose the K value that helps the model achieves the least error.

Some of the KNN algorithm advantages are:

- It is a simple algorithm and easy to implement.
- There is no need to build a complicated model and tune several parameters.
- It works well on non-linear problems.
- It is a versatile algorithm since it is used to solve several problems such as classification, regression, and search problems.

One of the disadvantages of KNN is it requires storing all the training data. If we have a high number of examples or predictors, this requires high storage.

CHAPTER 2

RELATED WORK

In this chapter, we are going to discuss different prior works that help us achieve our goal. Our main problem has two phases. The first phase (phase 1) is predicting the average eCPM around certain date and the second phase (phase 2) is optimizing the reserve price (the floor price).

To solve phase 1, we shall understand the problem to solve it successfully. The goal is predicting the average eCPM around certain date for the next one week, two weeks, and one month based on the data history that we have. The problem that we face is a time series forecasting problem. So, we will go deeper in the prior work that discusses different time series forecasting problems that used different types of models to achieve their goal as will be explained in section 2.1.

For the second phase, online advertisement has two sides: advertiser side and publisher side. Our main concern is the publisher side, the owner of the website. The publishers are trying to improve their revenues by optimizing the reserve price. Since the reserve price must be balanced because if the reserve price was high, this might led to fewer revenues as explained in subsection 1.1.2. For this reason, we will discuss different literature works that help in increasing the publisher revenues by optimizing the reserve price in section 2.2.

2.1 Time Series Forecasting

Time series forecasting becomes a hot topic and very thorough field of research that appears in recent years (Shelatkar, Tondale, Yadav & Ahir, 2020). Time series is a set of observations for data points that are taken sequentially at time (Box, Jenkins, Reinsel & Ljung, 2015). It is used in several fields like statistics, signal processing, weather forecasting, science, engineering domains, etc. Forecasting in time series is very important in many applications such as business, stock market, etc. It provides the companies and organizations with useful information that is necessary for making important decisions based on the prediction. So, a lot of research is done in this field to help organizations make the right decision.

In (Torres, Hadjout, Sebaa, Martínez-Álvarez & Troncoso, 2021), the researchers discussed the time series forecasting problem in detail. They explained the mathematical fundamentals of the time series. They discussed the characteristics of the three time series components (Trend, Seasonality, and Residuals). In (Torres *et al.*, 2021), the authors explained the most common deep learning architectures that are used with the time series problems. They explained deeply: Deep Feed Forward Neural Networks (DFFNN), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNN). Besides, they showed the different applications that used these deep learning architectures.

Also, in (Mahalakshmi, Sridevi & Rajaram, 2016), the authors discussed in detail the various techniques that are applied for forecasting different types of time series datasets. They discussed four forecasting techniques: traditional forecasting techniques, stochastic forecasting techniques, soft computing based forecasting techniques, and fuzzy based forecasting techniques with the models that should be used in the forecasting. Additionally, they explained the various performance evaluation parameters used for evaluating the predictions of models.

After the previous discussions we can say that time series forecasting methods can be divided into two main categories: traditional statistical methods and methods based on machine learning models (Elsworth & Güttel, 2020). There are several of traditional statistical methods such as Autoregressive Moving Average Model (ARMA), Autoregressive Model (AR), Moving Average Model (MA), Autoregressive Integrated Moving Average (ARIMA), Multivariate Autoregressive Integrated Moving Average (Multivariate ARIMA), and Vector Auto-Regression (VAR). For machine learning methods, there are Support Vector Machines (SVM), Random Forests (RF), KNN, etc. For deep learning methods, there are CNN, LSTM, MLP, etc.

The most popular method in the traditional techniques is ARIMA because it upwards of the other traditional methods in precision, achieves better accuracy in predicting the next lags in time series forecasting, and fits well with time series forecasting problems and seasonal adaptation (Siami-Namini, Tavakoli & Namin, 2018), (Piccolo, 1990). The common method in deep learning methods is LSTM method since it is the most famous method that used in time series forecasting

(Brownlee, 2021b), (Gers, Schmidhuber & Cummins, 2000), (Schmidhuber, Hochreiter et al., 1997), (Schmidhuber, 2015). Therefore, we can say that ARIMA and LSTM models are the most popular techniques in the traditional and deep learning techniques respectively. Nevertheless, which method of these two methods is the suitable one that achieves the best model accuracy in time series prediction?

To answer the previous question, the researchers made several experiments and researches on different datasets to decide which method is the best in time series prediction. In (Siami-Namini *et al.*, 2018) and (Siami-Namini & Namin, 2018), the researchers made comparison between ARIMA and LSTM models in time series forecasting. They extracted the historical monthly financial time series data from Yahoo finance Website. After running various experiments, the authors concluded that LSTM outperforms ARIMA. Since LSTM model improved the prediction by 85% on average compared with ARIMA model.

While in (Yamak, Yujian & Gadosey, 2019), ARIMA model outperforms LSTM model. Since the authors compared between ARIMA, LSTM, and Gated Recurrent Unit (GRU) models in making a time series forecasting. The authors run the experiments using Bitcoin's price dataset. The results shown that ARIMA outperfroms LSTM and GRU and GRU outperforms LSTM.

From the previous discussions, we can not say that ARIMA surpasses LSTM or LSTM surpasses ARIMA in all problems because each problem has it is own criterion. Choosing the suitable model is based on the problem description and the behaviour of the dataset that is used to solve the problem.

Several datasets involve differences and exceptions in the data behaviours. This strange behaviour is called **Anomaly detection** or outlier detection. Anomaly detection in time series data becomes a hot research topic for a long time (Salvador & Chan, 2005).

In recent years, there is a lot of research that used machine learning models trying to detect the anomalies on time series. As the researchers made comparisons between ARIMA and LSTM models in anomalies detection, other researchers try to improve these techniques using

deep neural networks. So, in (Braei & Wagner, 2020), the authors compared the evaluation of statistical, machine learning and deep learning methods in anomaly detection on time series. They studied 20 univariate anomaly detection methods from the three categories. In the statistical approach they used regressive models like: AR, MA, ARMA, ARIMA, etc. For machine learning approach, they used anomalies detection methods like: Subsequence Time-Series Clustering (STSC), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Local Outlier Factor (LOF), Residual Neural Network (Resnet), etc. For deep neural network they selected the most popular methods like: MLP, CNN, Resnet, LSTM, etc.

After all experiments, the results showed that the statistical approaches can achieve better accuracy than the other two approaches in detecting point and collective anomalies with less computational time. While the deep learning approaches can work better than the statistical approaches with a univariate dataset with contextual anomalies, they still require higher computation time than the statistical approaches.

We will describe the relevancy of the reviewed work with our work in section 2.3.

2.2 Floor Price Optimization in Online Advertisement

Reserve-price optimization is the main challenge for publishers to increase their revenues from online Advertisement (Feng, Lahaie, Schneider & Ye, 2020). It is difficult to optimize the reserve price because there are billions of auctions taking place every day in large Ad Platforms. The challenge for the reserve price optimizer is to reach a compromise between being simple to execute in a few milliseconds and at the same time being complex enough to take the merits of auction level data.

In (Austin, Seljan, Monello & Tzeng, 2016), the authors presented the first online, scalable, real-time reserve-price optimizer for RTB. This optimizer sets a reserve price dynamically and provides a scale to billions of auctions every day. They applied an online learning approach and maximized a custom cost function to optimize the reserve price. Furthermore, the authors performed their work in two ways. The first way is **online** for all publishers that use reserve price

optimization on their Ad Platform. The second way is **offline** on a small group of publishers. The results show that using this proposed price-optimizer achieves an increase of 34.4% in revenues.

Another important issue to solve the reserve price problem is understanding the bidder behavior in RTB auctions. The researchers in (Yuan, Wang, Chen, Mason & Seljan, 2014a) presented an empirical study for the bidders' behavior in RTB auction and their impact on setting the reserve price. The researchers wanted to improve the reserve price based on the analysis of real-world data and online experiments on a production platform.

$$\alpha(t+1) = \begin{cases} (1-\varepsilon^t\lambda_h)\alpha(t), & \alpha(t) > b_1(t) \\ (1+\varepsilon^t\lambda_e)\alpha(t), & b_1(t) \geq \alpha(t) \geq b_2(t) \\ (1+\varepsilon^t\lambda_t)\alpha(t), & b_2(t) > \alpha(t) \end{cases} \tag{2.1}$$

The researchers proposed a game-theory-based algorithm called the OneShot algorithm. The algorithm is depicted in (2.1). We can summarize the algorithm as follows: if the winning bid is higher than the reserve price, the reserve price will increase slowly otherwise the reserve price will decrease drastically. From (2.1), we can see that the authors introduced parameters ($\alpha, \epsilon, \lambda$) to control the magnitude of the change under different situations. ϵ is a decay factor with respect to time, allowing the reserve price to converge if needed such that $\epsilon \in (0, 1]$ and $\lambda_h, \lambda_e, \lambda_l \in$ [0,1]. Here λ_h controls the decreasing ratio when the reserve price is too high; λ_e controls the continued exploration when the reserve price is successful, and λ_l controls the increasing ratio when the reserve price is too low. Also, note that α(t) is the reserve price at time t. From the experimental results the authors found that the good values for these parameters are: $\epsilon = 1.0$, $\lambda_h = 0.3$, $\lambda_e = 0.01$ and $\lambda_l = 0.02$. The results obtained show that the OneShot algorithm outperformed the state-of-the-art algorithms significantly in most cases.

On the other hand, as discussed previously in subsection 1.1.3, there are two types of auctions: first-price auction and second-price auction. In the second-price auction, the publishers of the

online advertising put their reserve price for their inventories. All bids that are equal to or higher than the reserve price can participate in the auction otherwise, the bids will be discarded.

Accordingly, as discussed in subsection 1.1.2, the reserve price is an important key for increasing or decreasing the publisher revenues. The publisher must use a specific mechanism that specifies the reserve price accurately to increase their revenues. In (Rhuggenaath, Akcay, Zhang & Kaymak, 2019a), the authors proposed a method for learning optimal reserve prices in the second-price auction. Their experiments were done on limited information where the values of the bids are not revealed and no historical information about the values of the bids is available. Their proposed method is based on the principle of Thompson sampling combined with a particle filter to approximate and sample from the posterior distribution. Thompson sampling (DeepAI, 2021) is a heuristic learning algorithm that chooses an action that maximizes the expected reward for a randomly assigned belief.

The authors proposed two models: Thompson sampling with particle filter and only modeling the top bid (TS-PF-TB) and Thompson sampling with particle filter and risk-aware pricing (TS-PF-RAP). TS-PF-TB is the initial model based only on top bids. While TS-PF-RAP is the extended model that improves the performance of the initial model by exploiting the setting of the second-price auction. These two methods are based on optimization techniques.

The authors used real-life data from ad auction markets. They used the publicly available *iPinYou* dataset, which contains information from the perspective of 9 advertisers on a DSP.

The authors used 4 performance metrics to evaluate the performance of the methods. The 4 performance metrics are: cumulative average return, success rate, revenue rate, and revenue rate given success. The cumulative average return is the main metric to determine the profitability of a strategy. The success rate measures how often reserve prices are set too high. While revenue rate measures the rate at which the top bid is extracted. Lastly, the revenue rate given success measures the rate at which revenue is extracted given that a sale is successful.

The authors compared the algorithm performance with two MAB algorithms (UCB algorithm and EXP3 algorithm). The results showed that TS-PF-RAP generally outperforms the other methods and UCB outperforms EXP3. The TS-PF-RAP achieved better performance compared with UCB because TS-PF-RAP can track changes in the top bid better than UCB. It is better in selecting reserve prices that are closer to the top bid when the gap between the top bid and second bid is large.

While in (Rhuggenaath, Akcay, Zhang & Kaymak, 2019b), the authors proposed a method for learning the optimal reserve price in limited information for the second online auction. This method considered the gap between the winning bid and the second highest bid to take better decisions for the reserve prices. The method is based on a method from computational intelligence called Particle Swarm Optimization (PSO). The proposed method is called PSO based explore-then-commit algorithm with risk-aware pricing (PSO-ETC-RAP).

The PSO-ETC-RAP is suitable for non-stationary environment. This method exploits the dynamic movements of particles in PSO to learn an estimate of the top bid. This estimate is combined with an estimate of the second bid to set a reserve price.

The PSO-ETC-RAP has 4 steps: the first and second steps are exploration phases, the third step is a commit phase, and the fourth step is a test phase.

In the first and second steps (exploration phases), the authors estimate the value of the second highest bid then they estimate a reference value for the winning bid. In the third step (commit phase), the publisher uses the estimated values for the winning bid and the second highest bid to determine a reserve price. The publisher has to use this reserve price for a fixed amount of time. In the fourth step (test phase), a test is done to see if they should start another commit phase or go back and start with another exploration phase.

The authors used real-life data from ad auction markets. They used *iPinYou* dataset. The authors run the experiments on the PSO-ETC-RAP method (the proposed algorithm), TS-PF-TB, UCB, and EXP3 (other public methods).

The results in (Rhuggenaath *et al.*, 2019b) showed that the PSO-ETC-RAP achieves higher performance than the other methods. Since PSO-ETC-RAP was able to track the changes in the top bid better than the other methods. Also, it was better than others in selecting the reserve prices that are closer to the top bid when the gap between the top bid and the second bid is large.

2.3 Relevancy to Our Project

In this section, we highlight the connection between the reviewed work and our work.

In this work, we will focus on phase 1: predicting average eCPM. Since our problem is a time series forecasting problem, we will use some of the machine learning models discussed in section 2.1.

For phase 2, we discussed different techniques that can be used as a starting point in solving phase 2 in the future but in this work we will focus only on solving phase 1.

CHAPTER 3

METHODOLOGY

In chapter 1, we discussed the process that happens between the advertisers and publishers in online advertising. We explained different terms in the online advertisement field such as the reserve price, the Ad Platform, and the RTB. We explained the process that happens via the RTB. Also, we explained the main concept of LSTM, MLP, and KNN models and the main idea of our work in chapter 1. Whereas we discussed the literature that related to our work in chapter 2.

In this chapter, we will discuss how we prepared the different datasets that we have to be suitable before feeding them to the LSTM and MLP models that were explained in chapter 1. Also, we will explain in detail the technique that we used to predict the average eCPM around certain date.

3.1 Data Preparation

In this work we used four datasets with four different ad sizes for one publisher. The four datasets are collected by our Industrial Partner Ad Platform. These datasets have a large number of features. This high dimensionality of features may lead to the well-known problem of curse of dimensionality (Wojtowytsch & Weinan, 2020). According to the business needs, we collected theses datasets based on six features as shown in table 3.1.

Table 3.1 Six Features Used to Aggregate the Four Datasets

Features	Description
Report-Date	We work on data from 01-11-2018 to 26-04-2021.
Ad Size	There are a lot of ad sizes, in this work we focus on four ad sizes (300×250, 300×600, 970×250, and 728×90).
Ad Type	There are two main ad types (header-bidding and Ad Platform). In this work we focus on header-bidding ad type.
Revenue Raw	Is the total revenue on report date by ad size and ad type.
Impression Raw	Is the total impression on report date by ad size and ad type.
Daily eCPM	Is the out of dividing the total revenue raw and total impression raw (total revenue raw / total impression raw).

In the four datasets, we ordered the data by *Report-Date* feature from November 2018 to April 2021. Thereafter, we removed the *Report-Date* feature. This reduces the data from a multivariate to a univariate dataset since the input to the model will be the daily eCPM only. Therefore, we

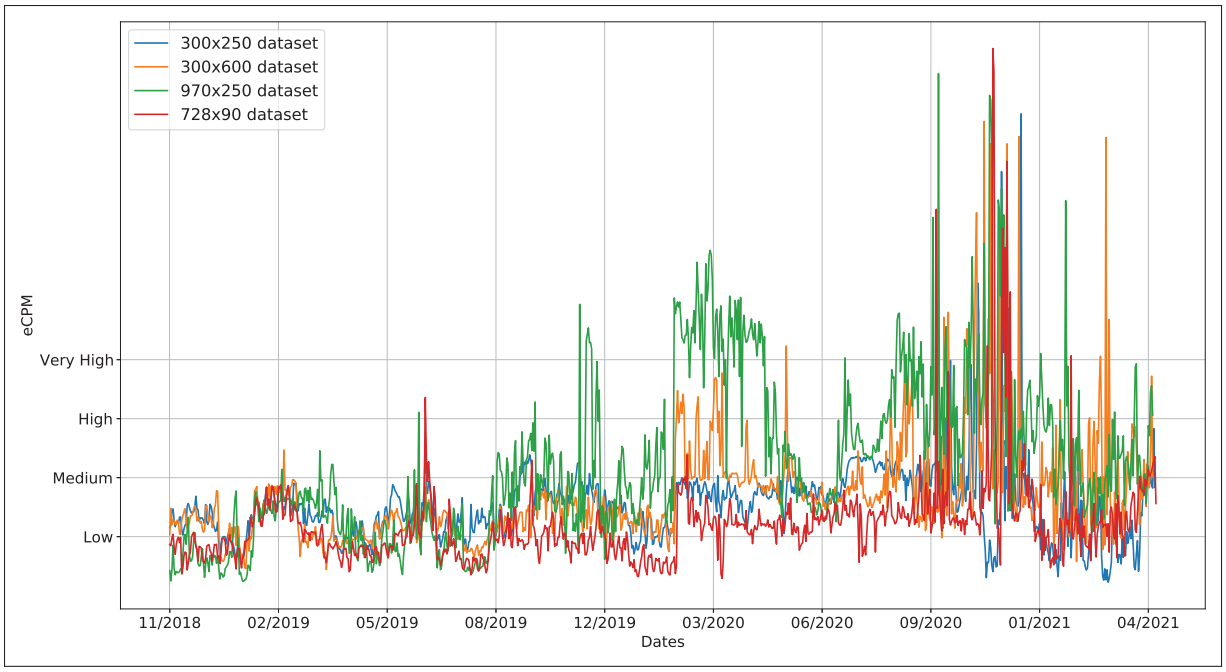

Figure 3.1 The four datasets with four different ad sizes

build a univariate model to be trained on this data. Because the univariate data are less prone to the curse-of-dimensionality problem (Köppen, 2000), this reduction can help in building a faster and more efficient predictive models.

Figure 3.1 shows the pattern of the four used datasets with different ad sizes from 01/11/2018 to 26/04/2021. This figure shows the raw data (the daily eCPM) for each dataset before any preprocessing. For respecting the confidentiality of our business partner, the y-axis (eCPM) represents the daily eCPM as low, medium, high, and very high ranges instead of the real eCPM values. As shown in Figure 3.1, the four datasets have some peaks and the data pattern has many variations especially in the period from 09/2020 to 04/2021. These variations will make our mission more exciting and challenging because training the models will be on the past eCPM values while the prediction will be on the future values.

3.1.1 Splitting the Datasets

As discussed earlier in section 3.1, the problem has been reduced to a univariate forecasting problem. Since our data constitutes a measurements for the eCPM over some time interval, this data is considered a time-series. Time series is defined as "a sequential set of data points, measured typically over successive times. It is mathematically defined as a set of vectors $x(t), t = 0, 1, 2, \ldots$ where t represents the time elapsed" (Adhikari & Agrawal, 2013). We divide the dataset into a set of N sequential values which representing the model input and the associated average eCPM that represents the model output as show in Figure 3.2. In Figure 3.2, the first row shows the first step in which we take the first seven elements as input and the subsequent seven elements are used to compute the target (t) (e.g., the average eCPM for the next seven days). Then move one slot, leave index 1, and take the seven elements that starting from index 2 as the input size and the subsequent seven elements used to calculate the target (t). After that move another one slot and start from index 3 to take the next seven input size values and so on till the end of the dataset. Each of these input/output pairs is called sample.

3.1.2 Model Input Size

In time-series, deciding the history that the model should see to predict the next value (named time steps or lags) is a crucial factor in the model performance. In our experiments, we used different values for the time step corresponding to one week, two weeks, and one month. Taking one week as a time step means that the input to the model is seven values representing the eCPM for the past seven days. For two weeks and one month, the model receives 14 and 30 values, respectively, as input. In all cases, the model output will be the average eCPM over a window around certain date. The width of the this window may be 7, 14, or 30 days.

3.2 Predicting Average eCPM

This section introduces the general framework for predicting the average eCPM for the next window-width(W) days given the past input-size(N) business days as input. Then we show three

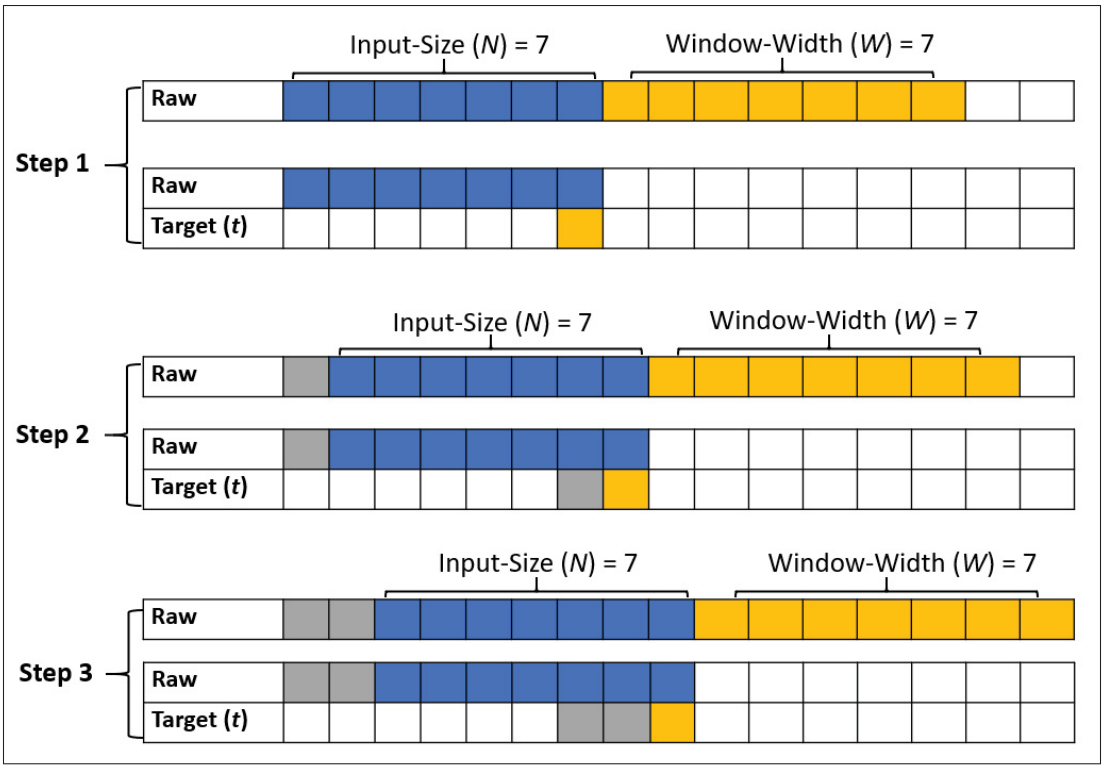

Figure 3.2 Three steps in splitting the dataset

concrete cases we used in this work, namely predicting the average eCPM for 7, 14, and 30 days given the past seven days. In terms of the coding, we can use any window-width (W) value for predicting the average eCPM. However, we used these values, (7, 14, and 30), because they are recommended by the Industrial Partner.

3.2.1 Framework for Average eCPM Prediction

Average eCPM prediction is the process of predicting the average of the next W days given the eCPM values for the past N days.

Toward this objective, we split our dataset to a set of sequences such that: $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_S, y_S)\}$ where S is the total number of obtained sequences. Each sequence consists of a pair of values, (x_i, y_i). In this case, x_i is an N-dimensional vector representing the average eCPM for a set of consecutive N days. While y_i represents the value of

the average eCPM for the next W days. The value of y_i represents the target to be predicted by certain model (i.e, label).

The model in our case should be capable of capturing the temporal dependencies between the input sequence and the associated label. Long short-term memory (LSTM) models may raise as a powerful candidate model for solving this problem since it can make prediction on the time series data that may have delay of unknown period between the important events in time series (Shu, Zhang, Sun & Tang, 2020). However, according to (Gers, Eck & Schmidhuber, 2002), LSTM is designed to capture very long temporal dependencies between input values. The problem under consideration may does not maintain this very far temporal dependencies. Rather, the value to be predicted is more related to the eCPM values of the most recent N days.

To verify the learning capabilities of different models in solving our problem, we build two neural network models. An LSTM architecture and an MLP architecture. We will compare these two models and select the suitable one that achieves the best model accuracy. The comparison between these two model done by running different experiments. After that, we will compare the selected model with a machine learning model, namely KNN model, to select the best model with the highest accuracy for each dataset.

The experiments were done using different model depths and different activation functions. For selecting the model depth and the activation function, the experiments done on one dataset (300×250 dataset) with target (t) = 7. The experiments were done on the 300×250 dataset because it is the most popular ad banner size used across the web (Match2One, 2021). Also, we used (t) = 7 to make the results more accurate since we expect that the model will achieve better accuracy with the short temporal dependencies than the long temporal dependencies. For selecting the model architecture, the experiments done on the four datasets (300×250, 300×600, 970×250, and 728×90) with (t) values equals 7, 14, and 30.

In the next chapter, we will explicitly discuss: the used datasets, different model depths, activation functions, and comparative experiments carried out to select the suitable model that achieves the best accuracy.

CHAPTER 4

MODELS COMPARISONS

In this chapter, we will explain the different types of models that we used. We will describe the activation functions used in the experiments. Also, we will explain all the experiments that we run with all the different hyper-parameters that we used.

We run the experiments on the four datasets (300×250, 300×600, 970×250, and 728×90). The four datasets contain 870 data points after splitting. We used 480, 120, and 270 data points for training, validation, and testing respectively. Since the standard splitting is 80%, 20%, this will make the test set 170 data points which we saw as small size. Thence, I added 100 points to the test set so it became 270 data points. In terms of model depth, we tried two different depths namely vanilla and stacked models.

4.1 Vanilla Model

The vanilla model is a simple model that contains one hidden layer. As shown in Figure 4.1, the model has two layers the input layer and the output layer.

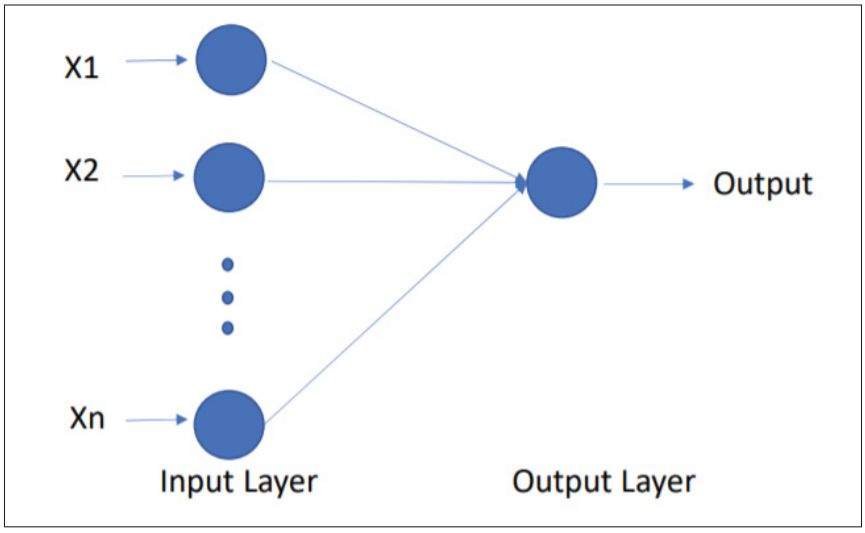

Figure 4.1 Vanilla Model

4.2 Stacked Model

The stacked model has at least two hidden layers. As shown in Figure 4.2, there is one input layer, several hidden layers (the number of hidden layers is much greater than 2), and one output layer.

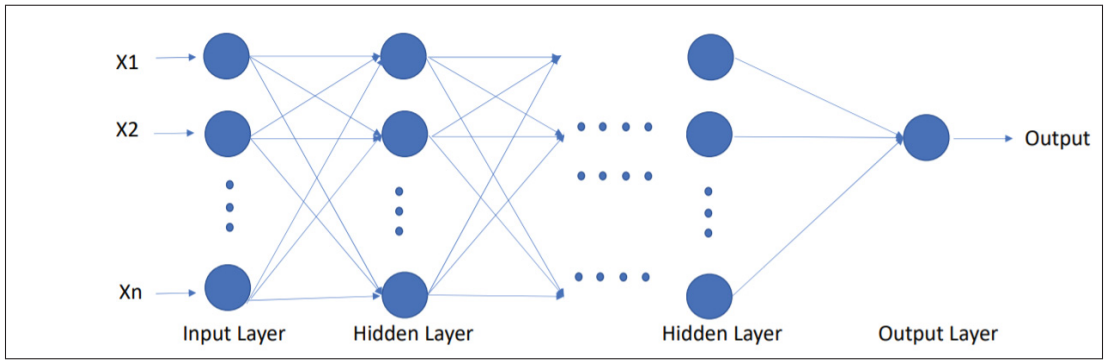

Figure 4.2 Stacked Model

On the other hand, to achieve the best eCPM prediction, we used different activation functions trying to choose the best function that helps the model achieve the best accuracy. We run the experiments using two activation functions (*Tanh* and *ReLU*).

4.3 Hyperbolic Tangent Activation Function (*Tanh*)

Hyperbolic Tangent Activation Function (Tanh) is the most used function with the time series problems.

Equation 4.1, shows the *Tanh* equation that used in machine learning (Sharma, 2021).

$$Tanh = \frac{2}{1 + e^{-2x}} - 1 \tag{4.1}$$

As shown in Figure 4.3, *Tanh* function takes real values as input and its outputs range from -1 to 1.

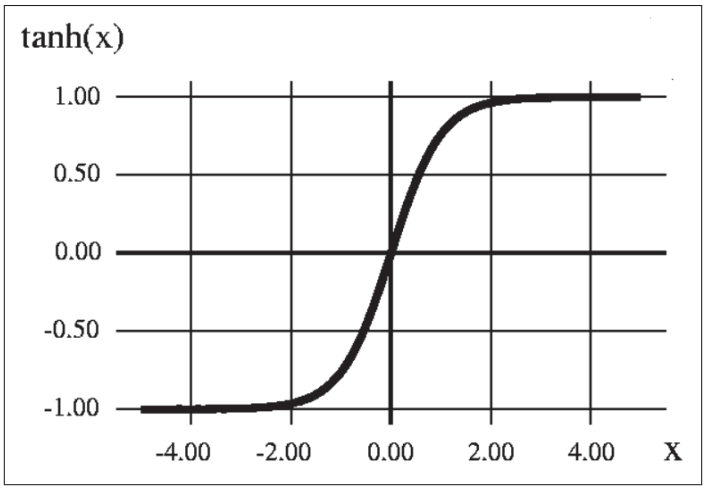

Figure 4.3 *Tanh* Activation Function
Taken from Schraudolph (2021)

4.4 Rectified Linear Activation Function *(ReLU)*

Rectified Linear Activation Function (ReLU) becomes the default activation function for many types of neural networks (Brownlee, 2020). As shown in Figure 4.4, the output will take the input value directly if it is positive, otherwise, it will output zero. So, it is become easier in model training and often achieves better performance.

We can see the *ReLU* activation function equation in (4.2).

$$ReLU(X) = Max(0, X) \tag{4.2}$$

4.5 Evaluation Metrics

In this section we will discuss the three evaluation metrics that we used in the experiments.

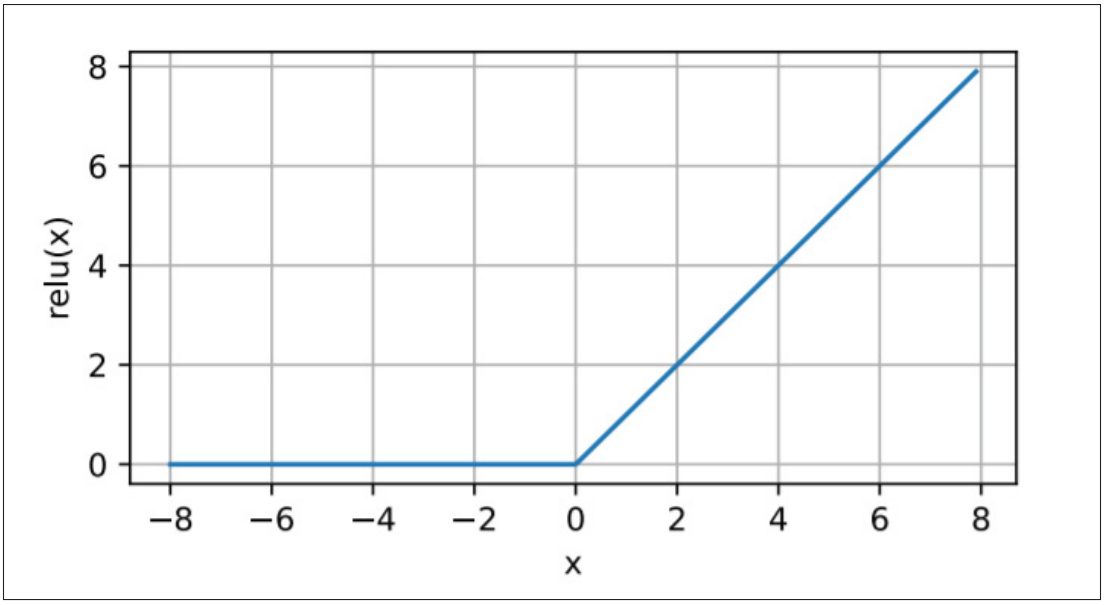

Figure 4.4 *ReLU* Activation Function
Taken from Zhang *et al.* (2021)

4.5.1 Mean Square Error (*MSE)*

The first evaluation metrics is *Mean Square Error (MSE)*. The *MSE* is the summation of the square distance between the actual value (y_i) and the predicted value ($\hat{y}_i$) as shown in (4.3).

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4.3}$$

We not only used the *MSE* function in model training as a loss function but also, we used it for models comparisons after the training.

This Loss function is used in all the experiments.

4.5.2 Normalized Mean Square Error (*NMSE)*

The *Normalized Mean Square Error (NMSE)* is the second evaluation metrics that we used to give us an overview of the models performance over the different dataset.

The *NMSE* is the summation of the square distance between the actual value (y_i) and the predicted value ($\hat{y}_i$), the *MSE* value, divided by the square of the actual value (y_i) as shown in (4.4).

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{(y_i)^2} \tag{4.4}$$

4.5.3 Percentage Error

The *percentage error* is the third evaluation metric that we used to define the accuracy of the different machine learning algorithms that we used on different datasets. It is a way to analyze how successful an experiment was.

As show in (4.5), the *percentage error* is the difference between the absolute error of the actual value (y_i) and the predicted value ($\hat{y}_i$) divided by the actual value (y_i). The absolute error value is a decimal value so, it multiplies by 100% to find the percentage error.

$$\mathcal{L}(y, \hat{y}) = \frac{|y_i - \hat{y}_i|}{y_i} \times 100 \tag{4.5}$$

4.6 Adam Optimization Algorithm

Adam algorithm is an optimization algorithm introduced by *Diederik Kingma* and *Jimmy Ba* in (Kingma & Ba, 2014). It is an extension to the well-established stochastic gradient descent algorithm which is widely used in machine learning applications. It's main task is to find the model parameters corresponding to the best possible fit between predicted and actual outputs (Stojiljković, 2021). The name of *Adam* optimization algorithm is derived from adaptive moment estimation since it uses the estimations of the first and second moments of the gradient to adapt the learning rate for each weight of the neural network.

We can summarize the main reasons for selecting Adam optimizer in our experiments as follows:

- Straightforward method to implement.
- Requires little memory space.
- Works well with large data and/or parameters.
- Robust and well-suited method to the wide range of non-convex optimization problems in the field machine learning (Kingma & Ba, 2014).
- Combines the advantages of two other extensions of stochastic gradient decent algorithm, *Gradient Descent Adaptive Gradient Algorithm (AdaGrad)* and *Root Mean Square Propagation (RMSProp)* (Brownlee, 2021a).

In the following sections, we will discuss the different experiments that were done to select the suitable activation function, model depth, and model architecture that achieve our goal.

4.7 Activation Function Selection

Experiment 1.1:

- **Objective:** Verify which activation function (*Tanh* or *ReLU*) helps the MLP model to achieve the best accuracy.
- **Description:** In this experiment we used vanilla-MLP model. We used *Adam* optimizer and *MSE* loss function. The model trained using input layer with 50 nodes and output layer with one node with number of epochs equal 300. In this experiment we used two different activation functions *Tanh* and *ReLU*. This experiment done on 300×250 dataset with target(t) value equal 7. The target is the average eCPM around certain date.

Experiment 1.2:

- **Objective:** Verify which activation function (*Tanh* or *ReLU*) helps the LSTM model to achieve the best accuracy.
- **Description:** we used vanilla-LSTM model with 50 nodes in the input layer, one node in the output layer, *Adam* optimizer, and *MSE* loss function with number of epochs equal 300. This

experiment done two times with two different activation functions (*Tanh* and *ReLU*). We run this experiment using 300×250 dataset with target(t) equal 7.

4.8 Model Type Selection

Experiment 2.1:

- **Objective:** Verify which MLP model (Vanilla or Stacked) will improve the model accuracy.
- **Description**: we run two models (Vanilla-MLP model and Stacked-MLP model) on the 300×250 dataset. We trained the Vanilla-MLP model with 50 nodes in the input layer and one node in the output layer.

  On the other hand, we trained the Stacked-MLP model with 150 nodes in the input layer, 50 nodes in the hidden layer, and one node in the output layer. In these two experiments we used *Adam* optimizer, *MSE* loss function, *ReLU* activation function, and target(t) equals 7. We trained the two models with numbers of epochs equal 300.

Experiment 2.2:

- **Objective:** Verify which LSTM model (Vanilla-LSTM model or Stacked-LSTM model) will improve the model accuracy.
- **Description:** We run two experiments with two different models. The first model is the vanilla-LSTM model with 50 nodes in the input layer and one node in the output layer.

  The second model is the Stacked-LSTM model with 150, 50, and one nodes in the input layer, the hidden layer, and the output layer, respectively. These two experiments were trained using *MSE* Loss function, *Adam* Optimizer, *Tanh* activation function, and target(t) equals 7. These experiments were done on 300×250 dataset with numbers of epochs equal 300.

4.9 Model Architecture Selection

Experiment 3.1:

- **Objective:** Verify the learning power of Vanilla-LSTM versus Vanilla-MLP in average eCPM prediction around certain date.

- **Description:** We used Vanilla-LSTM model and Vanilla-MLP model in Experiment 3.1. For Vanilla-LSTM model, we used two-layers architecture with 50 nodes in the input layer and one node in the output layer. Each layer has *Tanh* activation function. We used *MSE* loss function for training the model. We trained the model for 300 epochs with *Adam* optimizer. On the other hand, the Vanilla-MLP model consists of two-layers. We set 50 and one nodes for the input and output layers respectively. At each layer, *ReLU* activation function is employed. We trained the model for 300 epochs using *Adam* optimizer and *MSE* loss function. The experiments done on the four dataset (300×250, 300×600, 970×250, and 728×90) with target(t) equal 7, 14, and 30.

Experiment 3.2:

- **Objective:** Verify the learning power of machine learning models using the KNN algorithm versus the neural network models using the Vanilla-LSTM model on average eCPM prediction around certain date.
- **Description:** In this experiment we trained Vanilla-LSTM and KNN models trying to achieve the best model accuracy. For KNN model we used *MSE* as a loss function. We tried different values of K in range 200 values trying to choose the more accurate value of K that helps the model achieves the best accuracy. In this experiment, we used K with values equal 12, 25, and 12 for 300×250, 300×600, and 728×90 datasets respectively. Since the KNN model got the least error using these K values.

  On the other hand, we used Vanilla-LSTM model with 50 and one nodes for the input and output layers respectively. For training Vanilla-LSTM model we used *Tanh* activation function, *MSE* loss function and *Adam* optimizer. The model is trained using 300 epochs. This experiment done on 3 dataset (300×250, 300×600, and 728×90) with target (t) equals 7.

Experiment 3.3:

- **Objective:** Verify the learning power of machine learning models, using KNN algorithm, versus the neural network models, using Vanilla-MLP model, on average eCPM prediction around certain date.

- **Description:** For the machine learning model (KNN), we used K equals 12 for 970×250 dataset since the model achieved the least error using K equals 12. We used *MSE* as a loss function for training the KNN model.

  For training the neural network model (vanilla-MLP), we used two layers with 50 nodes in the input layer and one node in the output layer. We used *Tanh* activation function, *MSE* loss function and *Adam* optimizer. The vanilla-MLP model is trained using 300 epochs on 970×250 dataset with target (t) equal 7.

Finally, we summarize all the experiments in Table 4.1

Table 4.1 Conclusion of all Experiments

Experiment Type	**Experiment**	**Objective**
Activation Function Selection	1.1	Verify which activation function (*Tanh* or *ReLU*) helps the MLP model to achieve the best accuracy.
	1.2	Verify which activation function (*Tanh* or *ReLU*) helps the LSTM model to achieve the best accuracy.
Model Type Selection	2.1	Verify which MLP model (Vanilla or Stacked) will improve the model accuracy.
	2.2	Verify which LSTM model (Vanilla or Stacked) will improve the model accuracy.
Architecture Selection	3.1	Verify the learning power of Vanilla-LSTM versus Vanilla-MLP in eCPM prediction.
	3.2	Verify the learning power of machine learning models using KNN algorithm versus the neural network models using Vanilla-LSTM model on average eCPM prediction around certain date.
	3.3	Verify the learning power of machine learning models, using KNN algorithm, versus the neural network models, using Vanilla-MLP model, on average eCPM prediction around certain date.

In the next chapter, we will display and analyze the results of all experiments. We will explain the suitable model for each used dataset.

CHAPTER 5

RESULTS

In the previous chapters, we explained the background of this work and the main problem that we try to solve. We discussed the literature work that related to our work and the methodology that we used to achieve our goal. Also, we described the experiments that we did to achieve the best results.

In this chapter, we discuss the results for the different experiments described in chapter 4. In all experiments figures, the x-axis represents the Date and the y-axis represents the average eCPM values range. The orange and blue curves represent the prediction curves and the black curve represents the target curve.

5.1 Results of Activation Function Selection

There are many activation functions that can be used to build a machine learning model. In our experiments, we used two different activation functions that are suitable for our problem as explained previously in section 4.3 and section 4.4.

5.1.1 Results of Experiment 1.1

In this subsection, we show the result of **Experiment 1.1** that we explained explicitly in section 4.7 trying to choose the best activation function that helps the MLP model achieves the most accurate results.

For the figure in experiment 1.1, the black curve represents the target curve, the blue curve represents the Vanilla-MLP prediction curve using *Tanh* activation function, and the red curve represents the Vanilla-MLP prediction curve using *ReLU* activation function.

As shown in Figure 5.1, there is no significant difference between the two prediction curves except in the period between 12/2020 and 01/2021. The vanilla_MLP curve using *ReLU*, the red

curve, can predict the average eCPM better than the vanilla_MLP using *Tanh*, the blue curve. The model with the *Tanh* activation function can not predict the average eCPM well in the period between 12/2020 and 01/2021. As shown in the figure, in this period some of the output values are in the high range while vanilla_MLP model using *Tanh* predicts it as low values in the low range. While vanilla_MLP curve using *ReLU* can predict these high values sometimes as medium values or nearby the high values.

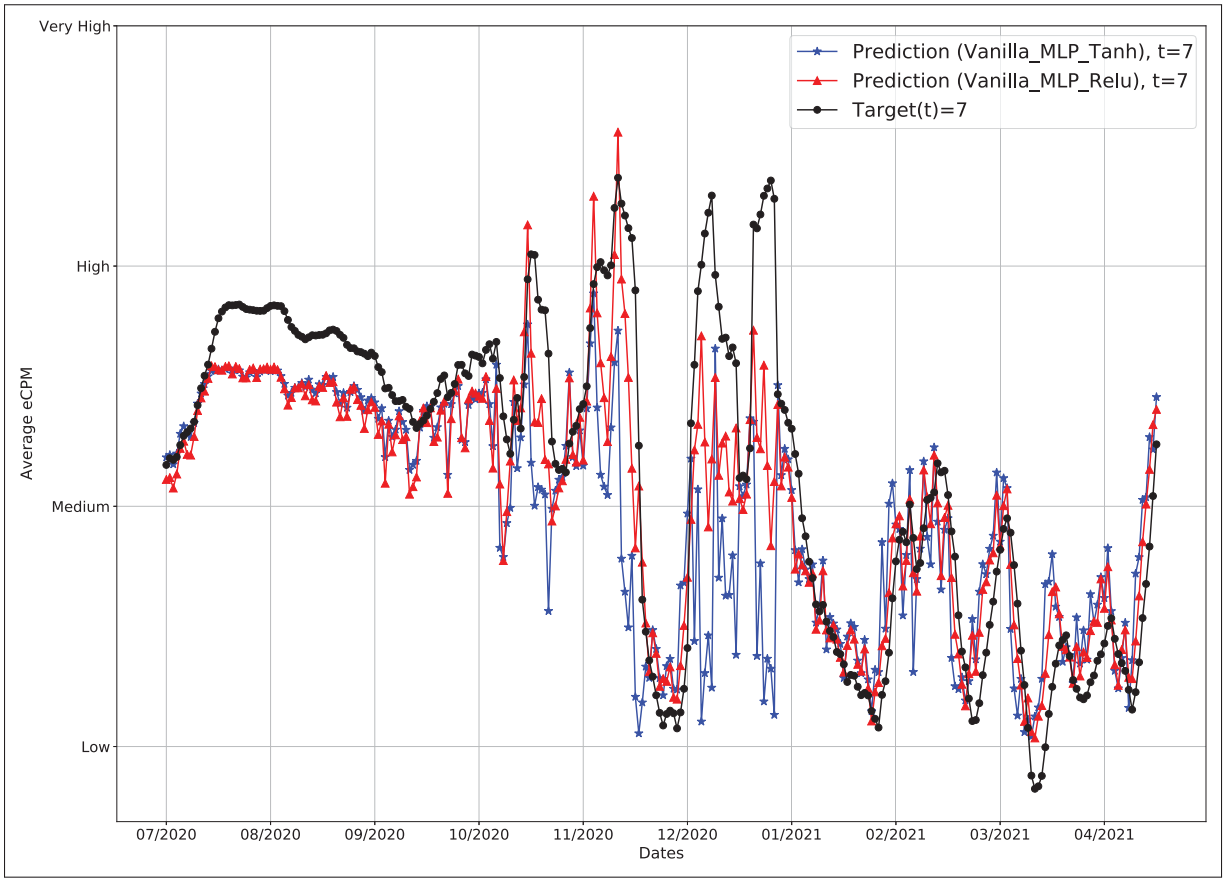

Figure 5.1 Activation Function Selection using MLP Model with $t = 7$ on 300×250 Dataset

So, we can say that MLP model can achieve the best results using the *ReLU* activation function with 300×250 dataset. Accordingly, we assumed that *ReLU* activation function will help the MLP model achieves the best accuracy with the three other datasets.

5.1.2 Results of Experiment 1.2

As explained in section 4.7, the main objective of experiment 1.2 is verifying which activation function is the most suitable one to be used with the LSTM model.

Figure 5.2 shows the two prediction curves using LSTM model with *ReLU* and *Tanh* activation functions. As shown in Figure 5.2, we can see that the prediction curve using *Tanh*, the blue curve, can predict the output (the average eCPM) better than the prediction curve using *ReLU*, the red curve. Since the red curve has some peaks especially in the period between 11/2020 and 01/2021.

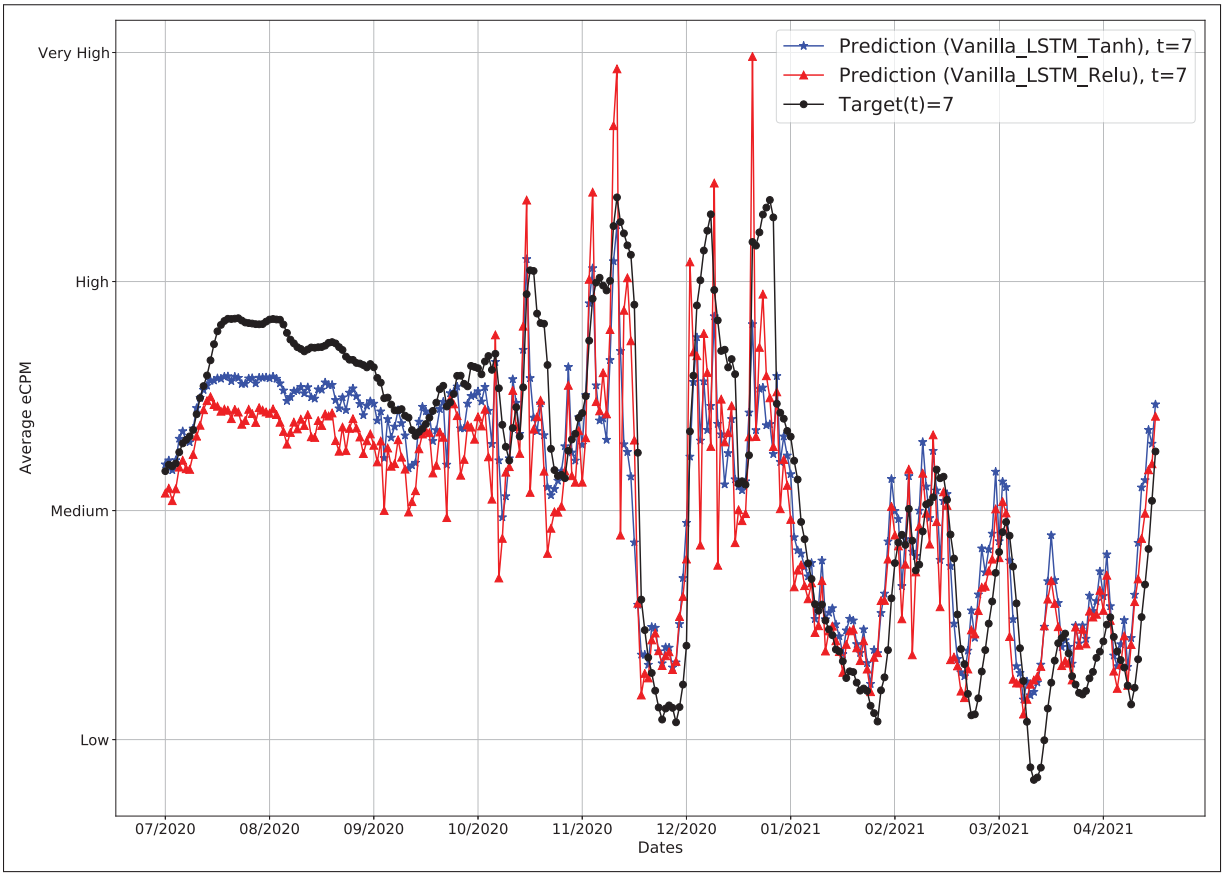

Figure 5.2 Activation Function Selection using LSTM Model with $t = 7$ on 300×250 Dataset

From this result we can conclude that, the LSTM model can achieves more accurate results using *Tanh* activation function than using *ReLU* activation function with 300×250 dataset. Accordingly, we assumed that *Tanh* activation function will help the LSTM model achieves the best accuracy with the three other datasets.

5.2 Results of Model Type Selection

To get the best model accuracy with the least error we should run different experiments using different types of models. The goal of these experiments is to choose the suitable type of model that achieves the best accuracy. As explained previously in section 4.8, we run experiments with two different types of models, vanilla and stacked models, then we choose the type of model that reaches the highest accuracy.

We start the experiments with MLP model followed by the experiment with LSTM model.

5.2.1 Results of Experiment 2.1

In this subsection we discuss the results of running two experiments using two different types of models, vanilla and stacked, using MLP model as discussed in section 4.8.

As shown in Figure 5.3, the vanilla_MLP prediction curve, the red curve, is very close to the values of the target curve, the black curve. The stacked_MLP prediction curve, the blue curve, has some peaks especially in the period between 11/2020 to 01/2021. The stacked_MLP model predicts the average eCPM values that exist in the high range as very high values.

Based on the previous comparison and discussion between the two model types, we can choose the vanilla-MLP model for average eCPM prediction since it can predict the average eCPM better than the stacked-MLP model.

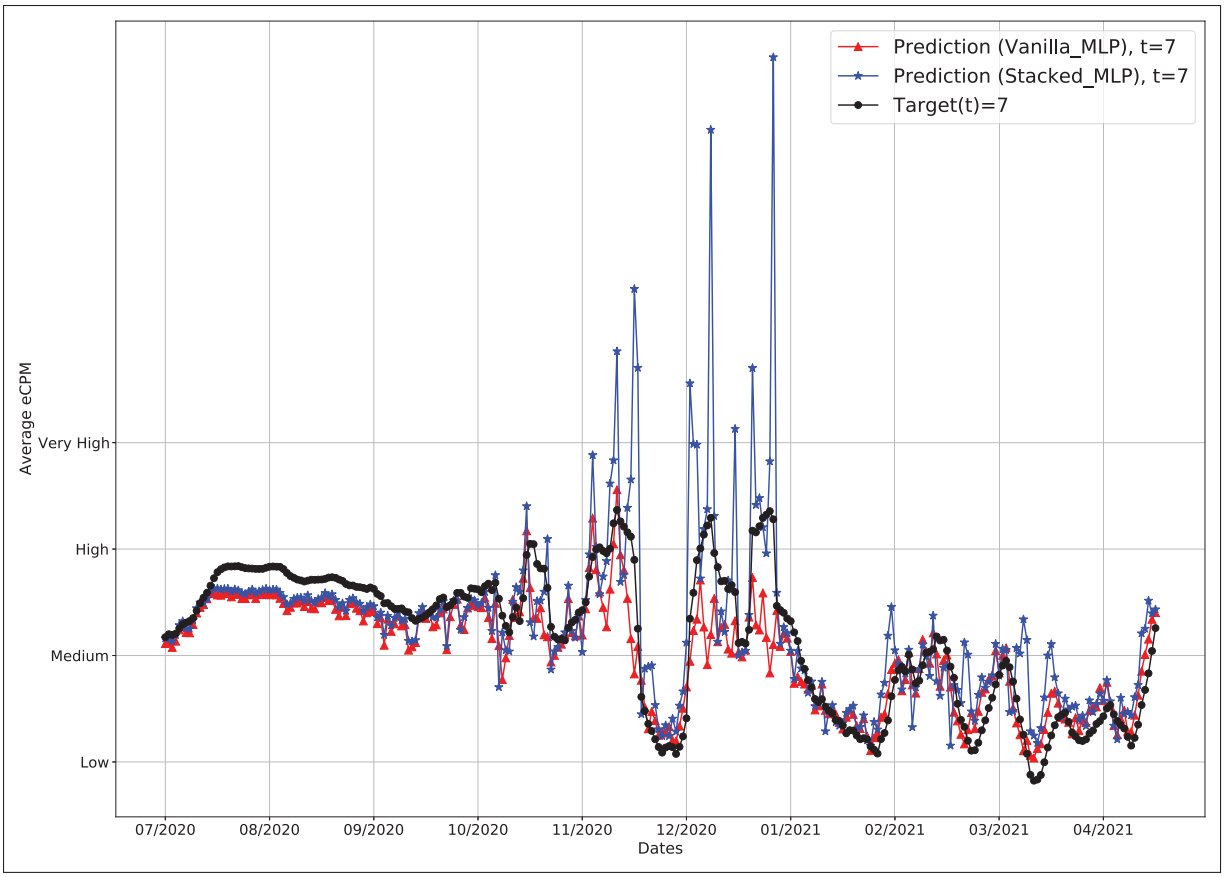

Figure 5.3 Model Type Selection using MLP Model with $t = 7$ on 300×250 Dataset

5.2.2 Results of Experiment 2.2

The main objective of **Experiment 2.2** as explained in section 4.8 is verifying which model type, vanilla or stacked, is the suitable one that achieves the best model accuracy using LSTM model.

As shown in Figure 5.4, the two prediction curves are approximately the same. The Vanilla-LSTM prediction curve, the blue curve, and the stacked-LSTM prediction curve, the red curve, are very close to the target curve. This means that we can achieve a very good prediction comparable with the average eCPM values, the target. Also, the *MSE* values for the Vanilla-LSTM model and stacked-LSTM model are approximately the same. The *MSE* values are 2.5726 and 2.6759 for vanilla-LSTM model and stacked-LSTM model respectively. Wherefore, We selected the

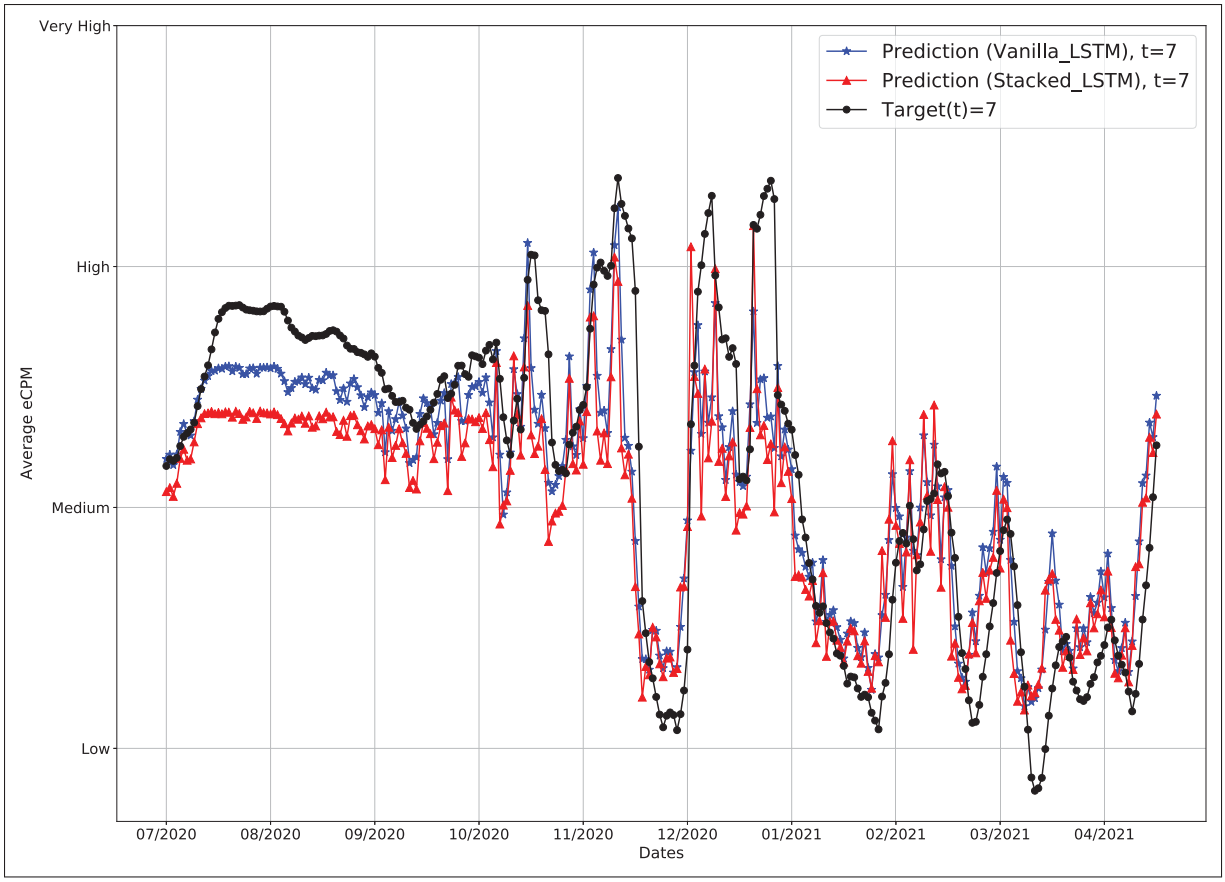

Figure 5.4 Model Type Selection using LSTM Model with $t = 7$ on 300×250 Dataset

Vanilla-LSTM model to run the remaining experiments to make the comparison more consistent and fair between the LSTM and MLP models in the next experiments.

From all previous experiments, We chose vanilla-LSTM model using *Tanh* activation function and vanilla-MLP model using *ReLU* activation function to run the next experiments that explained in section 4.9.

5.3 Results of Architecture Selection

In this work, our main goal is to select the best model that can predict the average eCPM around certain date with the highest accuracy. In the previous experiments, we chose the suitable activation function and model type that provided the best accuracy.

In the following subsections we will show the results of vanilla-LSTM model using *Tanh* activation function and vanilla-MLP model using *ReLU* activation function on average eCPM prediction around certain date using the four datasets (300×250, 300×600, 970×250, and 728×90). Also, Table 5.1 conclude the vanilla-LSTM and the vanilla-MLP models error values using *MSE*, *NMSE*, and *percentage error* with t values equal 7, 14, and 30.

After that, we will compare the best model that will achieve the best accuracy from the comparison between the two neural network models with a machine learning model using the KNN algorithm.

5.3.1 Results of Experiment 3.1

In Experiment 3.1 we will show the results of the comparison between the two neural network models Vanilla-LSTM and Vanilla-MLP models on 300×250, 300×600, 970×250, and 728×90 datasets with different targets values as explained in **Experiment 3.1** in section 4.9.

In all results in Experiment 3.1, the vanilla-LSTM prediction curve represents the blue curve with the star marker. The vanilla-MLP prediction curve represents the red curve with the triangle-up marker. The target represents in the black curve with the circle marker.

5.3.1.1 Results of Experiment 3.1 on 300×250 dataset

The first comparison between vanilla-LSTM and vanilla-MLP models on 300×250 with target(t) equal 7 is shown in Figure 5.5. From this figure we can see that, there is a little difference between the vanilla-LSTM curve and the vanilla-MLP curve. Whereas, vanilla-LSTM model achieves less error than vanilla-MLP model. The *MSE* of vanilla-LSTM model is 2.5726 while the *MSE* of vanilla-MLP model is 3.0073.

The second comparison between the two models is shown in Figure 5.6. This figure shows the vanilla-LSTM and vanilla-MLP prediction curves with the target curve with t equal 14.

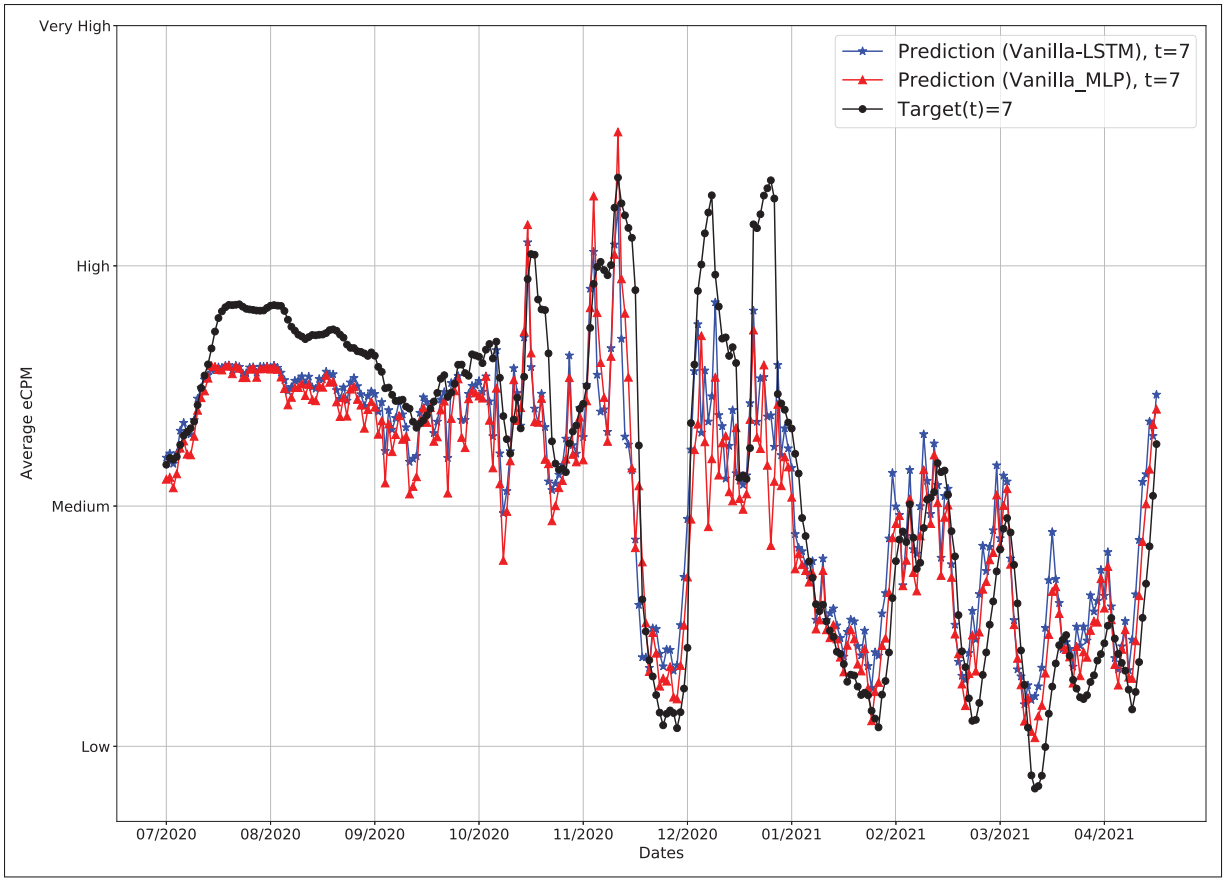

Figure 5.5 LSTM vs. MLP with $t = 7$ on 300×250 Dataset

The vanilla-LSTM curve, the blue curve, and the vanilla-MLP curve, the red curve, approximately have the same behavior in most cases except in the period between 11/2020 and 01/2021. We can see that the vanilla-MLP curve can not predict well in some points comparable with vanilla-LSTM around this period. Since vanilla-MLP curve predicts the output values that nearby high average eCPM values as low values and sometimes less than the low values. Also, the *MSE* of vanilla-MLP equals 2.3542, while the *MSE* of vanilla-LSTM equals 1.9891.

While in Figure 5.7 we can see that the two prediction curves, vanilla-LSTM and vanilla-MLP, with target(t) equal 30 can not predict the average eCPM values well comparable with the other curves in Figure 5.5 and Figure 5.6 with target(t) equal 7 and 14 respectively. Which means that the prediction being less accurate when increasing the (t) value. Although, We can see the same previous observations on Figure 5.7, since vanilla-LSTM curve can predict better than the

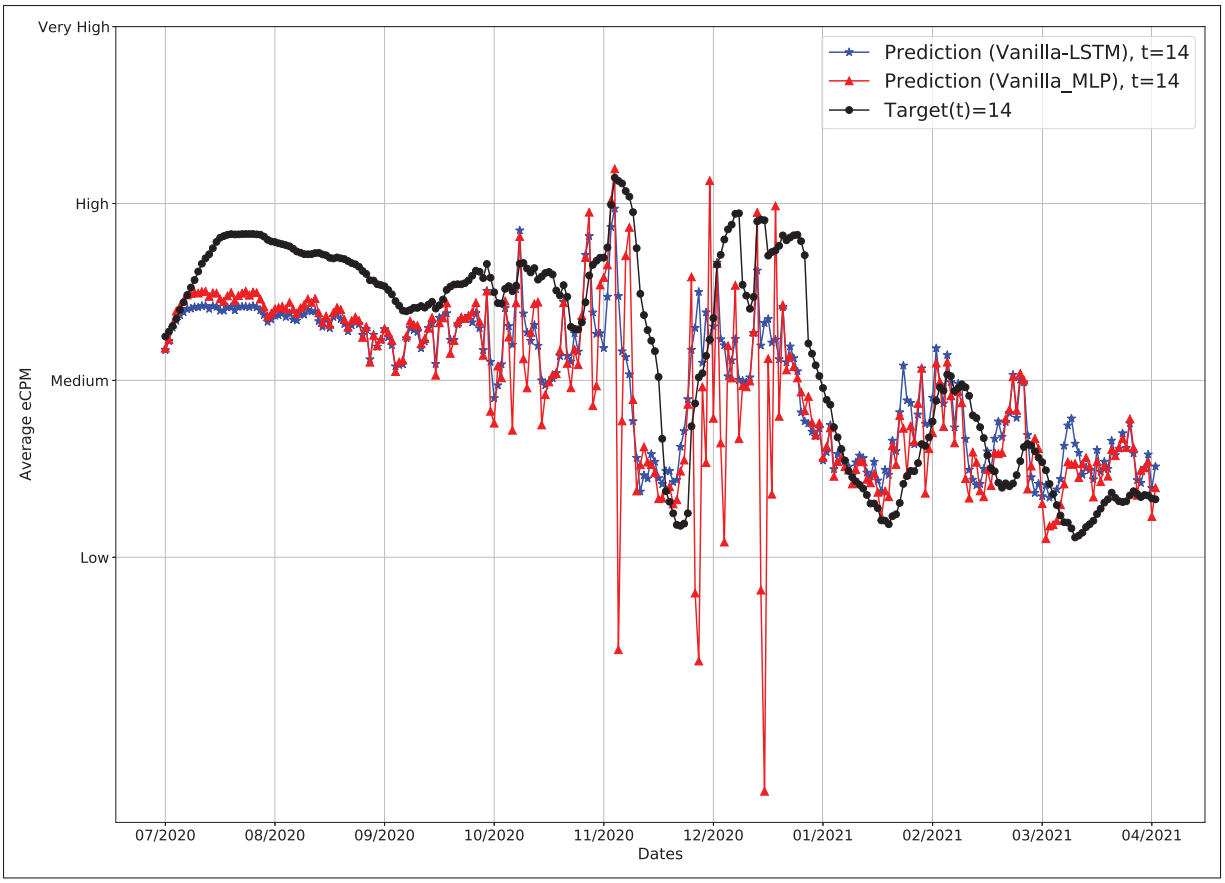

Figure 5.6 LSTM vs. MLP with $t = 14$ on 300×250 Dataset

vanilla-MLP curve. In Figure 5.7 we can see that the vanilla-MLP prediction curve has some low and less than low values in the period between 11/2020 and 12/2020. It, sometimes, predicts the average eCPM values between medium and high range as a high values in the high range.

Consequently, we can say that vanilla-LSTM model is better than vanilla-MLP model in predicting the average eCPM around certain date with target(t) equal 7, 14, and 30 on 300×250 dataset.

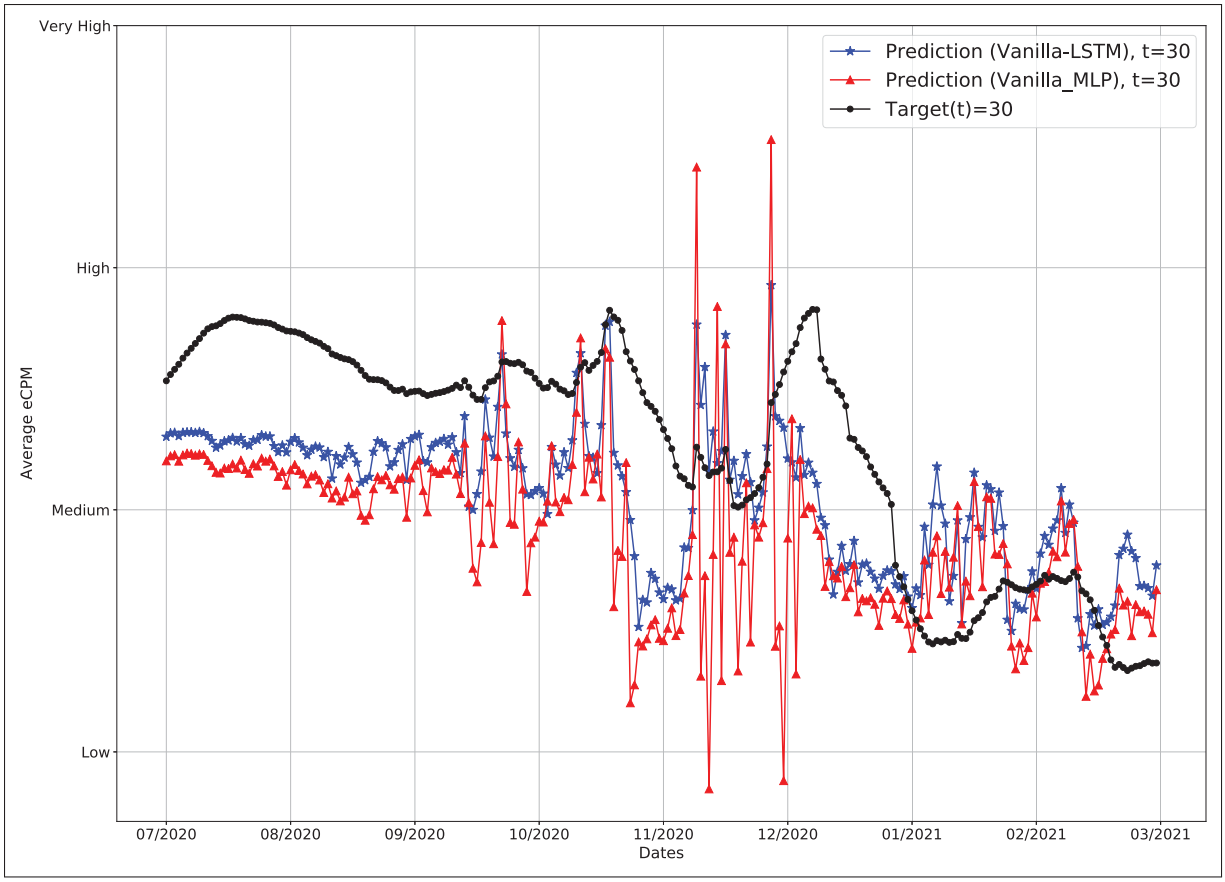

Figure 5.7 LSTM vs. MLP with $t = 30$ on 300×250 Dataset

5.3.1.2 Results of Experiment 3.1 on 300×600 dataset

In this experiment we display three figures that show the average eCPM prediction using vanilla-LSTM and vanilla-MLP models with target(t) values equal 7, 14, and 30 on 300×600 dataset.

Figure 5.8 shows two prediction curves, the blue and red curves, using vanilla-LSTM and vanilla-MLP models and one target curve, the black curve, that shows the average eCPM ranges around certain date with t equal 7. In Figure 5.8 we can see that the vanilla-LSTM prediction curve can predict the average eCPM well but there is a little difference between the vanilla-LSTM prediction curve and the target curve around the 12/2020 period. Whereas the vanilla-MLP prediction curve has several peaks especially in the periods between 10/2020 and 01/2021 and

between 03/2021 and 04/2021. It predicts some of the average eCPM values that exist in the high range as very very high values. Also, the *MSE* of vanilla-LSTM is 3.1656 while the *MSE* of vanilla-MLP is 7.0541.

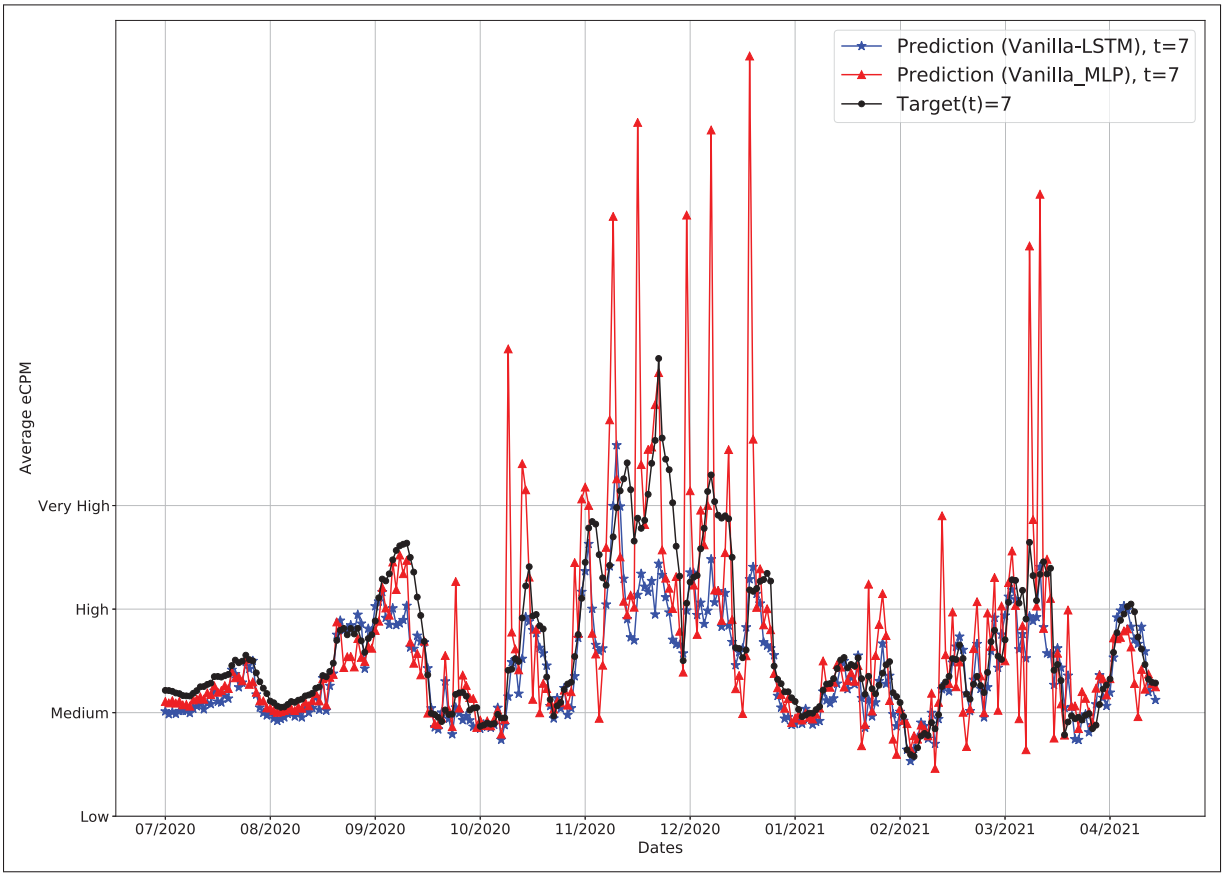

Figure 5.8 LSTM vs. MLP with $t = 7$ on 300×600 Dataset

So, we can say that the Vanilla-LSTM model can predict the average eCPM better than the vanilla-MLP model with less error at 300×600 dataset with t equals 7.

Similarly in Figure 5.9 we can see that Vanilla-LSTM model can predicts better than vanilla-MLP model in 300×600 dataset with (t) equals 14. Since the vanilla-MLP prediction curve has several peaks in many data points. For example, it predicts the values in the medium and high range as very high values. Also, it predicts the value in the high range as a very low value after 03/2021 period. While vanilla-LSTM prediction curve can approximately reach most of the average eCPM values.

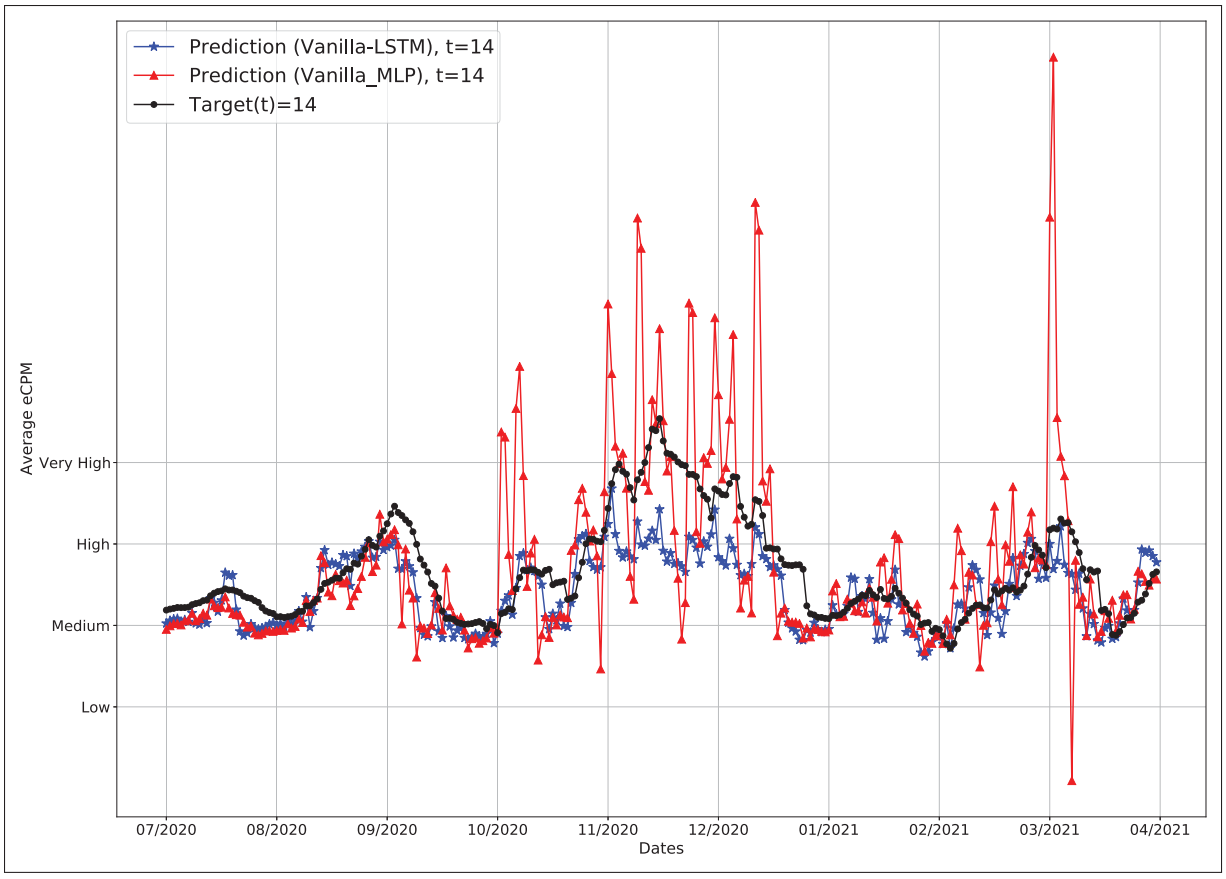

Figure 5.9 LSTM vs. MLP with $t = 14$ on 300×600 Dataset

The same problem exists in Figure 5.10. The vanilla-MLP curve has many peaks. While the vanilla-LSTM prediction curve can predict the target better than the vanilla-MLP prediction curve. Also, vanilla-LSTM achieves less error than vanilla-MLP as shown in Table 5.1.

From the previous observations in Experiment 3.1 on the 300×600 dataset, we can say that we can achieve the best model accuracy using the vanilla-LSTM model. Also, the prediction is being less accurate when increasing the t value. Since the predicting is more accurate with $t = 7$ than $t = 30$.

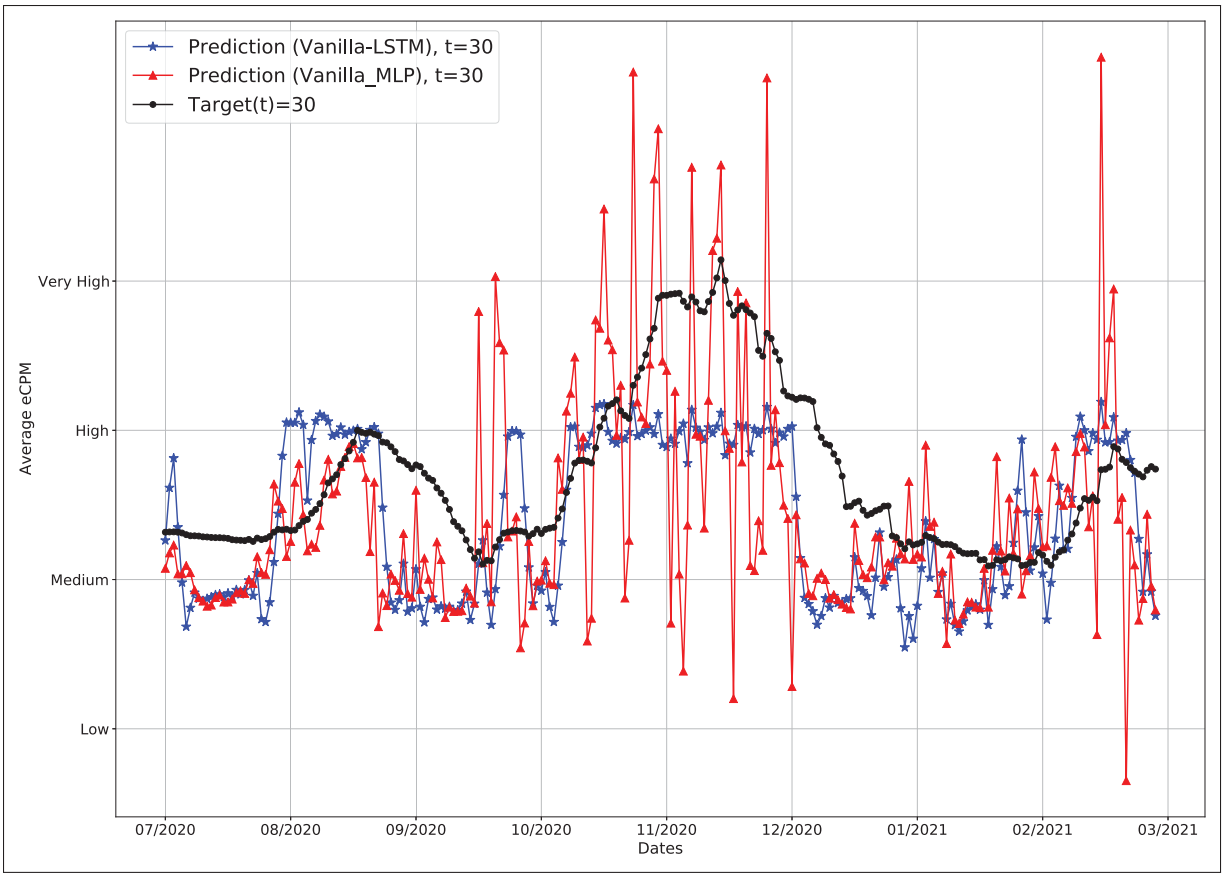

Figure 5.10 LSTM vs. MLP with $t = 30$ on 300×600 Dataset

5.3.1.3 Results of Experiment 3.1 on 970×250 dataset

In this sub-subsection we will discuss the vanilla-LSTM and vanilla-MLP models behavior on 970×250 dataset with (t) values equal 7, 14, and 30. In the next experiments we will observe differences in the behavior of the two models at the 970×250 dataset comparable with the models behavior at 300×250 and 300×600 datasets.

In Figure 5.11 with (t) equals 7 on 970×250 dataset we can see that, the vanilla-MLP model can predict the average eCPM better than the vanilla-LSTM model. The vanilla-LSTM prediction curve has strange behavior especially from 07/2020 till 12/2020 period. Also, at the period before and after 01/2021. It predicts the output values that exist in the high and very high ranges as low values. It achieves a very high error since it's *MSE* value = 8.3856. On the other hand,

the vanilla-MLP prediction curve can predict the average eCPM values very well in most cases. Also, the *MSE* of vanilla-MLP = 2.1123 which is very low comparable with the vanilla-LSTM error value.

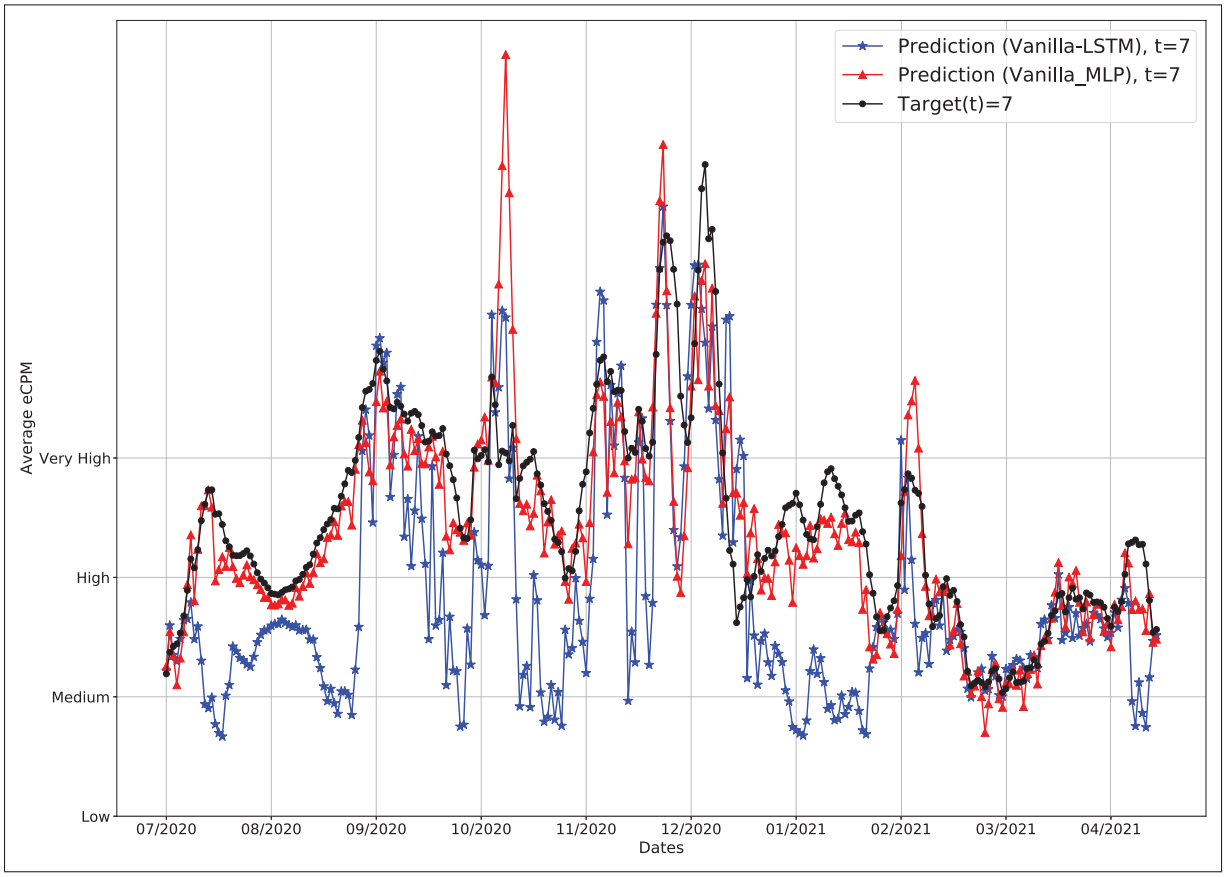

Figure 5.11 LSTM vs. MLP with $t = 7$ on 970×250 Dataset

Also, we can see the same behavior in Figure 5.12 with (t) equals 14. The vanilla-MLP prediction curve, the red curve, can predict the average eCPM better than the vanilla-LSTM prediction curve, the blue curve. In most cases, the red curve approximately has the same target curve pattern. The red curve can predict the average eCPM values very well. whereas the blue curve is overfitting at many periods. Also, the *MSE* that achieved by the vanilla-LSTM model is higher than the *MSE* that achieved by vanilla-MLP model.

The same behavior in Figure 5.13 with (t) equals 30. The vanilla-MLP model is more accurate with 970×250 dataset than vanilla-LSTM model. Most of the vanilla-LSTM prediction values

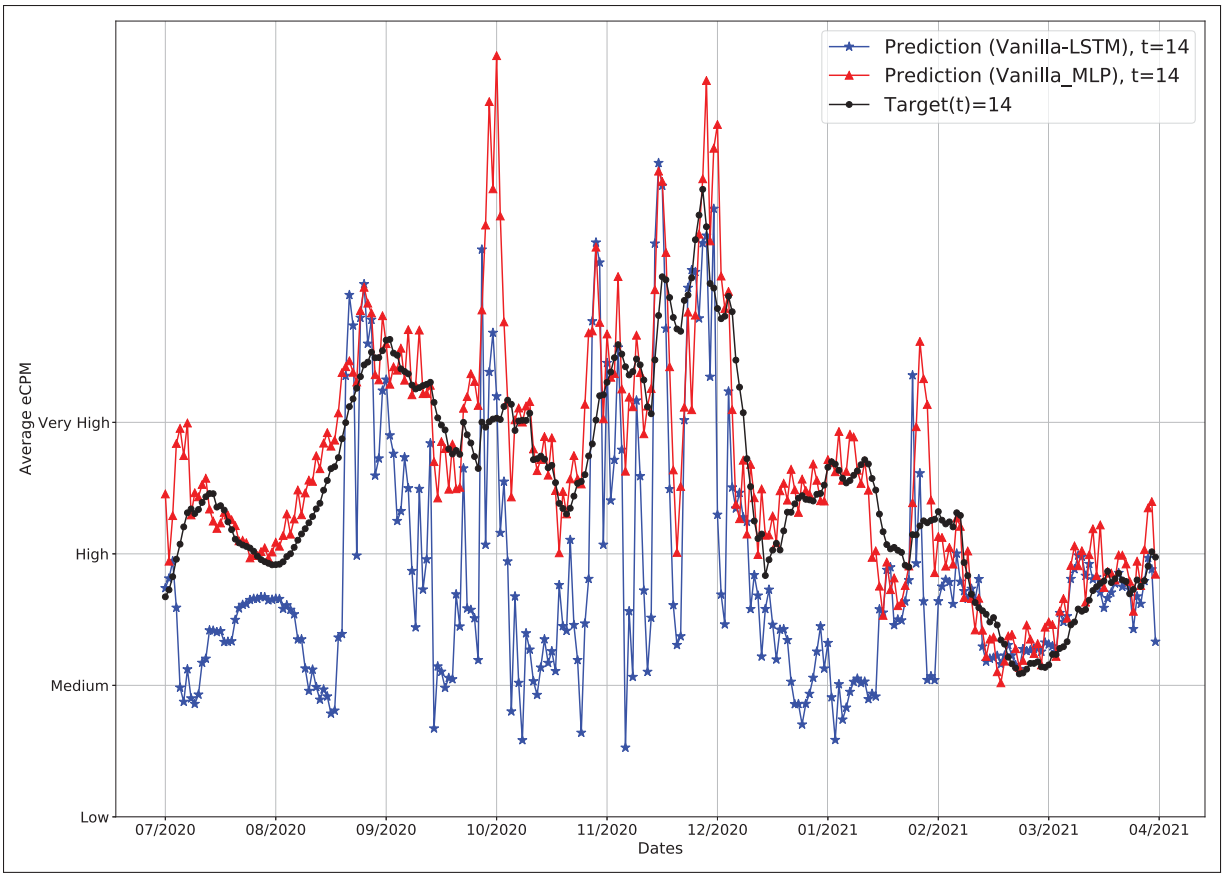

Figure 5.12 LSTM vs. MLP with $t = 14$ on 970×250 Dataset

are in the medium range while the most target values exist between high and very high range. From this figure and the *MSE* values that exist in Table 5.1 we can conclude that, vanilla-MLP model can predict the target better than vanilla-LSTM model.

In accordance with the previous experiments on the 970×250 dataset, we can conclude that the vanilla-MLP model can achieve good model accuracy comparable with the vanilla-LSTM model. The behavior of the 970×250 dataset is easier than the behavior of the previous datasets. Since the MLP model can predict well with the simple datasets. While LSTM model can achieve good model results comparable with the MLP model with the complex datasets since LSTM model is more complicated than the MLP model (Weytjens, Lohmann & Kleinsteuber, 2019). Also, the two models can predict better with the short forecasting periods than the long forecasting

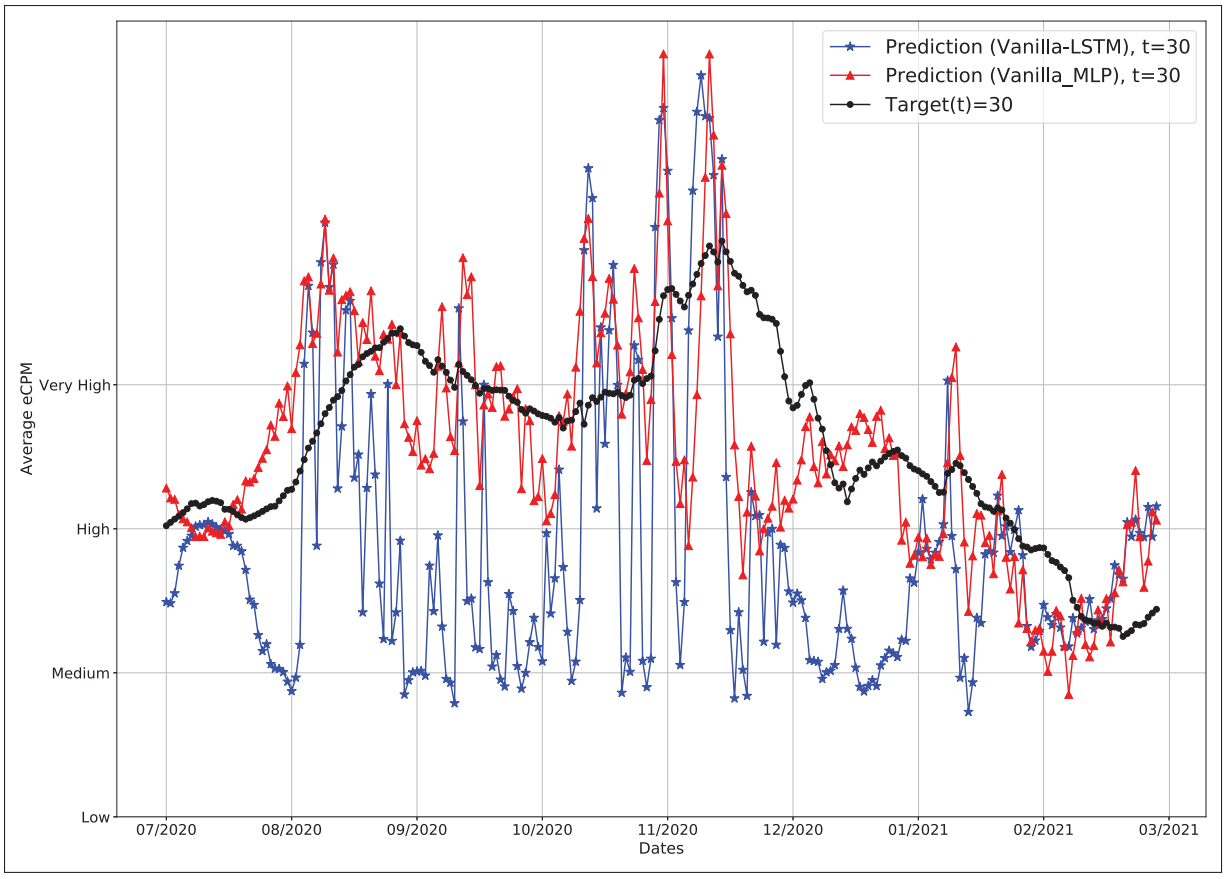

Figure 5.13 LSTM vs. MLP with $t = 30$ on 970×250 Dataset

periods. Since the two models forecast the average eCPM for the next 7-days and 14-days better than forecasting the next 30-days.

5.3.1.4 Results of Experiment 3.1 on 728×90 dataset

In this experiment we will compare between the vanilla-LSTM and vanilla-MLP models accuracy with t values equal 7, 14, and 30 on 728×90 dataset.

Figure 5.14 shows the first experiment on 728×90 dataset with t equals 7. In Figure 5.14 we can see that, the vanilla-LSTM model can predict the average eCPM well. Since the vanilla-LSTM prediction curve, the blue curve, is approximately the same as the target curve, the black curve, in most periods except around 12/2020 period. It can not predict the peaks that exist around this

period. On the other hand, the vanilla-MLP, the red curve, can predict the average eCPM well in most prediction periods comparable with the target curve. Although, it can not predict well at some prediction periods like after 10/2020 and before and after 12/2020. In the period around 12/2020, it predicts the value that exists in the very high range as a very low value (negative value). Also, the *MSE* of vanilla-LSTM model = 2.4737 while the *MSE* of vanilla-MLP model = 3.3251.

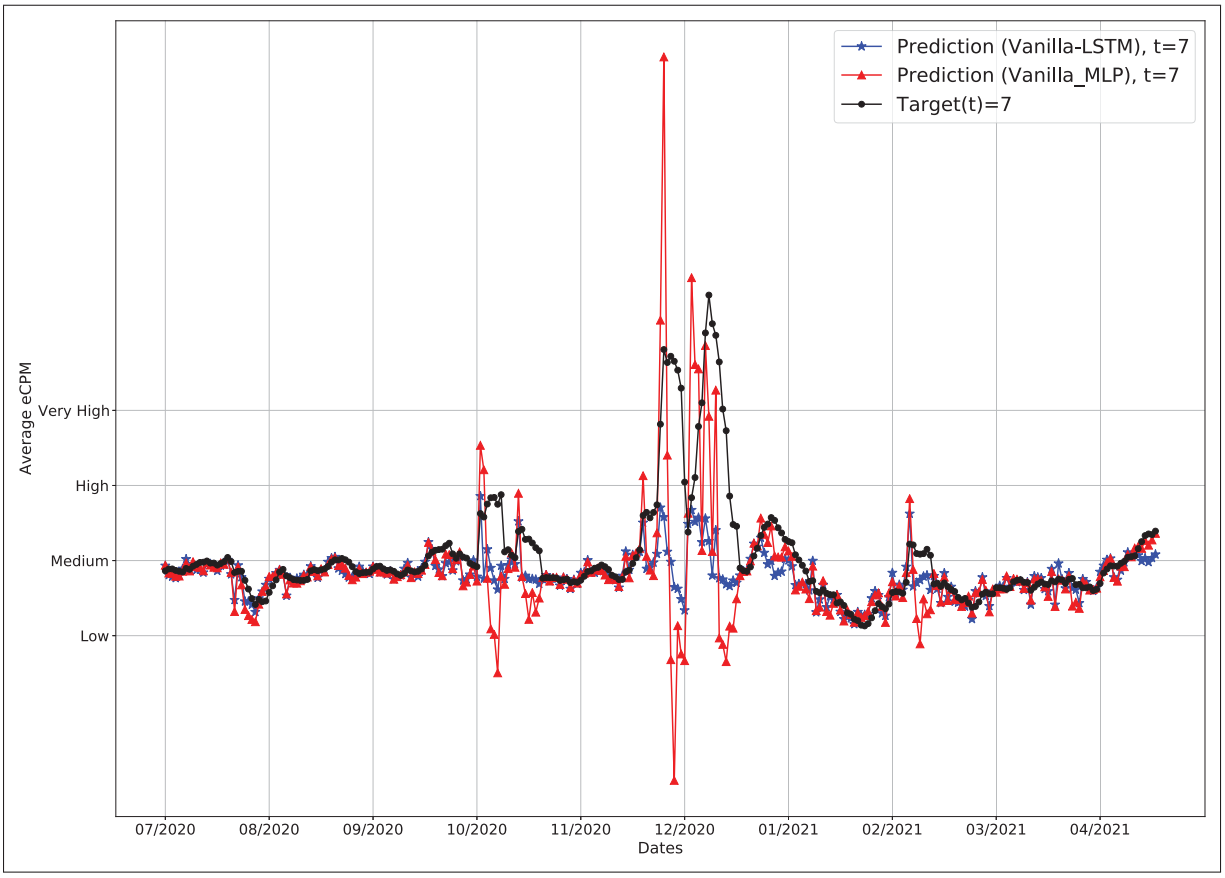

Figure 5.14 LSTM vs. MLP with $t = 7$ on 728×90 Dataset

Based on the prediction curves and the *MSE* values we can say that, vanilla-LSTM model can achieve better model accuracy than vanilla-MLP model on 728×90 dataset with t equals 7.

Figure 5.15 shows the second experiment on 728×90 dataset with t equals 14. In Figure 5.15 we can see that vanilla-LSTM prediction curve is better than vanilla-MLP prediction curve. Since the vanilla-LSTM curve can predict most of the real data points explained in the target

curve except the period before and after 12/2020. While vanilla-MLP curve can not predict well before 10/2020, before and after 12/2020, and at 02/2021. In the period before 12/2020, the vanilla-MLP curve predicts a negative output value while the average eCPM values must be positive. Also, the vanilla-MLP model achieved higher error than the vanilla-LSTM model.

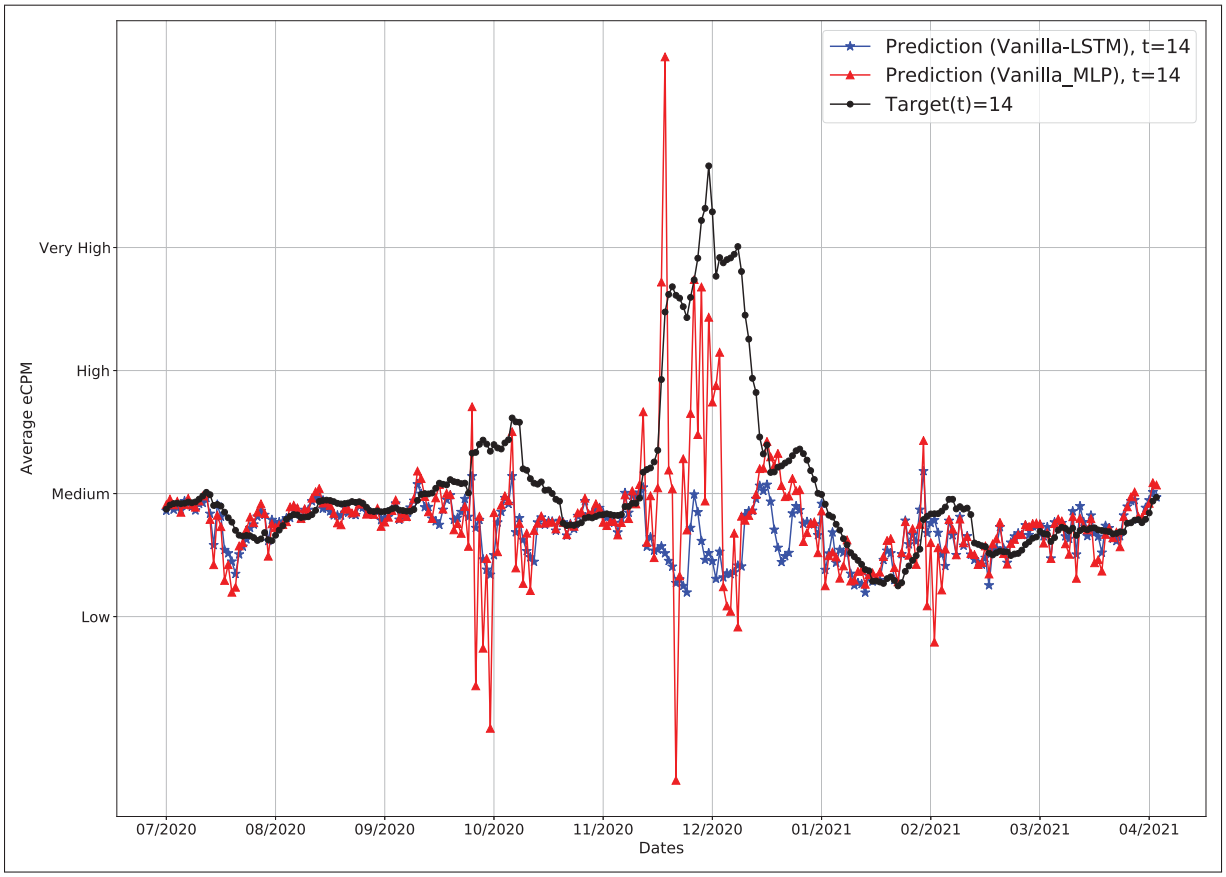

Figure 5.15 LSTM vs. MLP with $t = 14$ on 728×90 Dataset

So, we can conclude that, the vanilla-LSTM model can predict the average eCPM better than the vanilla-MLP model on 728×90 dataset with t equals 14.

The third experiment on 728×90 dataset shown in Figure 5.16 with t equals 30. In Figure 5.16 we can see the same observations that extracted from Figure 5.14 and Figure 5.15. Since vanilla-LSTM model can predict the average eCPM, that explained in the target curve, better than the vanilla-MLP model. Also, vanilla-MLP predicts the average eCPM values that exist in medium and high range as a very low output values (negative predicted values).

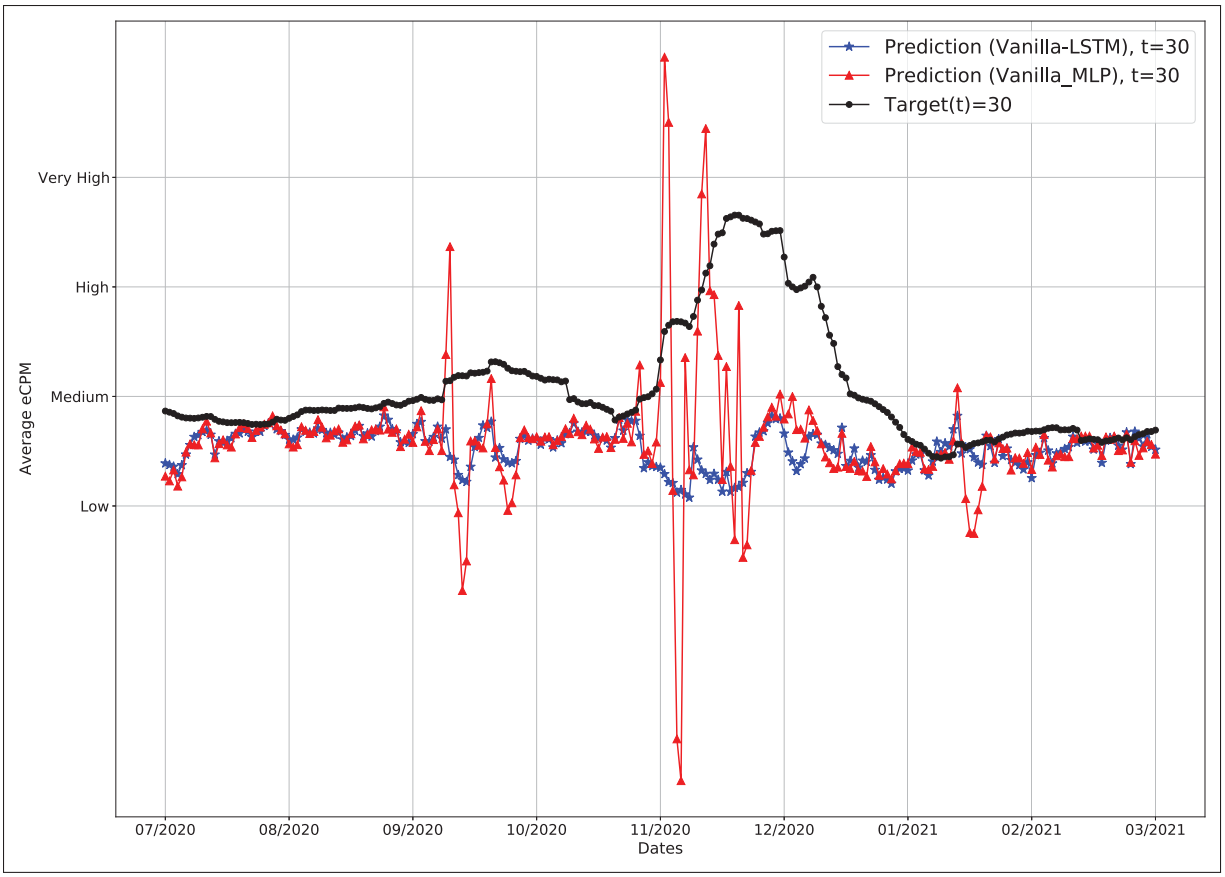

Figure 5.16 LSTM vs. MLP with $t = 30$ on 728×90 Dataset

After the three experiments on 728×90 dataset with t values equal 7, 14, and 30 we can say that the vanilla-LSTM model is more accurate than the vanilla-MLP model in predicting the average eCPM around certain date.

After all the experiments that explained in **Experiment 3.1**, we can conclude that:

- Vanilla-LSTM model can achieves the best model accuracy on 300×250, 300×600, and 728×90 datasets with t values equal 7, 14, and 30.
- Vanilla-MLP model can achieves the best model accuracy on 970×250 dataset with t values equal 7, 14, and 30.
- The pattern of the 970×250 dataset is easier than the pattern of 300×250, 300×600, and 728×90 datasets.

- Vanilla-LSTM and vanilla-MLP models can forecast the average eCPM with short periods, with t equal 7 and 14, better than the average eCPM with long periods, with t equals 30. So, we can say that these two models are useless with t = 30.

We concluded all the error values that the Vanilla-LSTM and Vanilla-MLP achieved with the four datasets during forecasting the average eCPM around certain date with different t values equal 7, 14, and 30 using *MSE*, *NMSE*, and *Percentage-Error* in Table 5.1. For example, from Table 5.1, you can see horizontally the different *MSE*, *NMSE*, and *Percentage-Error* values that the vanilla-LSTM and vanilla-MLP achieved with the 300×250 dataset with t equals 7, 14, and 30. You can see that Vanilla-LSTM achieved minimum *MSE*, *NMSE*, and *Percentage-Error* values comparable with Vanilla-MLP. The highlighted numbers represent the numbers that have minimum model error.

Table 5.1 Conclusion for the *MSE*, *NMSE*, and *Percentage-Error* values for all datasets with vanilla-LSTM and vanilla-MLP models with t values equal 7, 14, and 30

Datasets	**Error**	**Vanilla-LSTM t = 7**	**Vanilla-MLP t = 7**	**Vanilla-LSTM t = 14**	**Vanilla-MLP t = 14**	**Vanilla-LSTM t = 30**	**Vanilla-MLP t = 30**
300×250	*MSE*	**2.5726**	3.0073	**1.9891**	2.3542	**1.3963**	1.9091
	NMSE	**0.709**	0.7173	**0.3468**	0.3742	**0.2160**	0.3132
	Percentage-Error	**53.31%**	54.82%	**43.42%**	44.98%	**35.80 %**	40.28 %
300×600	*MSE*	**3.1656**	7.0541	**2.5759**	4.0606	**2.7044**	4.5279
	NMSE	**0.1483**	0.4622	**0.1109**	0.2289	**0.1202**	0.2445
	Percentage-Error	**30.67%**	41.67%	**27.70%**	34.41%	**30.81%**	34.66%
970×250	*MSE*	8.3856	**2.1123**	8.7115	**5.8440**	8.4554	**4.2029**
	NMSE	0.2205	**0.2021**	0.2240	**0.1912**	0.2163	**0.1707**
	Percentage-Error	38.96%	**33.58%**	39.89%	**32.87%**	39.72%	**30.28%**
728×90	*MSE*	**2.4737**	3.3251	**2.1553**	3.5636	**2.0471**	2.6487
	NMSE	**0.1256**	0.2860	**0.1028**	0.2953	**0.1038**	0.1780
	Percentage-Error	**27.05%**	33.65%	**25.42%**	35.32%	**26.34%**	33.33%

5.3.2 Results of Experiment 3.2

In subsection 5.3.1 we studied the results of the LSTM and MLP models with the four datasets that we have with different t values in predicting the average eCPM around certain date. From the previous study we concluded that we should use the LSTM model with 300×250, 300×600, and 728×90 datasets and the MLP model with 970×250 dataset.

In **Experiment 3.2** we will study the results of LSTM model and KNN model in predicting the average eCPM with 300×250, 300×600, and 728×90 datasets with t equals 7. We run this experiment to validate our claim that these three datasets are complex and using a complicated model, such as LSTM model, will be more efficient than using a simple model, such as KNN, with these three datasets.

In **Experiment 3.2**, the blue curve with the start marker represents the vanilla-LSTM prediction curve. The red curve with the triangle-up marker represents the KNN prediction curve. The black curve with the circle marker represents the target curve.

Figure 5.17 shows the vanilla-LSTM and KNN prediction curves with the target curve using t equals 7 on 300×250 dataset. From Figure 5.17 we can see that the vanilla-LSTM prediction curve can reach most of the data points that exist in the target curve. Also, the KNN prediction curve has the same target curve pattern but it can not predict the average eCPM well especially in the period between 11/2020 and 01/2021.

After this comparison we can say that, Vanilla-LSTM model can predict the average eCPM better that KNN model in 300×250 dataset with t equals 7.

Figure 5.18 shows the two prediction vanilla-LSTM and KNN curves with t equals 7 using 300×600 dataset. From this figure we can see that there is no big difference between the two prediction curves in most of the prediction periods except in the period before and after 12/2020. The KNN prediction curve can not predict well in this period. Consequently, we can say that the vanilla-LSTM model can predict the average eCPM better than the KNN model.

Figure 5.19 shows the vanilla-LSTM model versus the KNN model with t equals 7 on 728×90 dataset. In this figure we can see the same previous observation that exist in Figure 5.18. Since there is no big difference between the vanilla-LSTM prediction curve compared with the KNN prediction curve. Both prediction curves can reach the target curve in most prediction periods except in the period before and after 12/2020. Since the dataset has high peaks in this period.

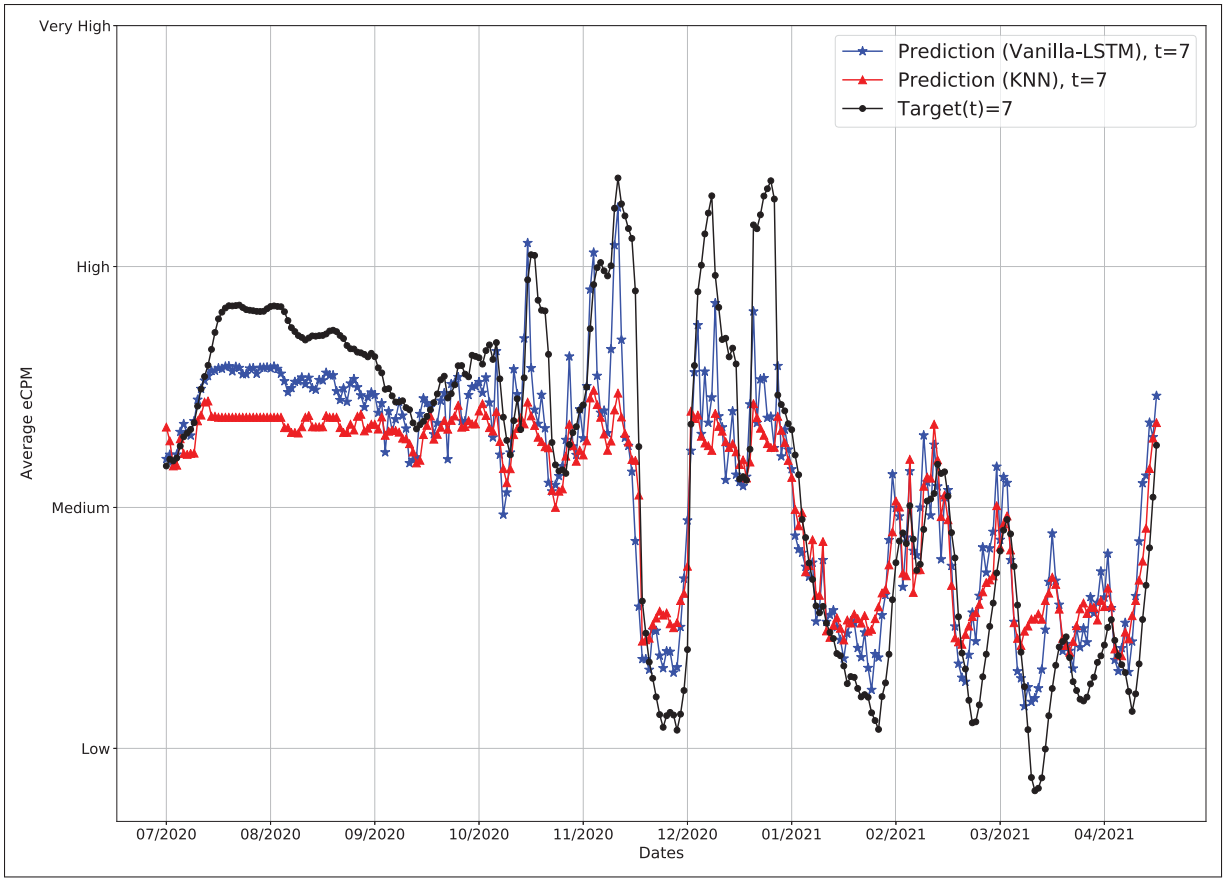

Figure 5.17 LSTM Vs. KNN with $t = 7$ and $K = 12$ on 300×250 Dataset

Nevertheless, we can say that the vanilla-LSTM model is better than the KNN model since it can reach some data points better than the KNN model.

At the end of **Experiment 3.2**, we can conclude that the simple models such as KNN model can not solve our problem very well with 300×250, 300×600, and 728×90 datasets. Since the pattern of these three datasets is not easy to predict. So, we should use a more complicated model like LSTM model to achieve more accurate results.

From the previous observations on experiment 3.2, we can conclude that the vanilla-LSTM model is the best model that we can use to predict the average eCPM around certain date to achieves the best model accuracy.

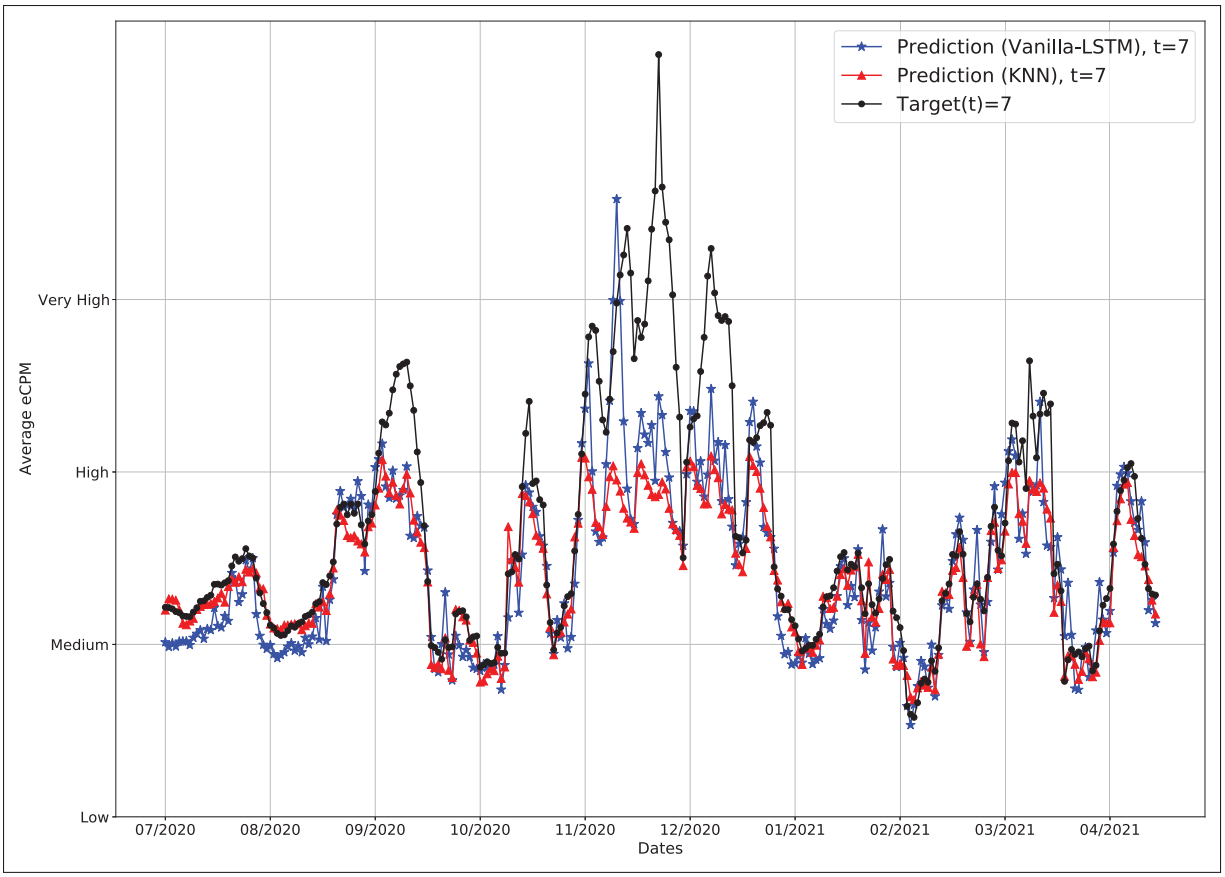

Figure 5.18 LSTM Vs. KNN with $t = 7$ and $K = 25$ on 300×600 Dataset

5.3.3 Results of Experiment 3.3

In this experiment, we will discuss the result of the comparison between the vanilla-MLP and KNN models for predicting the average eCPM that is explained in the target curve with t equal 7 on 970×250 dataset. We run this experiment to prove our claim that the 970×250 dataset is not a complex dataset to use a complicated model such as the LSTM model. Also, it is not a very easy dataset to use a simple model such as the KNN model.

Figure 5.20 shows the result of the comparison between the two models on 970×250 with t equals 7. In Figure 5.20 we can see that, the vanilla-MLP prediction curve, the red curve, can predict most of the target data points better than the KNN prediction curve, the blue curve. The KNN prediction curve can not reach the target data points in most periods.

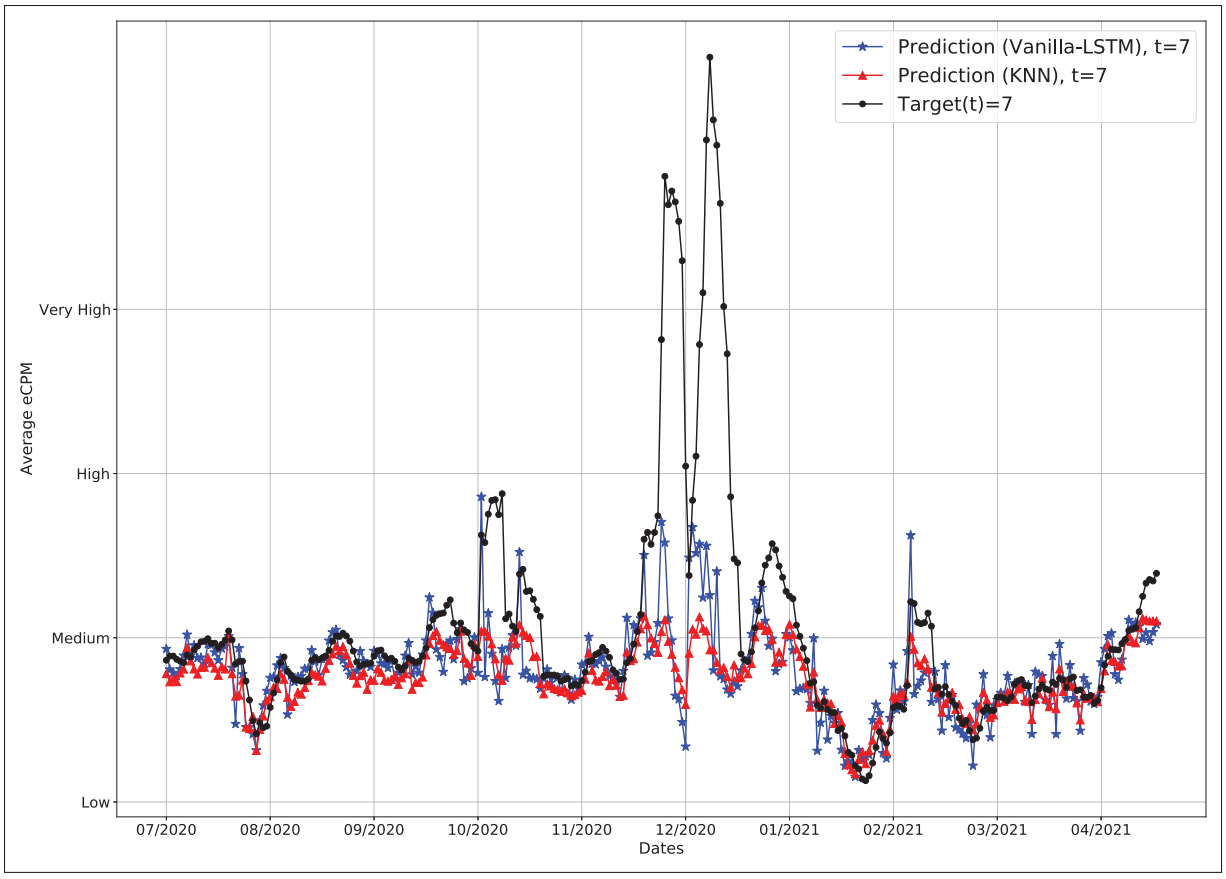

Figure 5.19 LSTM Vs. KNN with $t = 7$ and $K = 12$ on 728×90 Dataset

Finally, we can say that the pattern of the 970×250 dataset is not simple enough to use a simple model like KNN model. Also, it is not very complicated to use a complex model like LSTM in average eCPM forecasting. Consequently, we can conclude that the vanilla-MLP model is the suitable model that can achieve the best model accuracy on the 970×250 dataset.

5.4 The Final Models architecture

In this section, we will explain the architecture of the two final used models (LSTM and MLP). Also, we will show the results of the two models with the datasets that helped each model achieve the best accuracy.

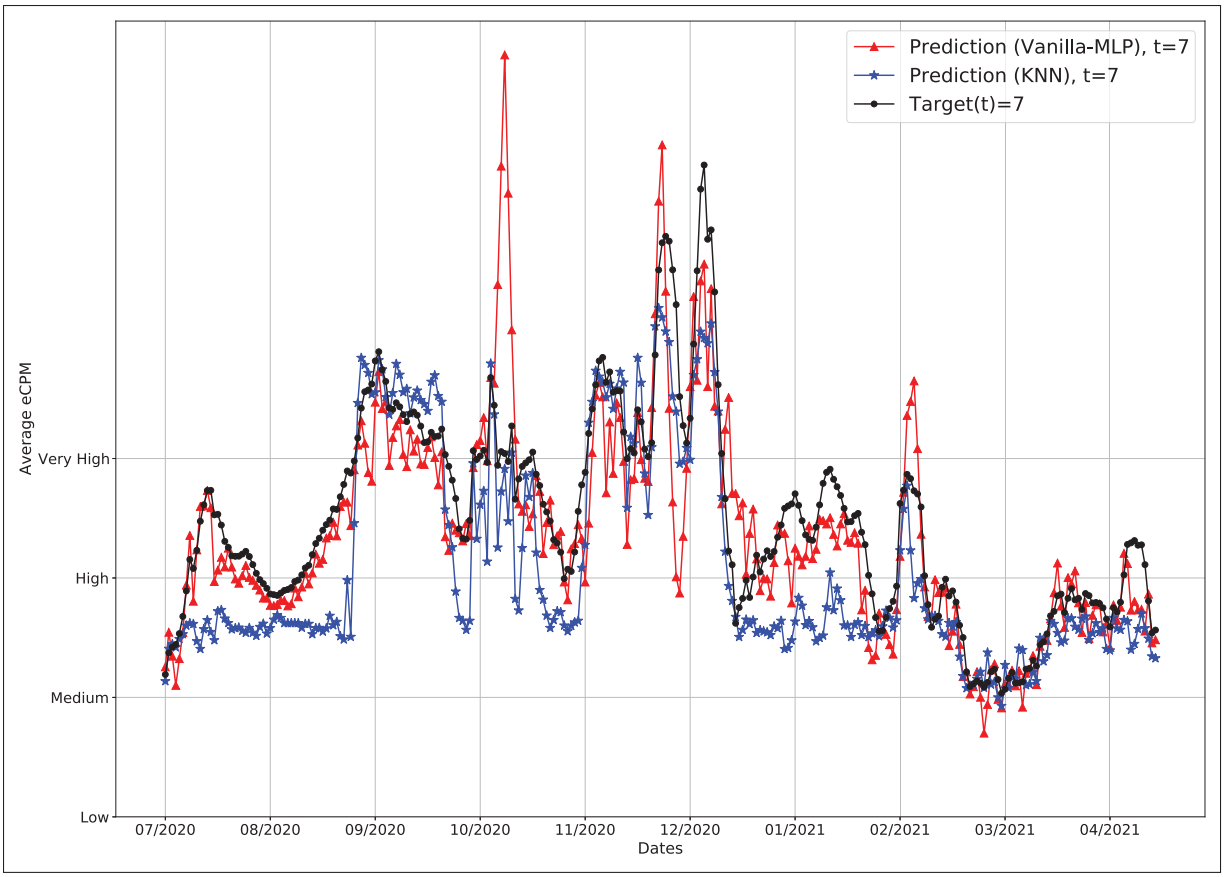

Figure 5.20 MLP Vs. KNN with $t = 7$ and $K = 12$ on 970×250 Dataset

5.4.1 Predictive Model (Vanilla-LSTM Model)

After running different experiments on the 4 datasets that we have, We selected the the vanilla-LSTM model as the final model with 300×250, 300×600, 728×90 datasets with target (t) values equal 7, 14, and 30. We built vanilla-LSTM architecture with input layer and output layer. We used 50 and one nodes for the input and output layers respectively, as described in chapter 4. Also, we used *Tanh* as an activation function, *MSE* as a loss function, and *Adam* optimizer, these methods are described in more details in chapter 4. The model is trained using 300 epochs. Some of the technologies that used in training the model are (*Python*, *Pandas*, *TensorFlow*, and *Keras*).

5.4.1.1 Results of Vanilla-LSTM model on 300 × 250 dataset

Figure 5.21, Figure 5.22, and Figure 5.23 show the final results after using the vanilla-LSTM model on 300 × 250 dataset. These three figures show the prediction curve, after using the Vanilla-LSTM model, and the target curve with t values equal 7, 14, and 30 respectively. From these three figures we can conclude that the vanilla-LSTM model can forecast the average eCPM with t equal 7 and 14 better than t equals 30.

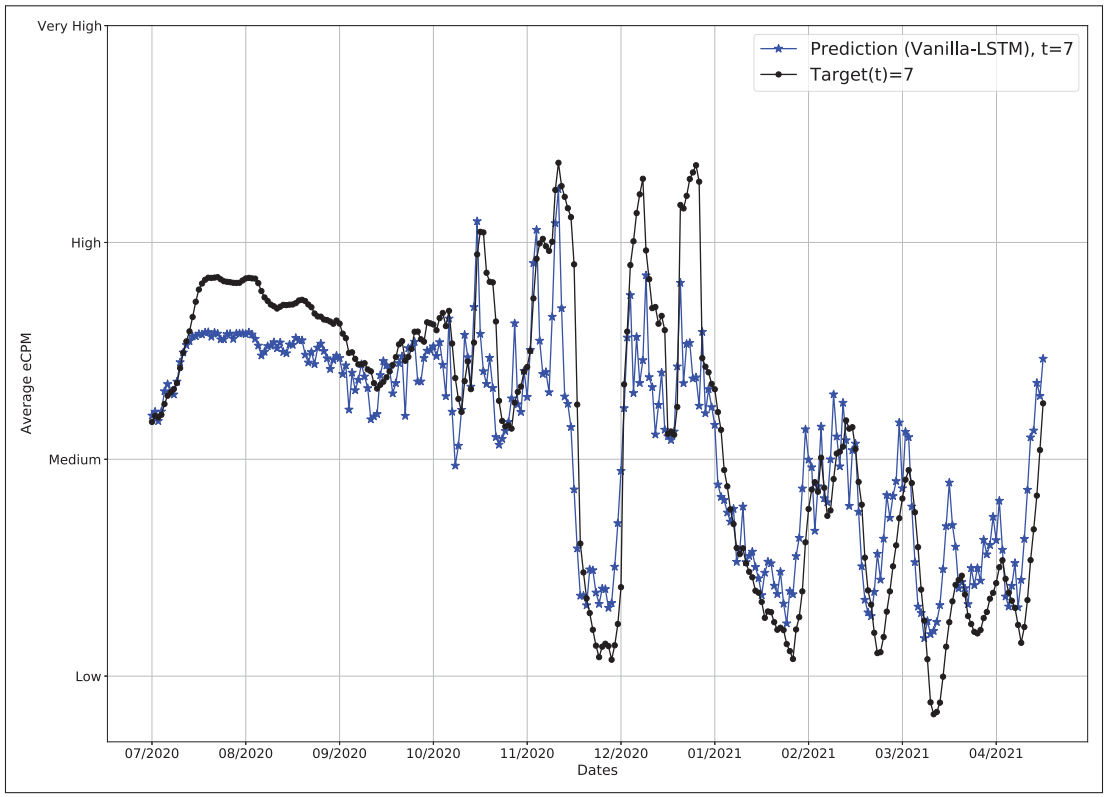

Figure 5.21 Vanilla-LSTM model with $t = 7$ on 300 × 250 Dataset

5.4.1.2 Results of Vanilla-LSTM model on 300 × 600 dataset

Figure 5.24, Figure 5.25, and Figure 5.26 show the prediction curve, after using the Vanilla-LSTM model, and the target curve with t values equal 7, 14, and 30 respectively on 300 × 600 dataset. From these three figures, we can conclude that the vanilla-LSTM model can forecast the average eCPM with t equal 7 and 14 well. While it is useless with t equals 30.

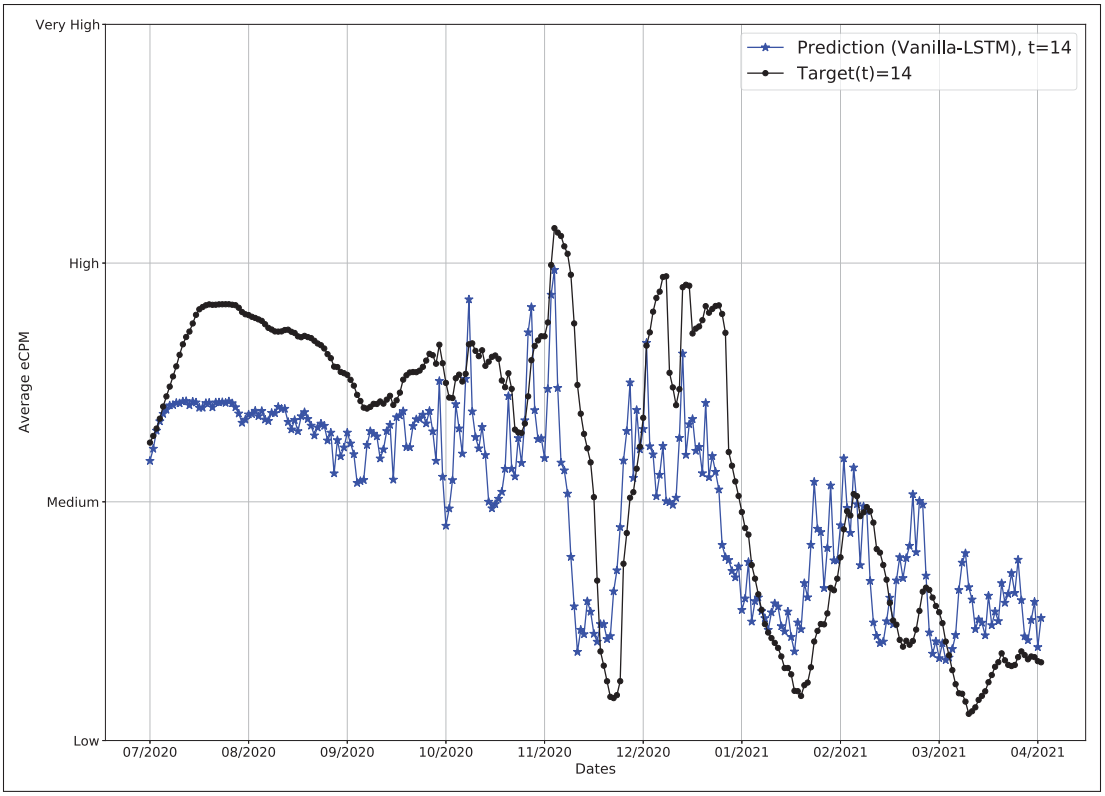

Figure 5.22 Vanilla-LSTM model with $t = 14$ on 300×250 Dataset

5.4.1.3 Results of Vanilla-LSTM model on 728×90 dataset

Figure 5.27, Figure 5.28, and Figure 5.29 show the prediction curve, after using the Vanilla-LSTM model, and the target curve with t values equal 7, 14, and 30 respectively on 728×90 dataset. From these three figures, we can conclude that the vanilla-LSTM model can forecast the average eCPM with t equal 7 and 14 better than t equals 30.

5.4.2 Predictive Model (MLP Model)

We trained LSTM, MLP, and KNN models on 970×250 dataset and the best model accuracy is achieved using MLP model, as explained in details in chapter 4 and chapter 5. We used vanilla-MLP model with 50 nodes in the input layer and one node in the output layer, explained in chapter 4. We trained the model using *ReLU* activation function, *MSE* as a loss function, and

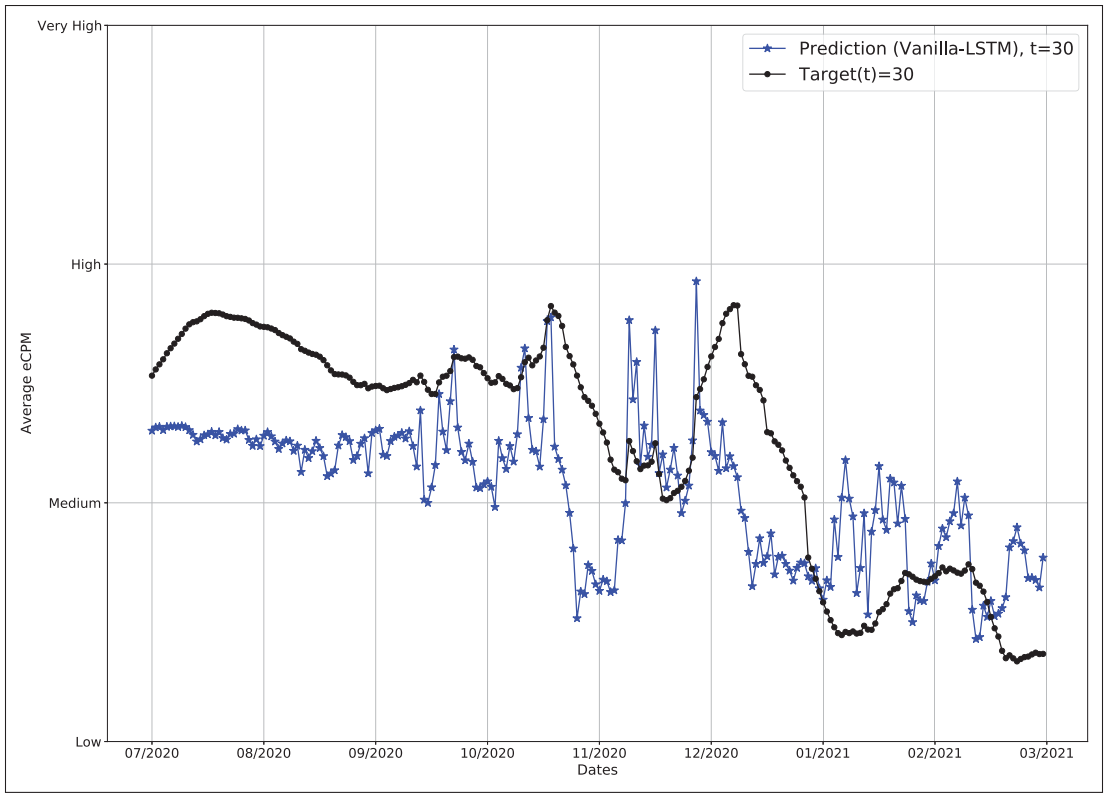

Figure 5.23 Vanilla-LSTM model with $t = 30$ on 300×250 Dataset

Adam optimizer. The functions and the optimizer are explained in chapter 4. The vanilla-MLP mode is trained with different target (t) values equal 7, 14, and 30 with number of epochs equals 300. We used technologies such as *Python*, *Pandas*, *TensorFlow*, and *Keras* in model training.

5.4.2.1 Results of Vanilla-MLP model on 970×250 dataset

Figure 5.30, Figure 5.31, and Figure 5.32 show the prediction curve, after using the Vanilla-MLP model, and the target curve with t values equal 7, 14, and 30 respectively on 970×250 dataset. From these three figures, we can conclude that the vanilla-MLP model can forecast the average eCPM with t equal 7 and 14 better than t equals 30.

5.5 Discussion

At the end of this chapter, we can conclude that the Vanilla-LSTM model is the best model for 300×250, 300×600, and 728×90 datasets. While Vanilla-MLP is the best model for the 970×250 dataset. However, one question may raise here which is: can we conclude that these two models are always the best for these datasets?

Since we have four datasets with different ad sizes, we can not ensure that the LSTM model or MLP is the best one with a specific dataset all the time. This is due to the fact that the behavior and pattern of such dataset are changing with time. The pattern of the datasets may change at any time based on the factors surrounding it.

To keep using the best model all the time, we can employ post-deployment monitoring. We can regularly monitor the models behavior and based on the percentage error we can trigger different actions. For example, if the model's percentage error does not exceeded a certain threshold, we can continue using the same model. Otherwise, a new evaluation is required. In this case, we can retrain the three models again, LSTM, MLP, and KNN models, and select the model with the minimum error.

This continuous monitoring comes with some advantages and disadvantages. The main advantage of the continuous monitoring is saving the cost of scheduling regular retraining. The main disadvantage of continuous monitoring is the possibility of wasting possible improvements. To clearly explain this point, assume the used model achieves 11% error which is acceptable for the Industrial Partner (i.e., less than the specified threshold). Therefore, no retraining will be triggered. However, maybe there is another model that could achieve 5% error if we reevaluated the different models. In this case, there is 6% possible improvements we will not catch.

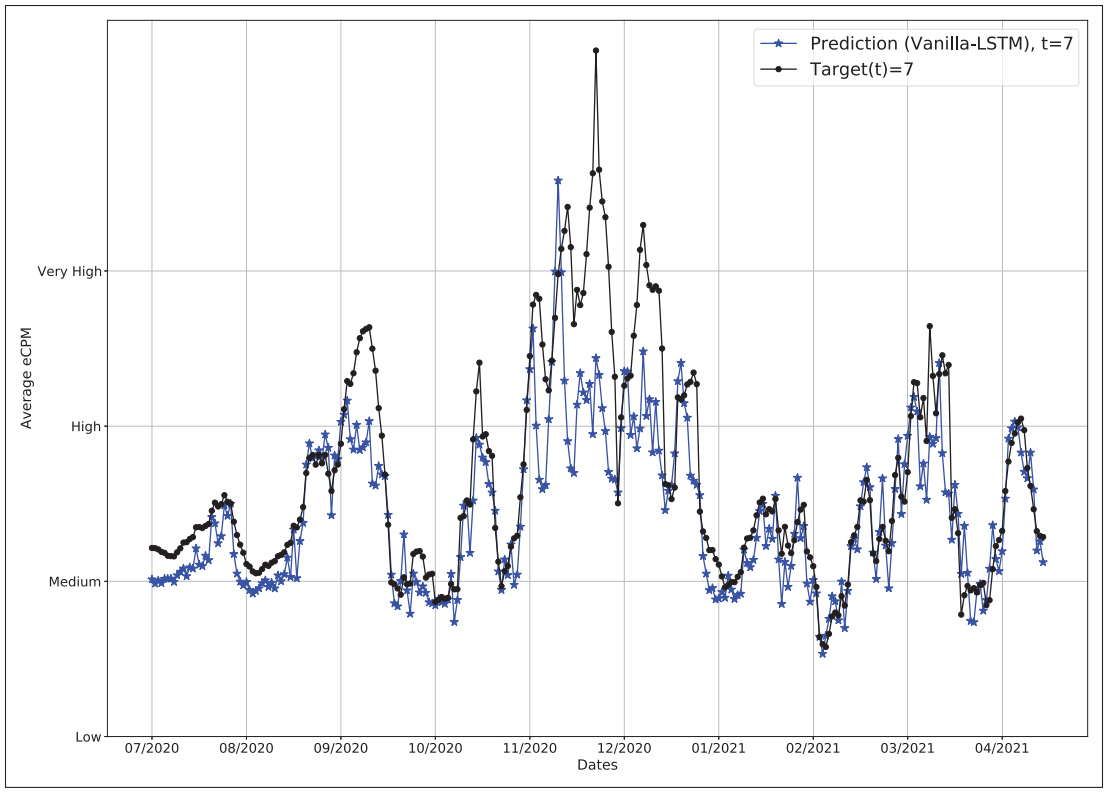

Figure 5.24 Vanilla-LSTM model with $t = 7$ on 300×600 Dataset

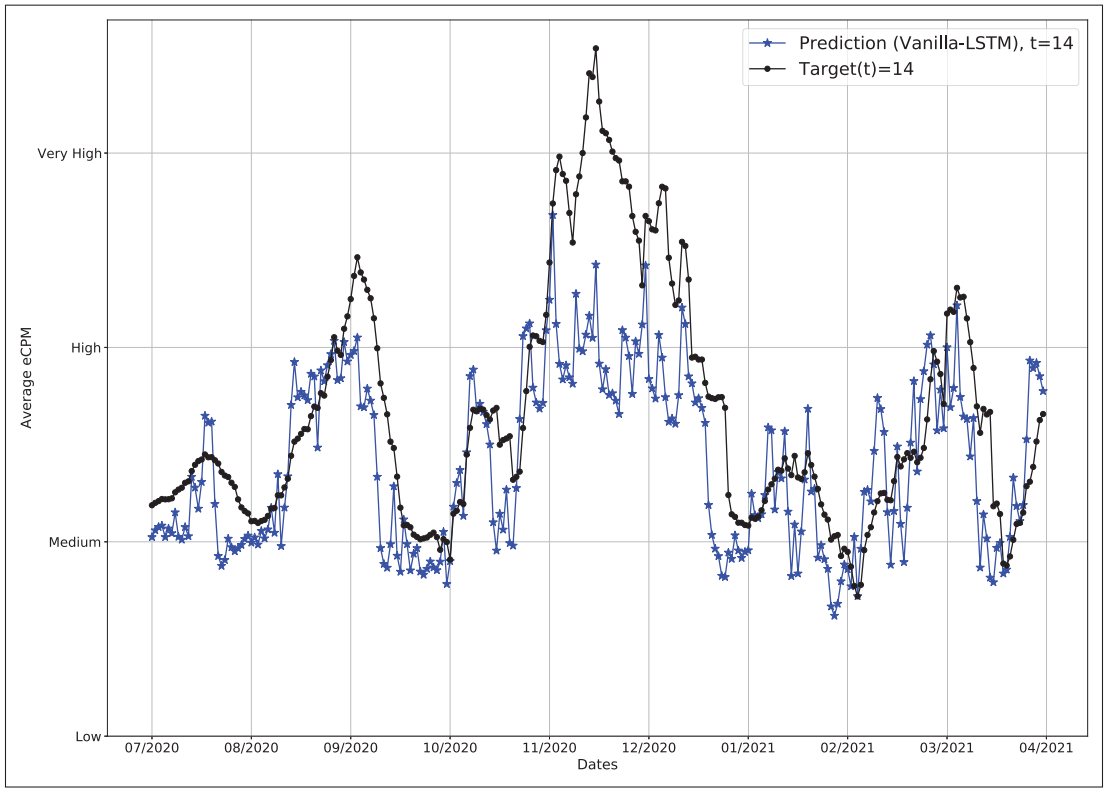

Figure 5.25 Vanilla-LSTM model with $t = 14$ on 300×600 Dataset

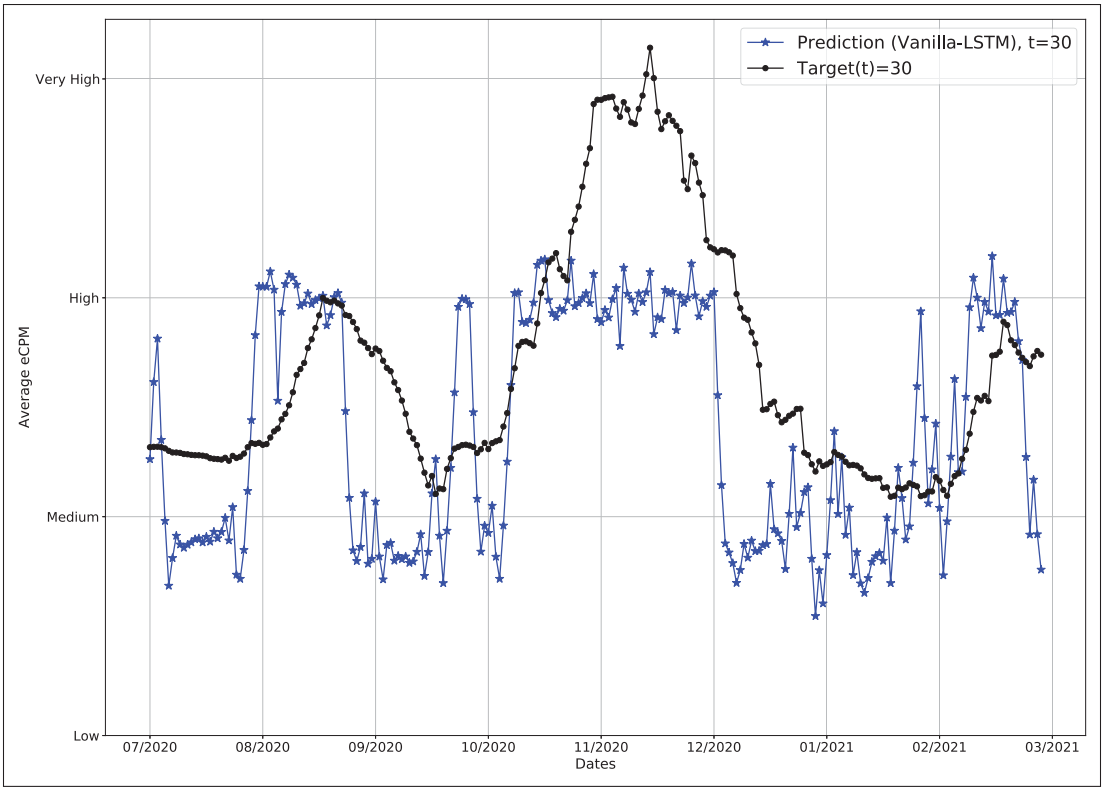

Figure 5.26 Vanilla-LSTM model with $t = 30$ on 300×600 Dataset

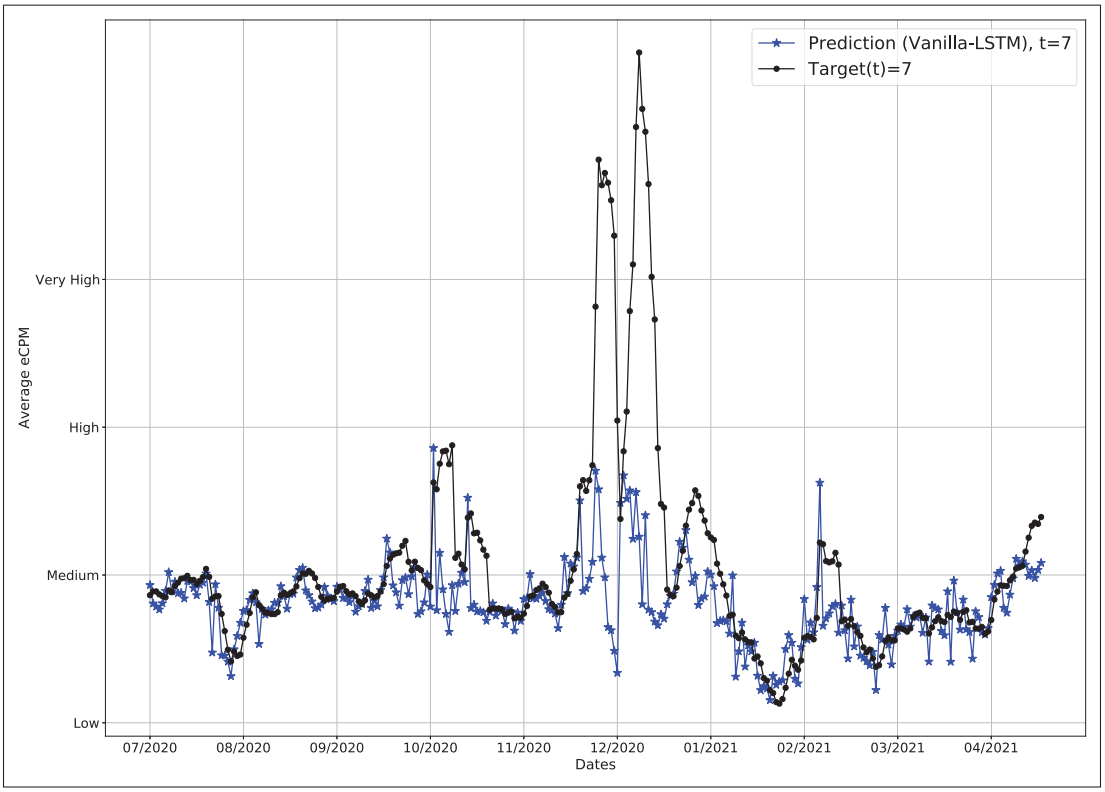

Figure 5.27 Vanilla-LSTM model with $t = 7$ on 728×90 Dataset

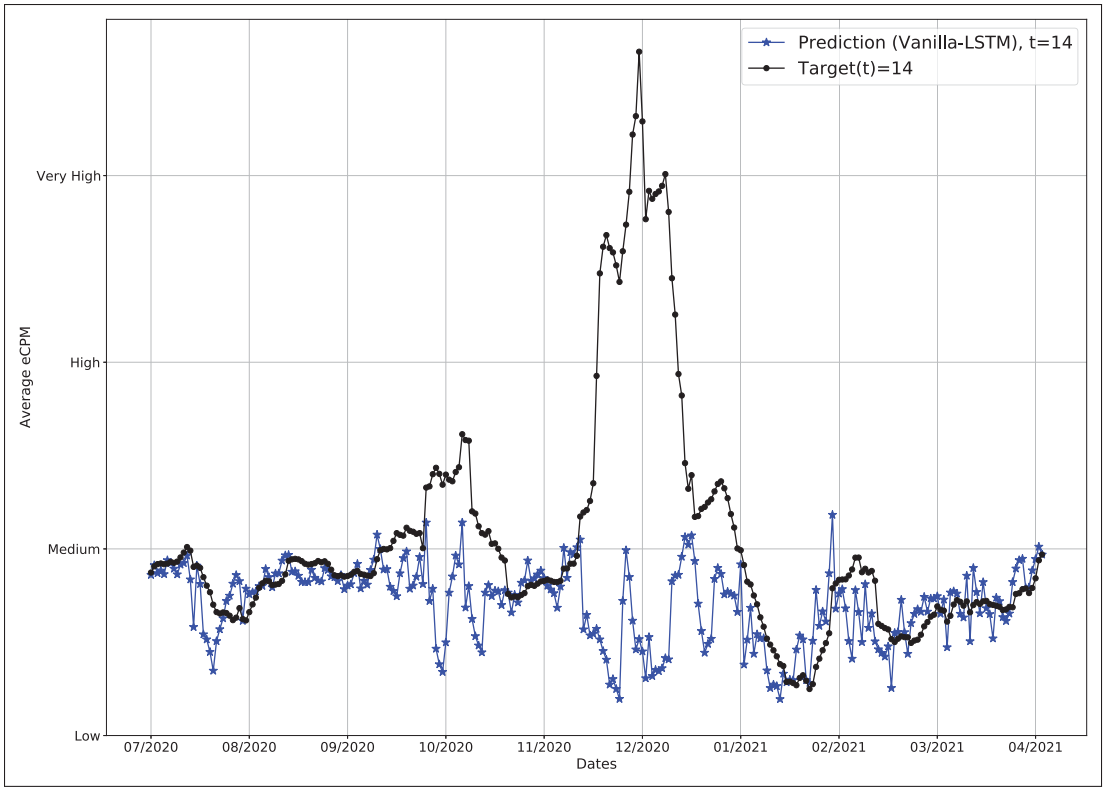

Figure 5.28 Vanilla-LSTM model with $t = 14$ on 728×90 Dataset

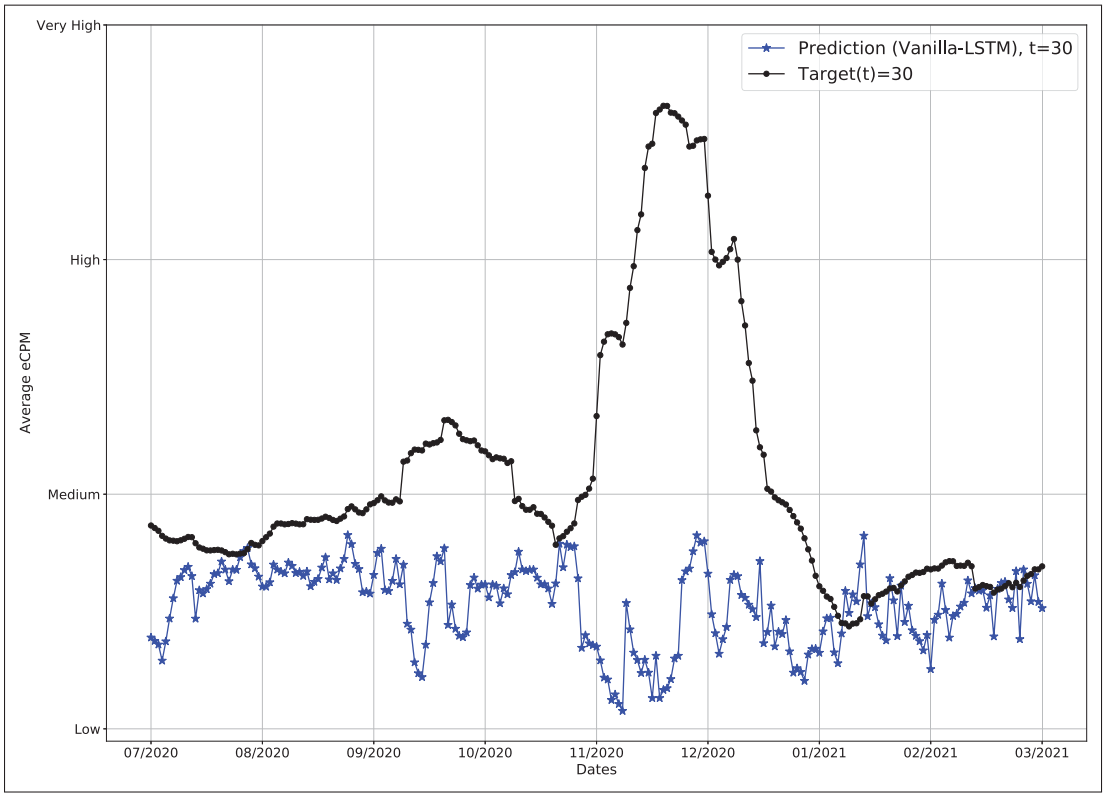

Figure 5.29 Vanilla-LSTM model with $t = 30$ on 728×90 Dataset

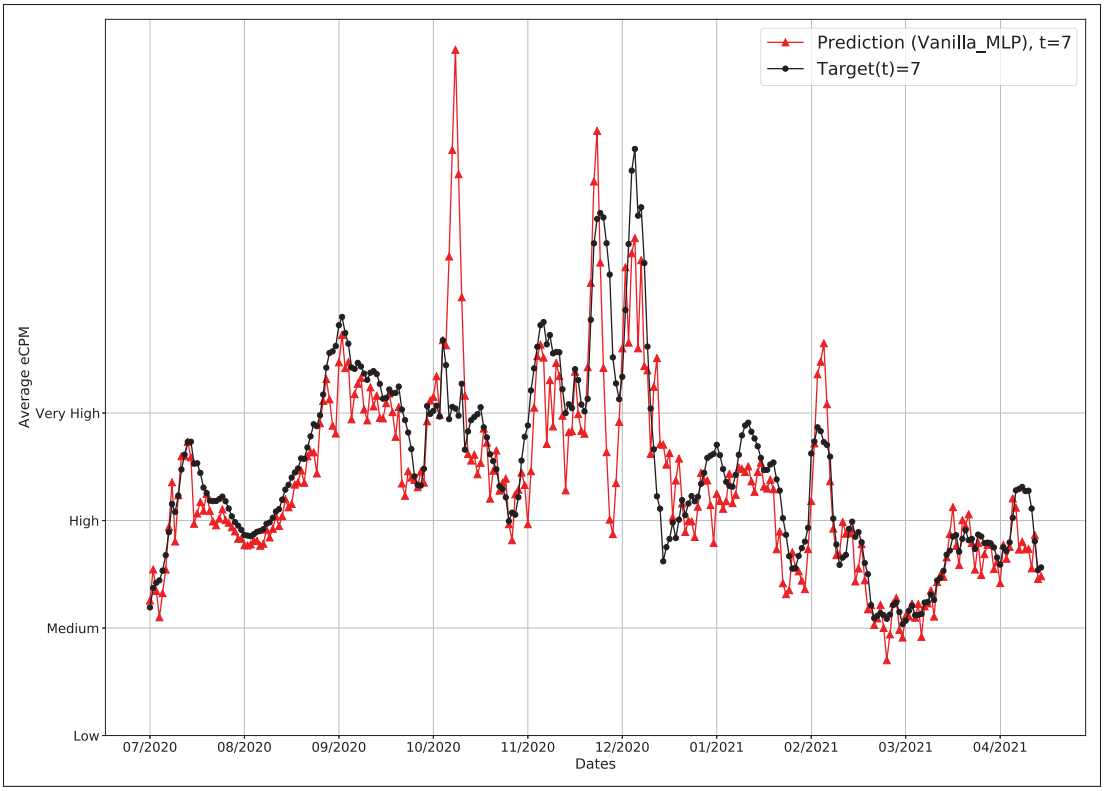

Figure 5.30 Vanilla-MLP model with $t = 7$ on 970×250 Dataset

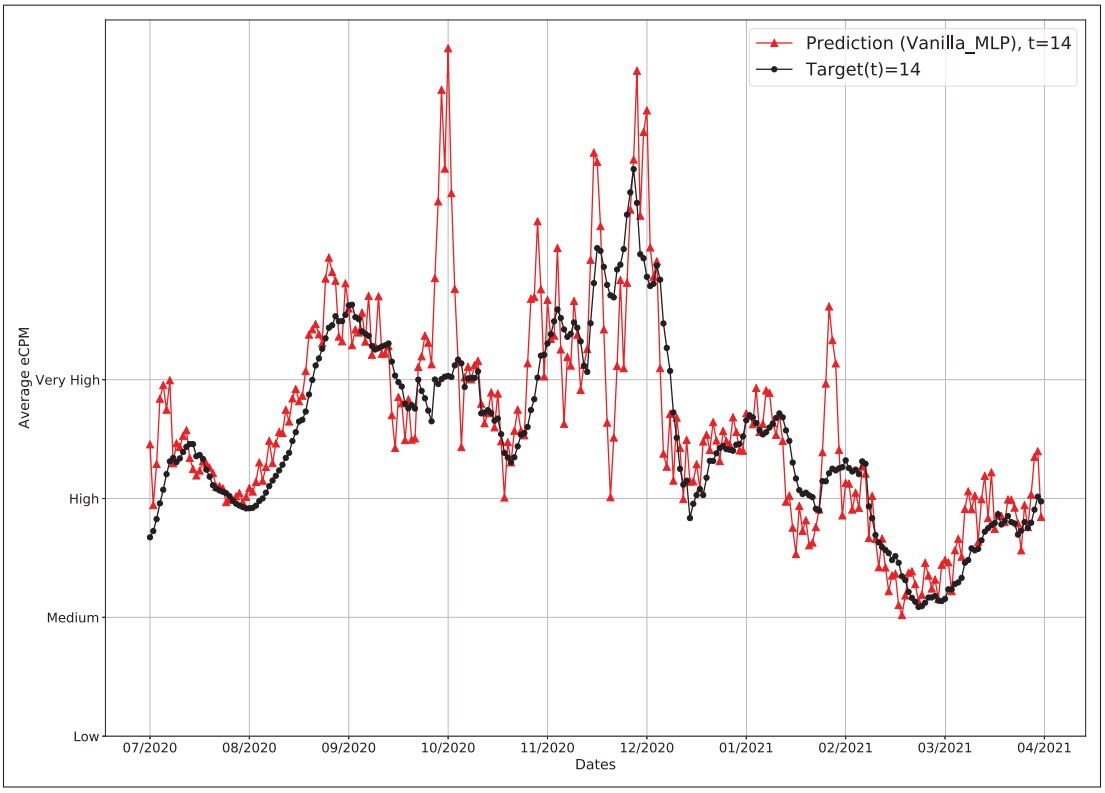

Figure 5.31 Vanilla-MLP model with $t = 14$ on 970×250 Dataset

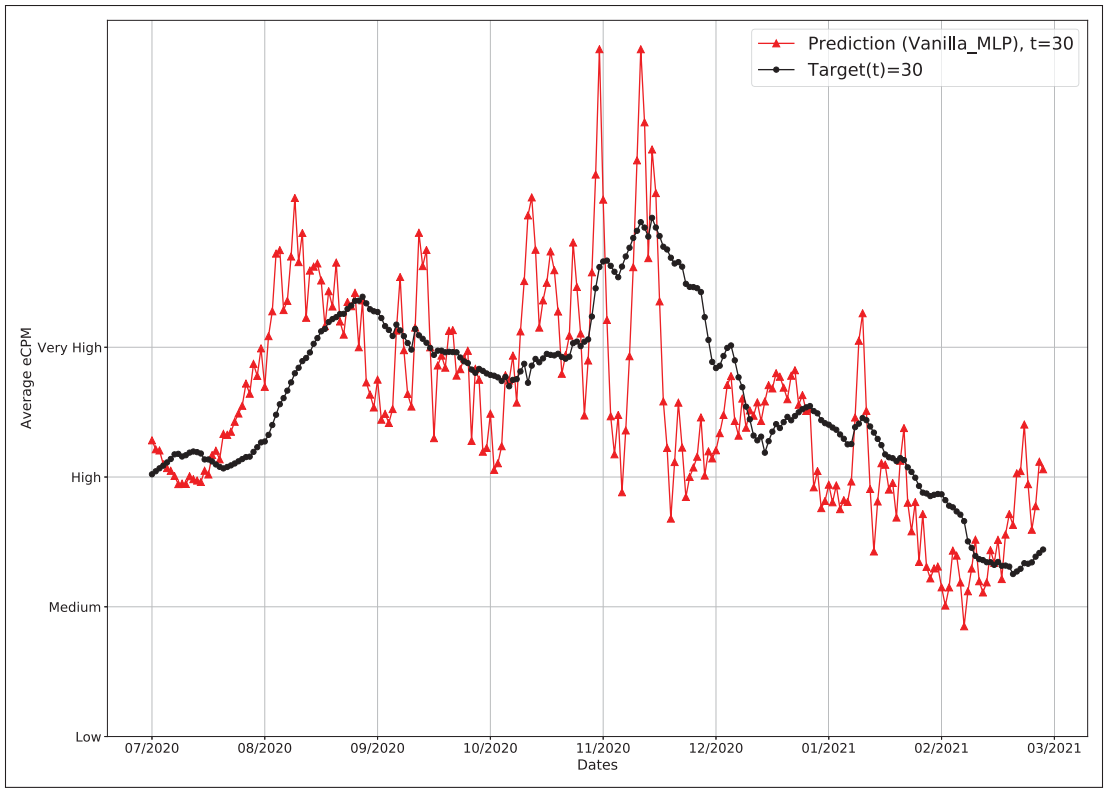

Figure 5.32 Vanilla-MLP model with $t = 30$ on 970×250 Dataset

CONCLUSION AND RECOMMENDATIONS

Conclusion

In this thesis, we predict the average eCPM around certain date for the next 7, 14, and 30 days using the eCPM time series history. We experimented different machine learning models to select the model that achieves the highest accuracy. All experiments are carried out using KNN, MLP, and LSTM models on four different datasets. From the obtained experimental results, we can conclude that there is no model performing better than other models for all datasets. However, the model accuracy depends on the used dataset. After comparing all models, we can conclude that LSTM model outperforms MLP and KNN models on 300×250, 300×600, and 728×90 datasets. Nevertheless, MLP model outperforms KNN and LSTM models on 970×250 dataset.

Recommendations

Through this thesis, we recommend that the suitable models for our problem are LSTM model on 300×250, 300×600, and 728×90 datasets and MLP model on 970×250 dataset. Although the obtained results are promising, the error margin could be further improved. The following points could be considered for future improvements:

- Smoothing out the predictions such that the predictions are not far from the previous eCPM value by certain margin.
- Solving the problem of negative eCPM prediction by post-processing. For example, we can replace the negative predictions with the minimum value in the input eCPM values.
- Adding more features that may improve the models accuracy such as the weather, holidays calendar, etc.

BIBLIOGRAPHY

Adhikari, R. & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

Austin, D., Seljan, S., Monello, J. & Tzeng, S. (2016). Reserve price optimization at scale. *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 528–536. doi: 10.1109/DSAA.2016.32.

Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*. Oxford university press.

Box, G. E., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Braei, M. & Wagner, S. (2020). Anomaly detection in univariate time-series: A survey on the state-of-the-art. *arXiv preprint arXiv:2004.00433*.

Brownlee, J. (2020, August, 20). A Gentle Introduction to the Rectified Linear Unit (ReLU) [Web Page]. Consulted at https://machinelearningmastery.com/.

Brownlee, J. (2021a, August, 5). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning [Web Page]. Consulted at https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

Brownlee, J. (2021b, July, 20). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras [Web Page]. Consulted at https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

Chatfield, C. (2000). *Time-series forecasting*. CRC Press. doi: 10.1201/9781420036206.

DeepAI. (2021, July, 1). Thompson Sampling [Web Page]. Consulted at https://deepai.org/machine-learning-glossary-and-terms/thompson-sampling.

Doe, J., Doe, J. & Doe, J. (1999). Title of the chapter. In Doe, J., Doe, J. & Doe, J. (Eds.), *Title of the book* (ed. 3, vol. 4, pp. 220-242). Location: Publisher.

Elsworth, S. & Güttel, S. (2020). Time series forecasting using LSTM networks: a symbolic approach. *arXiv preprint arXiv:2003.05672*.

Feng, Z., Lahaie, S., Schneider, J. & Ye, J. (2020). Reserve Price Optimization for First Price Auctions. *arXiv preprint arXiv:2006.06519*.

Garcia-Orellana, C. J., Gallardo-Caballero, R., Macias-Macias, M. & Gonzalez-Velasco, H. (2007). SVM and Neural Networks comparison in mammographic CAD. *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3204-3207. doi: 10.1109/IEMBS.2007.4353011.

Gardner, M. W. & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15), 2627–2636. doi: 10.1016/S1352-2310(97)00447-0.

Gers, F. A., Schmidhuber, J. & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451–2471. doi: 10.1162/089976600300015015.

Gers, F. A., Eck, D. & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01* (pp. 193–200). Springer. doi: 10.1007/978-1-4471-0219-9_20.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10), 2222–2232. doi: 10.1109/TNNLS.2016.2582924.

Ha, L. (2008). Online advertising research in advertising journals: A review. *Journal of Current Issues & Research in Advertising*, 30(1), 31–48. doi: 10.1080/10641734.2008.10505236.

Hayes, A. (2021, August, 2). Reserve Price [Web Page]. Consulted at https://www.investopedia.com/terms/r/reserve-price.asp.

He, X., Cai, D. & Niyogi, P. (2006). Laplacian score for feature selection. *Advances in neural information processing systems*, pp. 507–514.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

IAB. (2021, July, 2). Internet advertising revenue report [Web Page]. Consulted at https://www.iab.com/wp-content/uploads/2020/05/FY19-IAB-Internet-Ad-Revenue-Report_Final.pdf.

INMOBI. (2021, October 15). In-App Monetization: What Is A Second Price Auction And How Does It Works? [Web Page]. Consulted at https://www.inmobi.com/blog/2018/10/24/what-is-a-second-price-auction-and-how-does-it-work-video.

Karim. (2021, August, 18). Deep Learning via Multilayer Perceptron Classifier [Web Page]. Consulted at https://dzone.com/articles/deep-learning-via-multilayer-perceptron-classifier.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Köppen, M. (2000). The curse of dimensionality. *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, 1, 4–8.

Kramer, O. (2013). K-Nearest Neighbors. In Kramer, O. (Ed.), *Dimensionality Reduction with Unsupervised Nearest Neighbors* (pp. 13–23). Berlin, Heidelberg: Springer Berlin Heidelberg.

Mahalakshmi, G., Sridevi, S. & Rajaram, S. (2016). A survey on forecasting of time series data. *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, pp. 1–8. doi: 10.1109/ICCTIDE.2016.7725358.

Match2One. (2021, October, 28). Banner Sizes: The Must-Have Banners of 2021 [Web Page]. Consulted at https://www.match2one.com/blog/standard-banner-sizes/.

Noriega, L. (2005). Multilayer perceptron tutorial. *School of Computing. Staffordshire University*.

Phi, M. (2021, August, 10). Towards Data Science: Illustrated Guide to LSTM's and GRU's: A step by step explanation [Web Page]. Consulted at https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

Piccolo, D. (1990). A distance measure for classifying ARIMA models. *Journal of time series analysis*, 11(2), 153–164. doi: 10.1111/j.1467-9892.1990.tb00048.x.

Rhuggenaath, J., Akcay, A., Zhang, Y. & Kaymak, U. (2019a). Optimizing reserve prices for publishers in online ad auctions. *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pp. 1–8. doi: 10.1109/CIFEr.2019.8759123.

Rhuggenaath, J., Akcay, A., Zhang, Y. & Kaymak, U. (2019b). A PSO-based algorithm for reserve price optimization in online ad auctions. *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2611–2619. doi: 10.1109/CEC.2019.8789915.

Rosenkranz, S. & Schmitz, P. W. (2007). Reserve prices in auctions as reference points. *The Economic Journal*, 117(520), 637–653. doi: 10.1111/j.1468-0297.2007.02044.x.

ROXOT. (2021, August, 2). How to set unified price floors in GAM first-price auction [Web Page]. Consulted at https://roxot.com/blog/price-floors-first-price-auction.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1985). *Learning internal representations by error propagation*.

Salvador, S. & Chan, P. (2005). Learning states and rules for detecting anomalies in time series. *Applied Intelligence*, 23(3), 241–255. doi: 10.1007/s10489-005-4610-3.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85–117. doi: 10.1016/j.neunet.2014.09.003.

Schmidhuber, J., Hochreiter, S. et al. (1997). Long short-term memory. *Neural Comput*, 9(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

Schraudolph, N. (2021, May, 27). Multi-layer networks [Web Page]. Consulted at https://cnl.salk.edu/~schraudo/teach/NNcourse/multilayer.html.

Sharma, A. (2021, September, 18). Theory Of Everything: Understanding Activation Functions in Neural Networks [Web Page]. Consulted at https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

Shelatkar, T., Tondale, S., Yadav, S. & Ahir, S. (2020). Web traffic time series forecasting using ARIMA and LSTM RNN. *ITM Web of Conferences*, 32, 03017. doi: 10.1051/itmconf/20203203017.

Shu, X., Zhang, L., Sun, Y. & Tang, J. (2020). Host-parasite: Graph LSTM-in-LSTM for group activity recognition. *IEEE transactions on neural networks and learning systems*. doi: 10.1109/TNNLS.2020.2978942.

Siami-Namini, S. & Namin, A. S. (2018). Forecasting economics and financial time series: ARIMA vs. LSTM. *arXiv preprint arXiv:1803.06386*.

Siami-Namini, S., Tavakoli, N. & Namin, A. S. (2018). A comparison of ARIMA and LSTM in forecasting time series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1394–1401. doi: 10.1109/ICMLA.2018.00227.

steldSSP Company. (2021, August, 2). Maximize your website ad revenue with Steld steldSSP supply side platform [Web Page]. Consulted at http://steld.com/supply-side-platform.1.html.

Stojiljković, M. (2021, August, 5). Real Python: Stochastic Gradient Descent Algorithm [Web Page]. Consulted at https://realpython.com/gradient-descent-algorithm-python/.

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F. & Troncoso, A. (2021). Deep Learning for Time Series Forecasting: A Survey. *Big Data*, 9(1), 3–21. doi: 10.1089/big.2020.0159.

Weytjens, H., Lohmann, E. & Kleinsteuber, M. (2019). Cash flow prediction: MLP and LSTM compared to ARIMA and Prophet. *Electronic Commerce Research*, 1–21. doi: 10.1007/s10660-019-09362-7.

Wojtowytsch, S. & Weinan, E. (2020). Can shallow neural networks beat the curse of dimensionality? A mean field training perspective. *IEEE Transactions on Artificial Intelligence*, 1(2), 121–129. doi: 10.1109/TAI.2021.3051357.

Yamak, P. T., Yujian, L. & Gadosey, P. K. (2019). A comparison between arima, lstm, and gru for time series forecasting. *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pp. 49–55. doi: 10.1145/3377713.3377722.

Yuan, S., Wang, J., Chen, B., Mason, P. & Seljan, S. (2014a). An empirical study of reserve price optimisation in real-time bidding. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1897–1906. doi: 10.1145/2623330.2623357.

Yuan, Y., Wang, F., Li, J. & Qin, R. (2014b). 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.

Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. (2021). *MultiLayer Perceptron*. Release 0.16.4.