東京大学大学院新領域創成科学研究科
社会文化環境学専攻

2020 年度
修　士　論　文

二項分類器による乗客数の予測
Predicting Passenger Number via Binary Classifier

2020 年 7 月 10 日 提出

指導教員 柴崎　亮介　教授

陳　柏嘉

Chen, Pochia

Abstract

Tokyo is the biggest city in Japan with the highest density of population. In many public places in Tokyo especially train stations, there must be thousands of people crossing these places every day. Considering the large number of passengers, many of these places are facing problems of discipline management and potential risks. In order to solve this problem, we are particularly interested in how to predict the human mobility with a high accuracy (and in high resolution). To achieve the target, we are going to explore the related factors which infect the prediction result heavily and dig out the potential information of long-time history trajectory, through several experiments. In this study, we propose a framework including a complete data-processing procedure and an deep learning architecture based binary predictor to predict the number of passengers who will enter the station or area. Specifically, we use the GPS mobility data from our cell phones, which is related to human activity closely so that it can record important information of human activity. First, we interpolate the trajectory with 5-minute requirement to complete the missing data. This process is conducted by Dijkstra's algorithm. Second, we transfer the trajectory into meshes to represent location because there is no need to predict the exact latitude and longitude. We specially focus on Tokyo station as research area in this study. Then, we extract the trajectory by period in stead of using the whole trajectory. Working on the whole trajectory may cause the model learning meaningless information and increase extra computing burden to our computer. At last, we apply our model which elaborately formulated based on state-of-the-art structures to experiments.

Keywords: ubiquitous and mobile computing, deep learning, binary predictor, long-term trajectory

目次

図目次

表目次

第 1 章

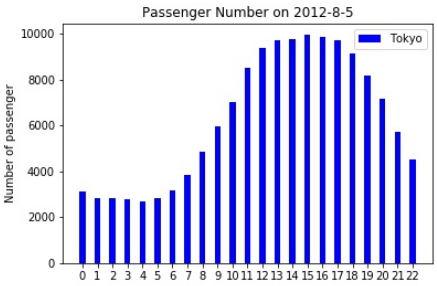
Introduction

1.1 Social Background

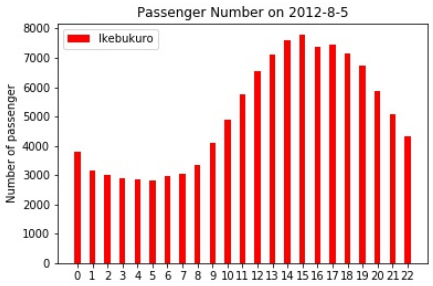
Trajectory prediction is of great significance for a wide range of location-based applications in intelligent transportation systems such as location-based advertising, route planning, traffic management,and early-warning systems. Tokyo is the biggest city in Japan with the most population. Thus, there is higher possibility to happen high population density event or situation than any other cities, which could increase the chaos or public safety risk in specific areas. Considering the large number of passengers, many places might face problems of discipline management and potential risks. Fig. 1.1 shows up the statistic number in 24 hours of 5 stations on August 5^{th}(Sunday), 2012. Fig. 1.2 shows up the statistic number in 24 hours of 5 stations on August 6^{th}(Monday), 2012. Fig. 1.3 shows up the statistic number in 1 week of Tokyo stations in the period of 2012.07.02-2012.07.08

When there are big events, it is more likely to happen high population density situation, e.g. concert, idol live house ,and soccer competition. Fig. 1.4 shows the chaos situations at the station. People will go to the target location waiting for the event in a period of time. In this period, the density of population grows up and it becomes crowded. This also happens after the event. And people living in Tokyo mainly depends on public transportation,like the train, to satisfy their traveling need, which increases the risk as well. In the communication time, it easily gets crowded and chaos at the platform and stations. At that moment, the risk grows up.

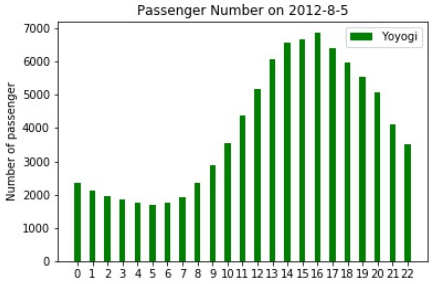
Hence, there is urgent need for those facilities and departments to get known the possible situation before the risk comes out as early as possible, so that they can prepare and prevent any miserable mistake. Fortunately, with the rapid development of location acquisition technology and upcoming 5G mobile technology, we have multiple methods to record, analyze and capture the human mobility pattern. This will highly benefit the transportation scheduling, urban regulation, and even



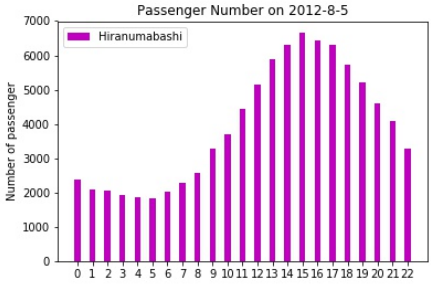
(a) Tokyo



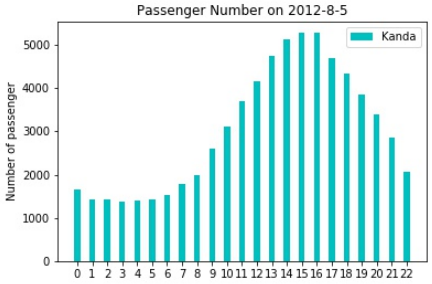
(b) Ikebukuro



(c) Yoyogi



(d) Hiranumabashi



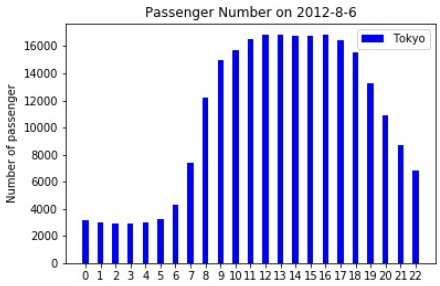
(e) Kanda

Fig.1.1 Passenger number on Sunday(2012.8.5)

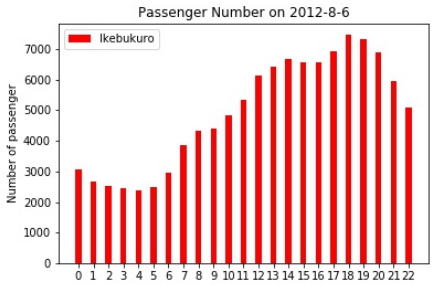
emergency management.

1.2 Research Background

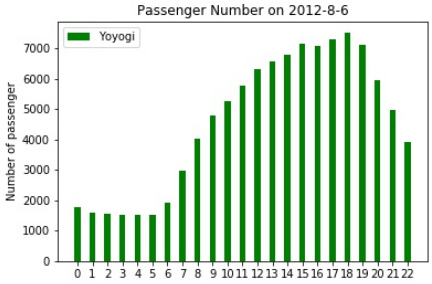
There have been already a lot of superior researches conducted by many other researchers, however the topic about long-term prediction based on long-term historical trajectory is still on the way. For long-term future prediction, the most essential challenge is how to make the model understand and capture the features in the extremely long trajectory. And it is even harder to re-



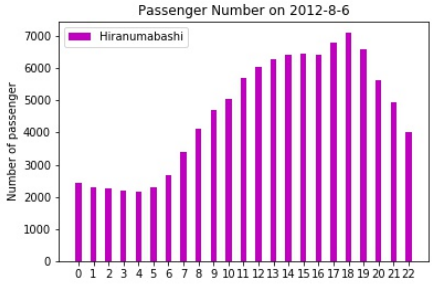
(a) Tokyo



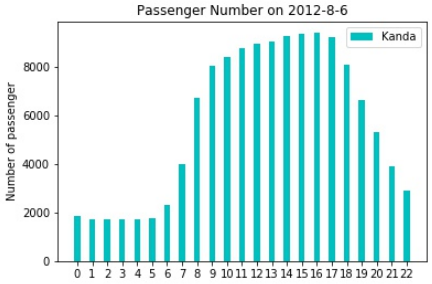
(b) Ikebukuro



(c) Yoyogi



(d) Hiranumabashi



(e) Kanda

Fig.1.2 Passenger number on Monday(2012.8.6)

produce the individual mobility patterns. We believe the the long-term future prediction is more meaningful than those shot-term future prediction because it has the ability to give a more holistic view of human mobility.

In the field of ubiquitous computing, human mobility analysis and prediction have been comprehensively explored from individual scale to city wide scale, while most of them are focusing on multi-class predictor in stead of a binary predictor. However their accuracy cannot satisfy the reality need because the multi-class predictor provides too many options resulting that none of the class option can have a really accuracy. Besides classification, regression is one of the idea

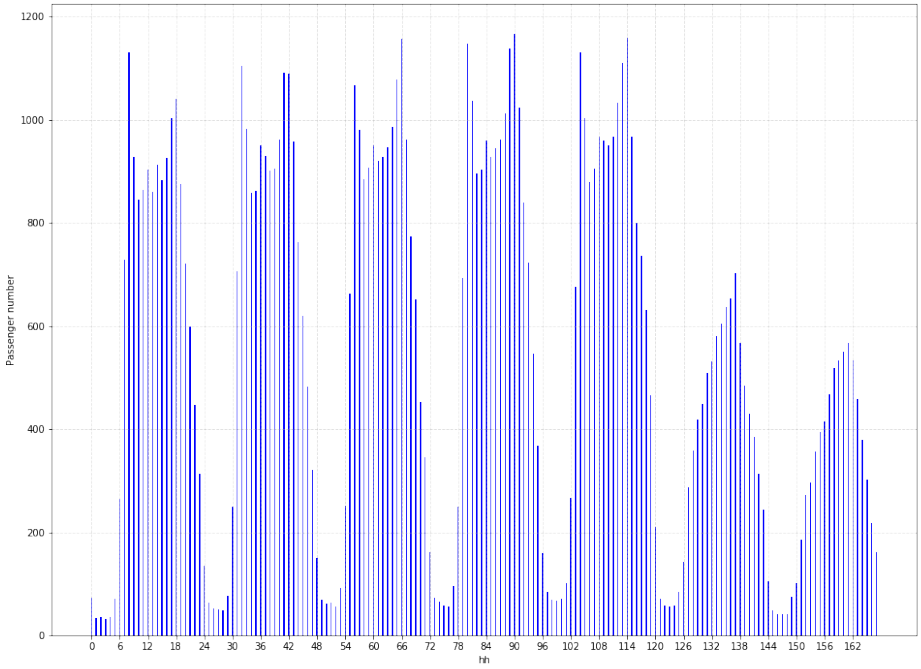


Fig.1.3 The passenger number at Tokyo station in 1 week with filtering the constant ID.

(a) Chaos of No. 15 typhoon

(b) Chaos of blizzard in winter

Fig.1.4 Draft of research target

to tackle the mobility prediction problem either, which can give an exact location in the future. In our case, we don't care or consider the exact location of a person, so we tend to solve this as a classification problem. The main obstacle for long-term mobility prediction problem is the sequence is so long that the existed methods cannot handle the massive information pretty well and capture the internal feature even further. Here are two related works about long-term tra-

jectory prediction based on historical trajectory. For DeepMove, it designs a historical attention module to capture periodicity and augment prediction accuracy. Fig. 1.5 shows up the general idea of DeepMove from Jie Feng and Yong Li. They come up with a framework that can capture multi-level periodical human nature or specifically regularity of mobility by historical attention module. Besides, ST-RNN from Qiang Liu extends a regular RNN by utilizing time and distance

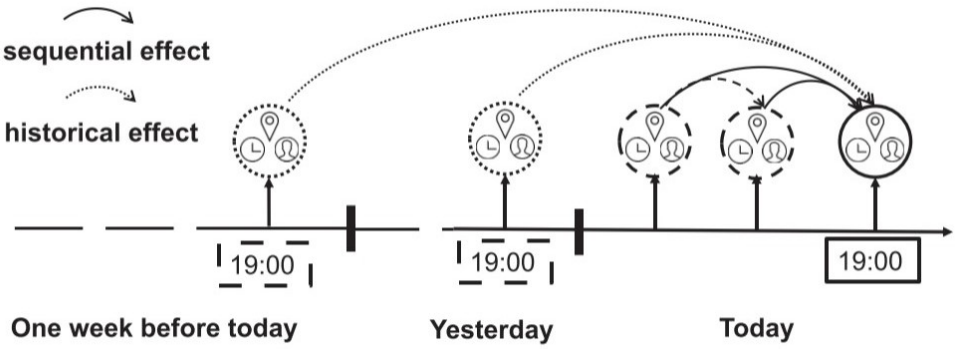


Fig.1.5 DeepMove

specific transition matrices to propose an ST-RNN model for predicting the next location for each individual' s movement. Instead of only one element in each layer of RNN, ST-RNN considers the elements in the local temporal contexts in each layer. In ST-RNN, to capture time interval and geographical distance information, we re- place the single transition matrix in RNN with time-specific transition matrices and distance-specific transition matrices. Fig. 1.6 shows the overview of the proposed model. In summary, most of the related work are applying multi-class predictor,

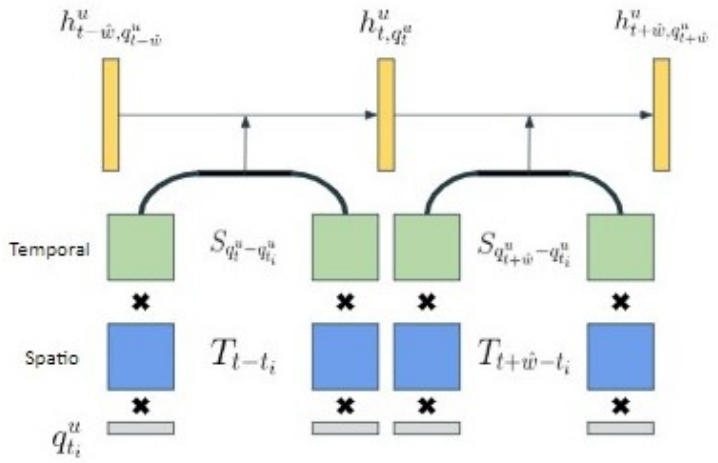


Fig.1.6 Overview of ST-RNN

and seldom of them are using long-term historical trajectory to conduct the research. So our work

focuses how to reproduce the long-term trajectory pattern at individual level.

1.3 Research Target

Given the background above, we are exploring a method that is based on the binary predictor to predict the human trajectory with higher resolution and accuracy, and longer future prediction. Making full use of this mechanism or method, we hope that we are able to provide with proper suggestion to event hosts or related traffic departments, helping discipline management and preventing the potential risks. In Fig. 1.7, we can see the chaos in station in the rush hour of going to work. For the best result of our research, it is possible to give a long-term prediction of the pas-

(a) (b)

Fig.1.7 Station chaos

senger number with high accuracy and in higher resolution by analyzing the long-term historical trajectory. And the length of historical trajectory can be adjusted according to the need. Eventually, we can build up one binary predictor for each station, and assemble the results together to give a general prediction in the city scale. With the higher accuracy result, the user can execute relevant measure to achieve better services. Fig. 1.8 and Fig. 1.9 show the general idea to provide better service and research application.

1.4 Research Application

In order to solve these problem we discussed in the first section, we are particularly interested in using long-history trajectory to predict the human mobility with a high accuracy and in higher resolution (1 km^2) via binary classifier. In this study, we propose a framework including a complete data-processing procedure and a machine learning architecture based binary predictor. First, we process the GPS trajectory with a 5-minute interval, so there are 288 positions in a day. Sec-

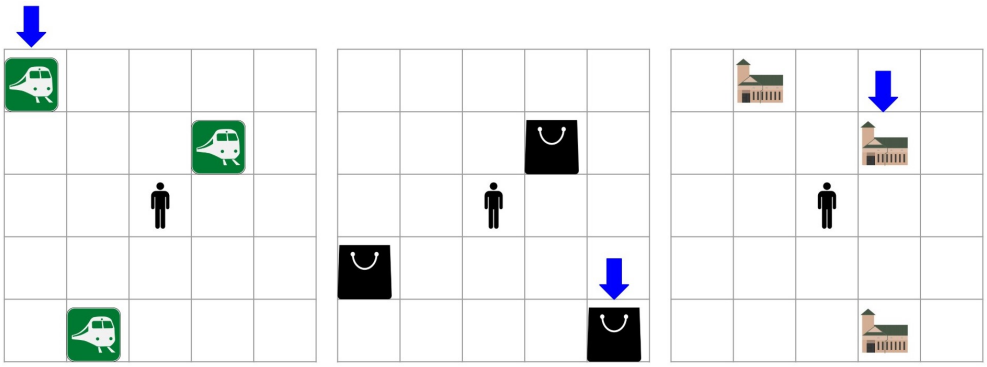


Fig.1.8 Research application

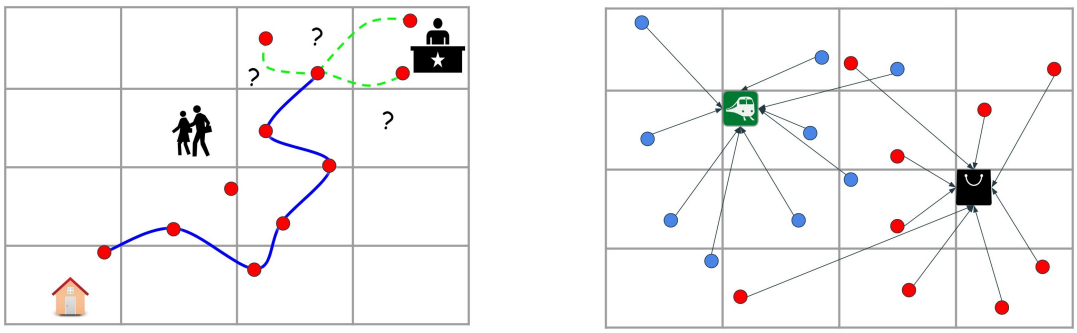
Fig.1.9 Better service

ondly, we turn the latitude and longitude into grid which is a method to represent the location for a specific area, which can reduce the computing burden. In Chapter 3, Fig. 3.1 will show the difference between points and grids, and how we trasfer it. Third, before the training, we will balance the number of positive and negative samples in each epoch of dataset. Then, we train and evaluate the model with our dataset. Eventually, we can predict whether a person will enter the station at a time of next week. And, we also explore the difference between different models with our MetaInfo method through several experiments. Fig. 1.10 demonstrates the research target.

1.5 Framework

Fig. 1.11 shows the framework of my research. In the data-processing procedure, there are 4 steps:

- Interpolate the raw trajectory into trajectory with a 5-minute interval.
- The origin data are points-like data. We transfer the trajectory data into grids to represent



(a) To go where (b) How many will come

Fig.1.10 Draft of research target

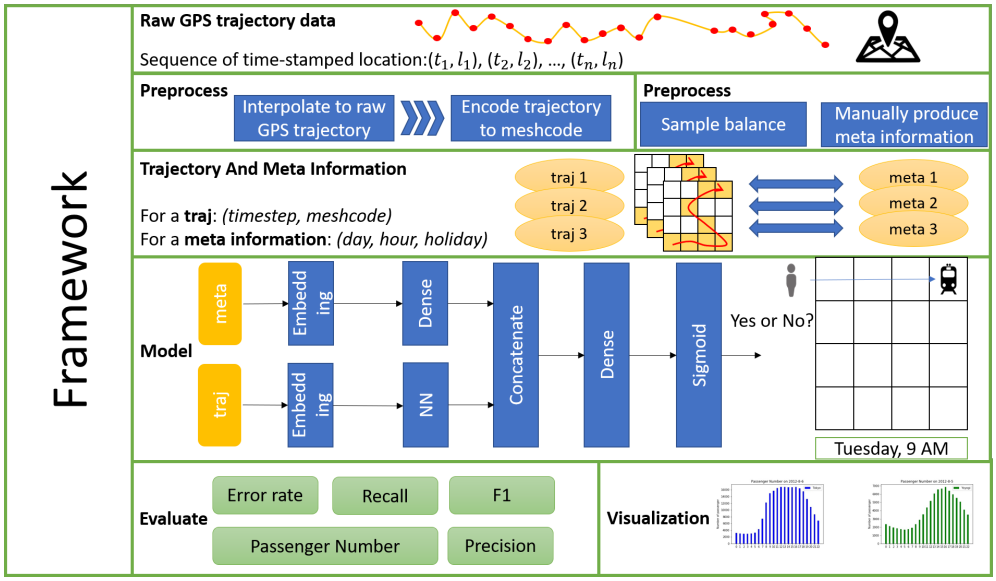


Fig.1.11 Overview of our method

location, because there is no necessity to predict the precise location as latitude and longitude. What we are concerned is the whole situation of the area which in our experiment is 1 km^2.

- Existence labeling: Using trajectory data over a period of time in the future such as 1 month, we add binary labels to represent the existence of a person at a specific time and area.
- Extract the processed trajectory with specific interval to reduce unnecessary computing.

Then, we produce the meta information manually. The meta information can include the multiple information of next week, like datetime information, event information, holiday information and etc. To build up the meta information of a week, we utilize a vector to record the meta information

of next week, e.g. datetime information-(2,8) which means 9 o'clock on Wednesday morning. If we have more information of next week, we can encode the information and add into the vector. After this preparation, we do the oversampling operation on trajectory data and meta data pairs to generate our training dataset.
In the deep learning architecture, we are building up our models based on several neural network structures, testing the performance on long-term trajectory prediction task with those models, and eventually choose the well performance models out and compare with the typical time series statistic model. In this part, we are trying to embed the meta information of a week with historical trajectory of last week to capture the regularity of mobility, aiming to dig out the correlation between historical trajectory and meta information.
Through this method, it is possible to create a binary classifier for a week and even a longer time period.

1.6 Summary

In summary, our work has the following key characteristics that make it unique:

- We are conducting a research using long-time history trajectory to predict the passenger number via binary classifier. Our research target is focusing on how to capture the periodicity pattern through history trajectory.
- We utilize the state-of-the-art techniques with meta information, so that we can automatically learn the sophisticated interactions in spatiotemporal features.
- We evaluate our entire framework on real-world dataset, and demonstrate that it can achieve satisfactory performance on specific area.

The remainder of this paper is organized as follows. In chapter 2, we briefly recap the preliminaries of our research. In chapter 3, we show up our methodology and related explanation. In chapter 4, we present data description, metric explanation, experimental settings, performance evaluation, and give the assumption based on the analysis. In chapter 5, we conduct 2 case study and discuss the problems in case study. In chapter 6 we make our conclusion and discuss possible future work.

第 2 章

Preliminaries

In this section, we formally formulate the mobility prediction problem in order to reduce unnecessary confusion. Then we introduce some related works about long-term trajectory based researches. Finally, we discuss the motivation and overview of our solution.

2.1 Problem Definition

- **Definition 1 (Trajectory)** Given a user identification *uid* and time window t_w, trajectory $traj_{t_w}^{uid}$ is a sequence of spatiotemporal points, i.e., $traj_{t_w}^{uid} = q_1 q_2 ... q_n$. *uid* is an identification registration that can differentiate among users. t_w is a specific period of time, i.e. a day, a week, 2 weeks and etc, i.e., $t_w = \{t_0, t_1, t_2, ..., t_n\}$. A spatiotemporal point q is a tuple of timestamp t and location identification l, i.e., $q = (t, l)$.
- **Definition 2 (Meta information)** Meta information $info$ is a tuple of several kinds of information at a timestamp. The information include day, hour, holiday and etc., i.e., $info_t = (day, hour, holiday)$. In this formula, *day* can represent 7 days in a week by integer numbers(0-6), *hour* denotes 24 hours in a day by integer numbers(0-23), *holiday* explains whether the day is a holiday by integer numbers(0,1)

2.2 Related works

In this section, we will give an introduction of the overview of mobility pattern recognition, and the related works.

Table2.1 Notation Description

Symbol	Description
$traj$	User trajectory sequence in the history
$info$	The information at a timestamp
uid	User ID for each user to represent their identification
t_w	Time window represents a period of time
q	Spatiotemporal point to represent the location at a time
t	The time of a spatiotemporal point
l	The location identification of a spatiotemporal point

2.2.1 General mobility models

When it comes to mobility prediction, researchers usually analyze the human mobility as a population flow or as an individual, aiming at reproducing the individual mobility patterns or general population people flows. In both cases, one necessarily take in to account the characteristic spatial and temporal scales of the mobility process, which is the trajectory and can vary from hundreds of meters to thousands of kilometers and from hours to years. For this reason, each of the cases have been tackled with distinct modeling frameworks[33][34][35][36][37]. Specifically, several studies have pointed out that the individual trajectory usually is possessing a high degree of regularity and predictability, which can be exploited to predict an individual's future position nd to construct realistic models of individual mobility.

To prevent from making the problem too complicated, we just analyze the problem in individual cases, and combine those results to build up a wider view on human mobility, which is the circumstance of a station. Predicting the number of passengers in a station belongs to the topic predicting the next location, which is one of the research branch in human mobility field. Predicting the next location is a well studied problem in human mobility, which comes up with several applications in real-world scenarios, from optimizing the efficiency of electronic dispatching systems to predicting and reducing the traffic jam.

Furthermore, our research focuses on how to make full use of long-term trajectory to predict the long-term future. A lot of researches on the basis of short-term trajectory and short-term future prediction have been conducted and well studied, however, researches about long-term trajectory and long-term future is still limited but with huge potential. Instead of short-term future prediction, we believe it is more promising to five a long-term future prediction because it is more

possible to obtain a more holistic view of future human mobility.
For mobility prediction problem, there are mainly 2 ways to solve it. One idea regards mobility prediction as a regression problem. In general, this idea makes use of the historical trajectory to calculate the discrete coordinate where the passenger is heading to. Another idea thinks the problem as a classification problem. In our research, we will expand the discussion based on classification method.

2.2.2 Multi-classification & binary classification

A typical classification problem can be divided into 2 kinds, multi-classification and binary classification. In a multi-classification case, most people use the model to choose a location, among a set of already known places, as the next location. Like Alberto Rossi's team [4] proposed an RNN to give the prediction, where the goal is to select, among a set of already known locations, the next taxi destination. However, there could be endless potential next locations in a real set of locations because of the diverse intention in human mind, the multi-classification just limited the possibility, where we think there could be something better.
In our idea, we prefer to consider the mobility prediction problem as a binary classification problem. As real-world scenario, it is not always needed to do a precise prediction for human mobility, because it is really hard to give an extremely accurate result based on the chaos situation. So we come up with the idea to simplify the problem, predicting the entering intention of passengers. Through predicting the intention of every passenger, we infer the number of passengers who will enter the station at a time in the future. In stead of focusing on the prediction among multi-choices, we believe it is more intuitive and efficient to give the prediction on less choice, like yes or no.

2.2.3 Researches on long-term trajectory

- DeepMove[5]
  DeepMove thinks predicting mobility is not trivial because of three challenges: 1) the complex sequential transition regularities exhibited with time-dependent and high order nature; 2) the multi-level periodicity of human mobility; and 3) the heterogeneity and sparsity of the collected trajectory data. There proposal combines two regularities in a principled way: heterogeneous transition regularity and multi-level periodicity, and predicts the soon future location.
- Spatialtemporal recurrent neural network (ST-RNN)[6]
  Spatial and temporal contextual information plays a key role for analyzing user behav-

iors, and is helpful for predicting where he or she will go next. With the growing ability of collecting information, more and more temporal and spatial contextual information is collected in systems, and the location prediction problem becomes crucial and feasible. ST-RNN can model local temporal and spatial contexts in each layer with time-specific transition matrices for different time intervals and distance-specific transition matrices for different geographical distances.

- Predicting Human Mobility via Variational Attention[7]
  Qiang Gao's team proposed a VANext (Variational Attention based Next) to conduct the prediction task. VANext is a latent variable model for inferring user's next footprint, with historical mobility attention. The variational encoding captures latent features of recent mobility, followed by searching the similar historical trajectories for periodical patterns.

2.3 Overview

As a powerful sequence model structure, the recurrent neural network can capture sequence information. However, when the sequence is too long, i.e. a long sentence with more than 20 words, it's performance will go worse rapidly[1]. In our mobility dataset, the length of one-day trajectory has 24 timestamps or 48 timestamps, where we extract the spatiotemporal point in every 30 minutes or 1 hour, which obviously exceeds the ability of recurrent neural network. And the longer the trajectory is, the worse performance the prediction gives. Thus, we can only use recurrent neural network to deal with limited length of trajectory with a short duration of one day or even shorter. Directly applying recurrent neural network to handle the mobility prediction is intuitive but inefficient. To enhance the efficiency, we propose the method of adding meta information into the models.

第 3 章

Methodologies

3.1 Workflow

We already give a illustration of our research framework, so we give a summary of the workflow in the framework. The whole workflow includes five steps:

1. Data preprocess
   （a）Transfer trajectory data from (latitude, longitude) to meshcode.
   By using the meshcode representation which is a list of integer numbers from 0 to 5000, we believe the computation need can be reduced.
   （b）Achieve sample balance
   After checking the data distribution of data, we found the dataset is extremely imbalanced.
2. Model creation
   There are different choices of machine learning models for classification problem. Each model comes from a different algorithm approach and will perform differently under different datasets. In this study, we utilize different machine learning models combining with meta information to predict whether a person will be in the station at a time in the future.
3. Model training
   The machine learning algorithm learns the mobility pattern from training dataset, and it outputs a trained model that captures the time series relation of human mobility. In our experiment, depending on propagation and gradient decent, the model can be generated by training, which own the capability to identify the mobility pattern, more precisely, the periodic mobility pattern.
4. Validation

In order to avoid over-fitting and other problems, when any internal parameter needs to be adjusted, it is necessary to have a validation dataset in addition to the training and testing datasets. If good results can be generated in validation, the corresponding model can be used in practical, otherwise, the model need to be retrained. Here, to evaluate the reliability of our model, we run the model with validation dataset after training in each epoch.

5. Testing

   Utilizing generated model, the passenger number of the station can be predicted by inputting the historical trajectory.

3.2 Mesh code

Originally, the data is sparse because of data loss or some kind of data transformation problem making the data not complete, which has an adverse impact for our analysis. To deal with sparse data problem, we process the GPS trajectory with a 5-minute interval by adding artificial spatiotemporal points with using existed interpolation algorithm to complement the vacant data, so there are totally 288 positions in a day. The GPS data after interpolation is a sequence of (latitude, longitude) spatiotemporal points. The data is representing as a tuple of 'float' format data, which can occupy a huge amount of space on computer and cost lots of computation resource. In order to reduce the computation burden from large dataset, we utilize meshcode to represent the location of a passenger. Mesh code is a location representation method. According to certain rules, a mesh is what divides the ground surface into a number of squares, with the resolution 100m. For each mesh, the rules mark it with a integer number to express a specific area. However, even though we can release some computation burden, some of the features that latitude and longitude has are abandoned. Fig. 3.1 shows the transformation from point data to mesh code data.

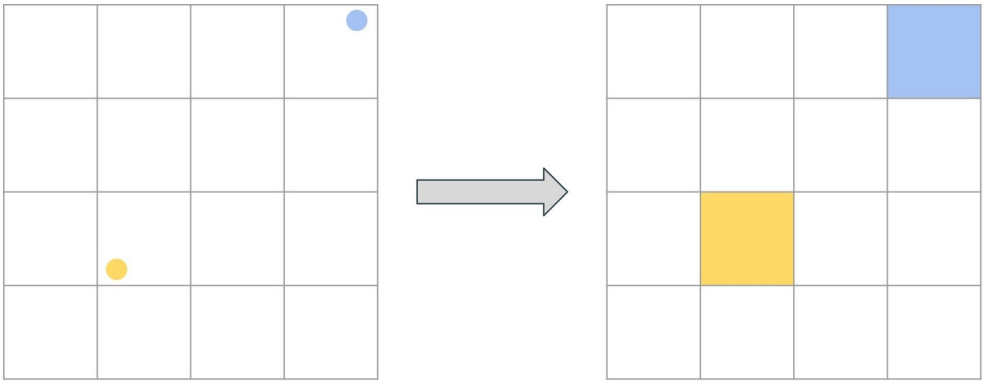


Fig.3.1 Point to mesh transformation

3.3 MetaInfo

For long-term trajectory prediction, one of the obstacle is the long-term input which the model cannot extract the spatiotemporal features out well. Most of the previous researches have made bunches of contribution to capture the internal correlation in the trajectory with a lot of mathematical methods. However, seldom of them tried to combine the meta information (MetaInfo) to help the model understand the trajectory better. It is possible that the models performance can be enhanced by learning other related data[38]. The meta information can support the model with diversity of information, e.g., weather, date time and etc. By providing these kind of information, we believe that the model can build up the correlation between trajectory and meta information, and furthermore learn the internal correlation of the long-term trajectory better.

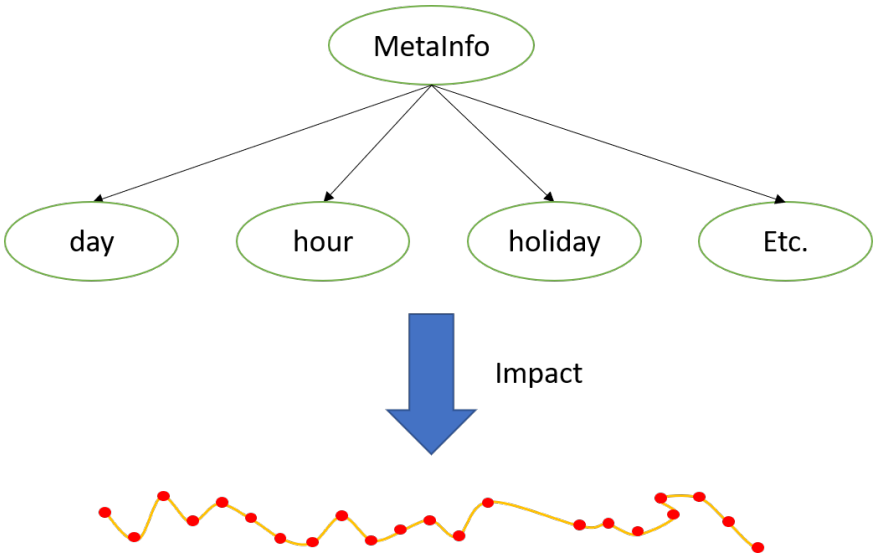


Fig.3.2 Idea of combining meta information with trajectory data

In the meta information, we are trying to use the potential factors that are not directly related to trajectory. For example, the day information, the time information and the holiday information. To utilize the factors, we encode these multiple information as a vector of integers, which is the format that can be understood by the model. For instance, $MetaInfo = (day, hour, holiday)$ Furthermore, if we can find out more kinds of reliable meta information, it is possible to assemble them together as the model input.

3.4 Sample Balance

After checking the data distribution of a week, we found that the dataset is extremely imbalanced. Fig. 3.3 shows the ratio of between positive and negative samples in a week. This dataset is

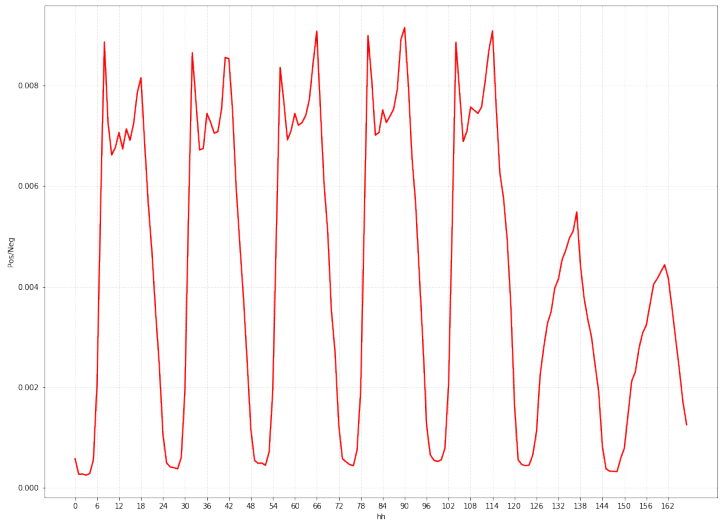


Fig.3.3 Visualization: the ratio of positive and negative samples

covering from 2012-07-02 to 2012-07-09, which means the length of the trajectory is 168 timestamps. From the figure, we can easily tell that for the most time of a week, the ratio is fluctuating around 0.004. In this situation, the minority class is positive samples and the majority class is negative samples. If the model is trained under this adverse condition, the model will be biased in favor of negative prediction. As a consequence of this problem, the model will carry out a prediction where many positive samples turn to negative, which apparently is wrong. In order to prevent this circumstance, there is need to balance the number of positive and negative samples. This is not we want for a classification problem. Generally, in order to make sure the model can collect and extract the right features of both positive and negative samples, we ensure the numbers of both positive and negative samples are the same.

There are several algorithms to proceed the sample balance[2]. Here, we list the well-known approaches and give an explanation on them. They are random undersampling, Near-Miss undersampling, Tomek links undersampling and etc. At last, we explain why we use random undersampling method in stead of other undersampling method.

3.4.1 k-nearest neighbors algorithm

Since k-nearest neighbors algorithm, also known as k-NN, is widely used in the undersampling algorithms below, we decide to give an introduction first. Here k means the number of classes, e.g., in our case, we only have two classes, so k should be 2. In pattern recognition, k-NN is a non-parametric method proposed by Thomas Cover used for classification and regression[9]. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

1. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). Here we have Fig. 3.4 to indicate the usage of k-NN on classification problem. The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. In our case, we will get a set of each class membership.
2. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

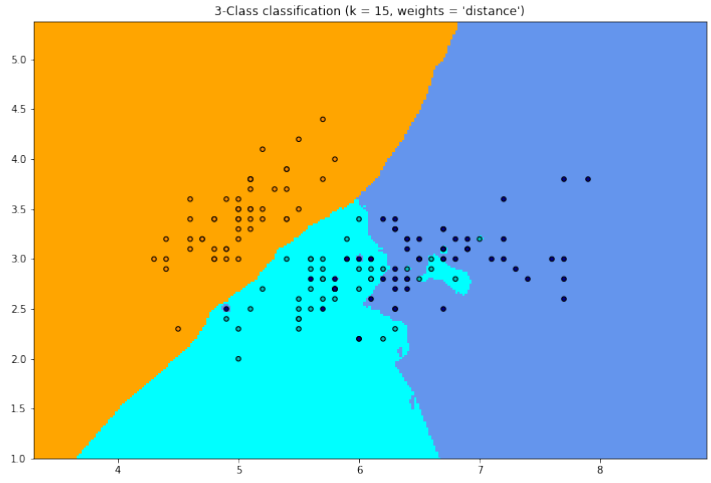


Fig.3.4 The typical usage of k-NN classification algorithm.

Before we introduce the algorithms in the following, we give a raw dataset illustration Fig. 3.5 to

explain the performance of the undersampling algorithms.

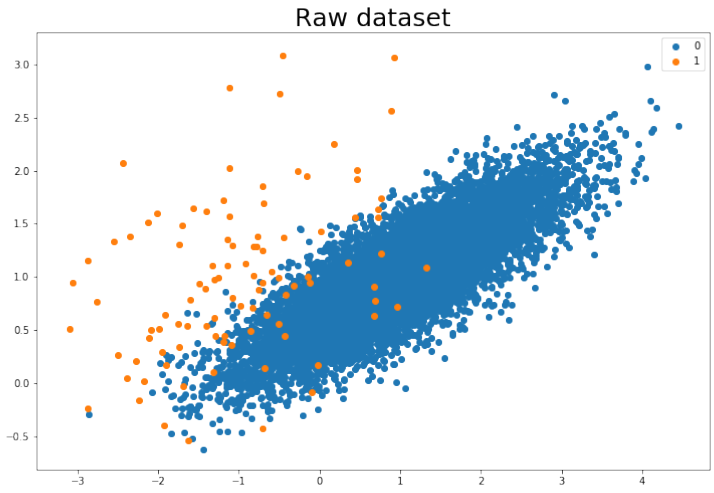


Fig.3.5 A raw dataset illustration for explaining undersampling algorithms

3.4.2 Algorithms that Select Examples to Keep

1. Random undersampling

   Random undersampling is the simplest and the most brutal approach, which keeps all the minority class and then randomly selects the same number of examples from the majority class to produce a new balanced dataset. Random undersampling costs little memory when the computer is processing the dataset, however, the distribution of the new dataset is rarely reasonable. This method kinds of relies on the fortunate, there too many data points in the majority class that it is easily to select those data points which are isolated from the minority class, and can be easily classified from the the minority class either.

2. NearMiss undersampling[10]

   NearMiss undersampling is a method based on k-nearest neighbors algorithms to select the samples from majority class. NearMiss refers to a collection of undersampling methods that select examples based on the distance of majority class examples to minority class examples. There are three versions of NearMiss algorithm, named NearMiss-1, NearMiss-2, and NearMiss-3. NearMiss-1 selects examples from the majority class that have the smallest average distance to the three closest examples from the minority class. NearMiss-2 selects examples from the majority class that have the smallest average distance to the three furthest examples from the minority class. NearMiss-3 involves selecting a given

number of majority class examples for each example in the minority class that are closest. In the algorithm, the distance is determined in feature space using Euclidean distance or similar. We have Fig. 3.6, three versions of NearMiss.

（a）NearMiss-1: Majority class examples with minimum average distance to three closest minority class examples.

（b）NearMiss-2: Majority class examples with minimum average distance to three furthest minority class examples.

（c）NearMiss-3: Majority class examples with minimum distance to each minority class example.

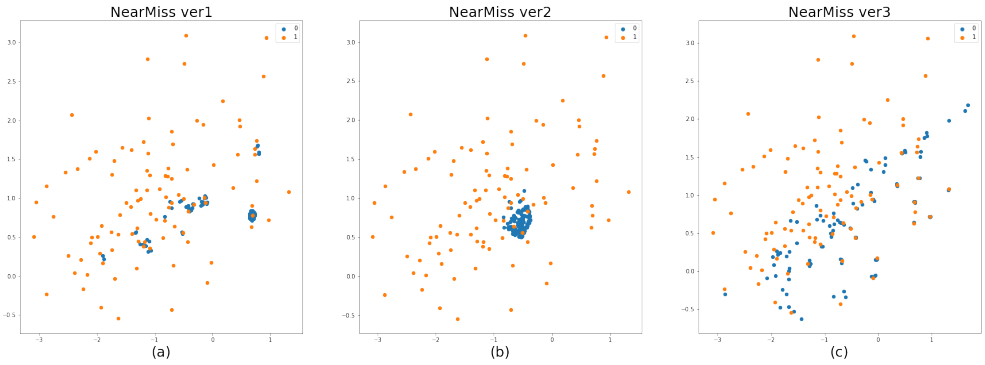


Fig.3.6 The 3 versions of NearMiss algorithm. (a)NearMiss ver1. (b)NearMiss ver2. (c)NearMiss ver3

3. Condensed Nearest Neighbor Rule for undersampling

Condensed Nearest Neighbors is an undersampling technique that seeks a subset of a collection of samples that results in no loss in model performance, referred to as a minimal consistent set. It is achieved by enumerating the examples in the dataset and adding them to the “store” only if they cannot be classified correctly by the current contents of the store. During the procedure, the k-NN algorithm is used to classify points to determine if they are to be added to the store or not. When used for imbalanced classification, the store is comprised of all examples in the minority set and only examples from the majority set that cannot be classified correctly are added incrementally to the store. Fig. 3.7 is the visualization of Condensed Nearest Neighbor Rule.

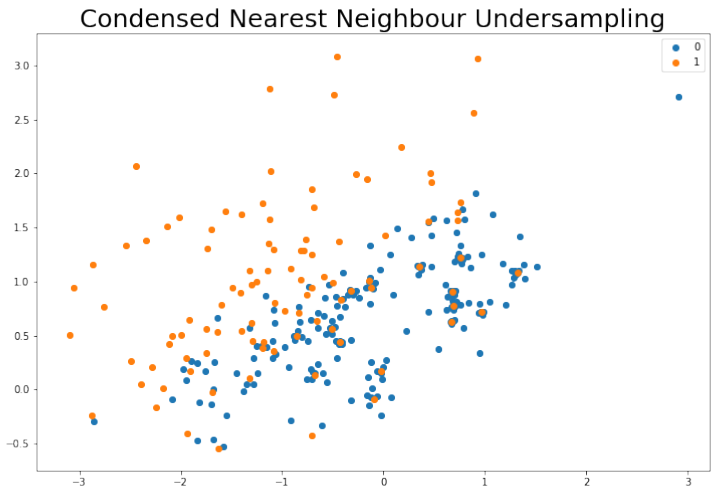


Fig.3.7 Condensed Nearest Neighbor Rule undersampling demonstration

3.4.3 Methods that Select Examples to Delete

In this section, we will take a closer look at methods that select examples from the majority class to delete, including the popular Tomek Links method and the Edited Nearest Neighbors rule.

1. Tomek Links undersampling

   A criticism of the Condensed Nearest Neighbor Rule is that examples are selected randomly, especially initially. This has the effect of allowing redundant examples into the store and in allowing examples that are internal to the mass of the distribution, rather than on the class boundary, into the store. Two modifications to the CNN procedure were proposed by Ivan Tomek in his 1976 paper titled “Two modifications of CNN” [11]. One of the modifications is a rule that finds pairs of examples, one from each class. They together have the smallest Euclidean distance to each other in feature space. This means that in a binary classification problem with classes 0 and 1, a pair would have an example from each class and would be closest neighbors across the dataset. These cross-class pairs are now generally referred to as “Tomek Links” and are valuable as they define the class boundary. The procedure for finding Tomek Links can be used to locate all cross-class nearest neighbors. If the examples in the minority class are held constant, the procedure can be used to find all of those examples in the majority class that are closest to the minority class, then removed. These would be the ambiguous examples.

From Fig. 3.8, we cannot find any obvious difference in the scatter plot of the transformed dataset after running the Tomek Link algorithm. This highlights that although finding the ambiguous examples on the class boundary is useful, alone, it is not a great undersampling technique. In practice, the Tomek Links procedure is often combined with other methods, such as the Condensed Nearest Neighbor Rule.

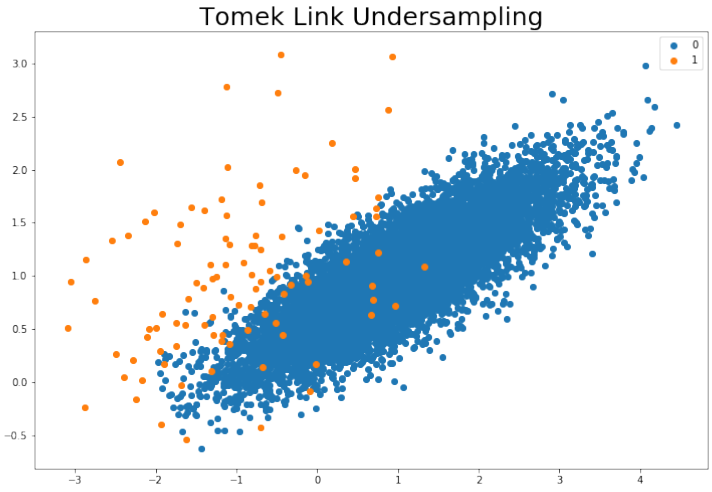


Fig.3.8 Tomek Link undersampling demonstration

2. Edited Nearest Neighbors Rule for undersampling
   Another rule for finding ambiguous and noisy examples in a dataset is called Edited Nearest Neighbors. This rule involves using k=3 nearest neighbors to locate those examples in a dataset that are misclassified and that are then removed before a k=1 classification rule is applied. This approach of resampling and classification was proposed by Dennis Wilson[12]. When used as an undersampling procedure, the rule can be applied to each example in the majority class, allowing those examples that are misclassified as belonging to the minority class to be removed, and those correctly classified to remain.

3.4.4 Combinations of Keep and Delete Methods

In this section, we will take a closer look at techniques that combine the techniques we have already looked at to both keep and delete examples from the majority class, such as One-Sided Selection and the Neighborhood Cleaning Rule.

1. One-Sided Selection for Undersampling

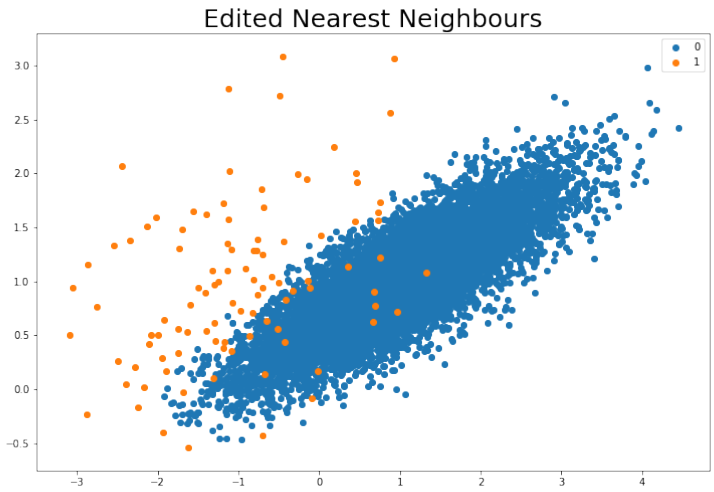


Fig.3.9 Edited Nearest Neighbors Rule for undersampling demonstration

One-Sided Selection is an undersampling technique that combines Tomek Links and the Condensed Nearest Neighbor Rule. This combination of methods was proposed by Miroslav Kubat and Stan Matwin[13]. Specifically, Tomek Links are ambiguous points on the class boundary and are identified and removed in the majority class. The CNN method is then used to remove redundant examples from the majority class that are far from the decision boundary. The CNN procedure occurs in one-step and involves first adding all minority class examples to the store and some number of majority class examples (e.g. 1), then classifying all remaining majority class examples with k-NN (k=1) and adding those that are misclassified to the store.

2. Neighborhood Cleaning Rule for Undersampling
   The Neighborhood Cleaning Rule is an undersampling technique that combines both the Condensed Nearest Neighbor (CNN) Rule to remove redundant examples and the Edited Nearest Neighbors Rule to remove noisy or ambiguous examples. Like One-Sided Selection, the Neighborhood Cleaning Rule is applied in a one-step manner, then the examples, which are misclassified, are removed according to a k-NN algorithm, by following the Edited Nearest Neighbors Rule. Unlike One-Sided Selection, less of the redundant examples are removed and more attention is placed on cleaning those examples that are reserved.

Even though the algorithms above are powerful and useful, there is a main obstacle before applying them, the computation cost. Our dataset is too large, we totally have over 15,120,000

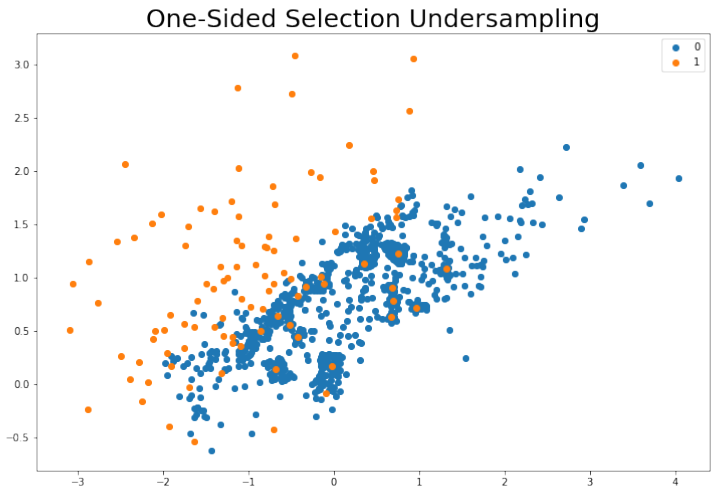


Fig.3.10 One-side Selection undersampling demonstration

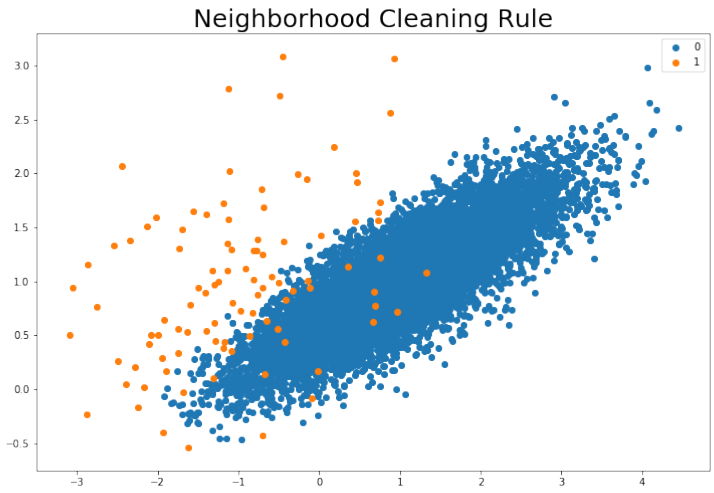


Fig.3.11 Neighborhood Cleaning Rule undersampling demonstration

trajectories and each trajectory crosses 168 timestamps, which can easily exceed the computation ability of our computer. Thus, we choose the random undersampling method to resample the dataset for training, and we will use the whole dataset when doing the case study in Experiment Chapter. In order to complement the weakness of random undersampling method, we will train the model for several times so that the model can learn more features of the negative samples.

3.5 Statistic Model

3.5.1 ARIMA

Box-Jenkings' ARIMA[16] stands for AutoRegressive Integrated Moving Average. ARIMA model is a class of statistical models for analyzing and forecasting time series data, by analyzing and capturing the stationary pattern of the data. However, ARIMA has the following weakness either: 1) It requires data or the data after differencing is stationary, which means the data shall has trend or periodicity. 2) It can capture the linear but cannot capture non-linear relation among the data. Technically, ARIMA can be spilted into 3 terms: $AR(p)$ stands for the AutoRegressive model, and the p parameter is an integer that confirms how many lagged(previous) series are going to be used to forecast periods ahead. And AutoRegressive indicates that the evolving variable of interest is regressed on its own lagged values. ARP represents as the following equation:

$$Y_t = \epsilon_t + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_p T_{t-p} \tag{3.1}$$

$$\tag{3.2}$$

$I(d)$ stands for the differencing part, and d parameter tells how many differencing-order is going to be used to obtain stationary series of data. $MA(q)$ stands for Moving Average model, and the q parameter states the number of lagged forecasting error terms in the prediction equation. Although the method can handle data with a trend, it does not support time series with a seasonal component.

3.5.2 SARIMA

To tackle the seasonal component left above, [31] comes up with SARIMA model. SARIMA stands for seasonal ARIMA, which is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component. The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but they involve backshifts of the seasonal period. It adds three new hyper-parameters to specify the AutoRegressive (AR), differencing (I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality. For SARIMA model, we just simply express it as the formula: $ARIMA(p, d, q)x(P, D, Q)$, where ARIMA part keeps same but added with x(P,D,Q), in which P is the number of seasonal autoregressive terms, D is the number of seasonal difference, and Q is the number of seasonal moving average terms.

3.6 Machine learning methods

In this section, we will introduce the widely used and well-performance machine learning methods: Hidden Markov Model, Recuerrent Neural Network and its variations, and Convolutional Neural Network.

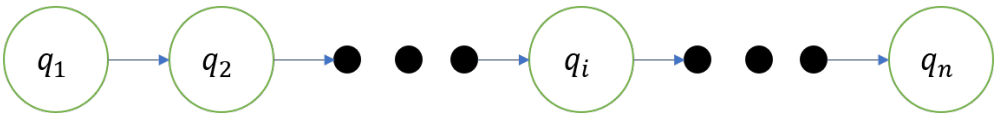
3.6.1 Hidden Markov model

The Hidden Markov Model (HMM) is a stochastic processes,which is the augment version of Markov chain. A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on value from some set. These sets can be words, tags, or symbols representing anything, in our case the sets are the trajectory. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state. It is as if we are predicting the next location of a passenger, we could examine current location, but we weren't allow to look at last location.
More formally, consider a sequence of spatiotemporal points $q_1, q_2, ..., q_i$, Markov chain embodies the **Markov Assumption** on the probabilities of the sequence: when predicting the future, the past doesn't matter, only the present, which is the Markov Assumption:

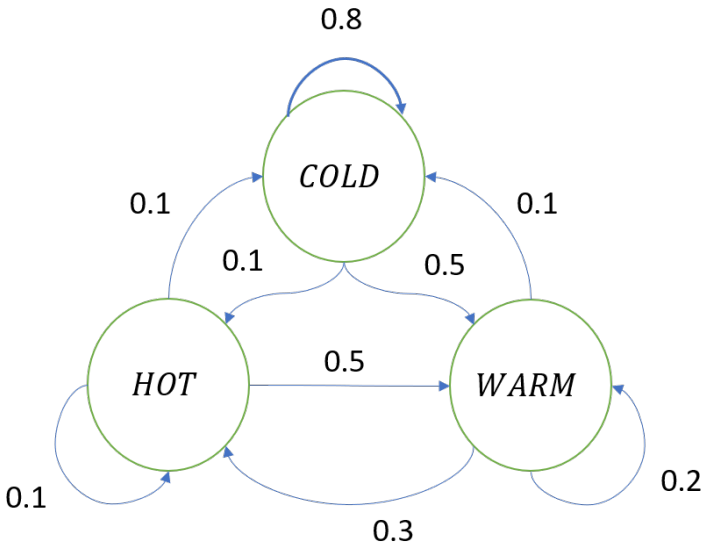
$$P(q_i = a|q_1..q_{i-1}) = P(q_i = a|q_{i-1}) \tag{3.3}$$

A Markov chain is useful when we need to compute a probability for a sequence of observable events. However, in many cases, it is not enough to consider only one previous state. Some influential events are hidden, in other words we don't observe them directly. For instance, we don't observe all the spatiotemporal points q of a trajectory. Rather, we only see the points that are extracted by time interval. So we call potentially influential spatiotemporal points hidden because they are not observed.
A hidden Markov model (HMM) allows us to consider both observed spatiotemporal points and hidden spatiotemporal points that we think of as causal factors in our probabilistic model. Except the Markov assumption, Hidden Markov model is following another assumption: the probability of an output observation o_i depends only on the state q_i that produced the observation and not on any other states or any other observations: **Output Independence**:

$$P(o_i = a|q_1...q_i...o_1...o_i...o_T)) = P(o_i|q_i) \tag{3.4}$$



(a) Markov chain abstract



(b) Markov chain for weather

Fig.3.12 Markov chain model

Thus, we assume the long-term trajectory before the targeted time are hidden, and apply HMM to predict the observable state, whether the passenger will enter the station.

3.6.2 LightGBM

Before introducing the widely used machine learning method, there is need to mention Gradient Boosting Decision Tree(GBDT). Gradient Boosting Decision Tree [15] is a widely used machine learning algorithm, due to its efficiency and accuracy, and interpret ability. Gradient boosting technique provides a way to ensemble or combines multiple weak prediction models, typically decision tree, and grows up to a more powerful prediction model to handle regression or classification problems. However, with the emergence of big data, GBDT is facing new challenges, especially in the tradeoff between accuracy and efficiency. Conventional implementations of GBDT need to, for every feature, scan all the data instances to estimate the information gain of all the possible split points. Therefore, their computational complexities will be proportional to both the number of features and the number of instances. This makes these implementations very time consuming when handling big data.

Consequently, Guolin Ke's team[14] propose LighGBM with utilizing two novel techniques to tackle this challenge. During training, LightGBM calculates the gradients of each sample and drop those low gradient data instances so that less instances are calculated. And LightGBM reduce the number of features by bundling those exclusive sparse features in feature space as a new feature. In this way, LightGBM achieve a better performance than conventional GBDT. In recent years, due to its fastness and accuracy, LightGBM is widely in various research fields[26][27] and industry fields[28][29] especially those challenging sequential problems[23][24][25].
To simply clarify the mechanism of LightGBM, we have typical decision tree algorithm which is the basis of both GBDT and LightGBM. From our point of view, the only difference between decision tree and them is the accelerating mechanism and tree structure update method. In Fig. 3.13, we use the typical decision tree to decide whether a person is having good credit risk or bad credit risk. By evaluating the features in each step, decision tree calculates the risk and update the weight of each feature by comparing the ground truth and the calculation result.

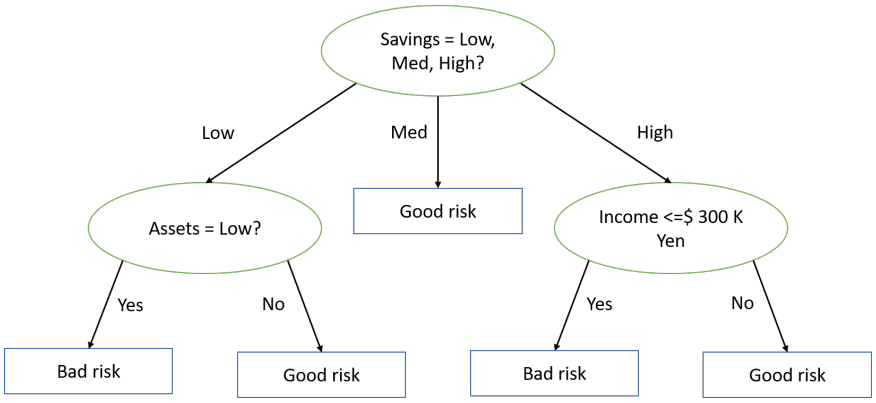


Fig.3.13 Typical decision tree in credit evaluation

3.7 Recurrent neural network

As a powerful sequence model structure, the Recurrent Neural Network (RNN) is widely used in sequential problems, e.g., natural language processing(NLP) tasks. The main contribution of RNN is that it can capture the sequential features in the data. When we are trying to understand a sequence problem, there is always a need to consider the previous states, in stead of only analyzing the current state. For instance, after office time when a person was in the office, the person is heading to the station to take the railway now. Then, what is after the station? If we don't know the person was at the office before only knowing that he is at the station, we will have 2 inferences going to office or going home. Thanks to the office precondition, we can definitely infer that the

person will go home after staying at station. Therefore, there is definitely a relationship between the next location and the previous locations. A typical RNN structure is shown in Fig. 3.14.

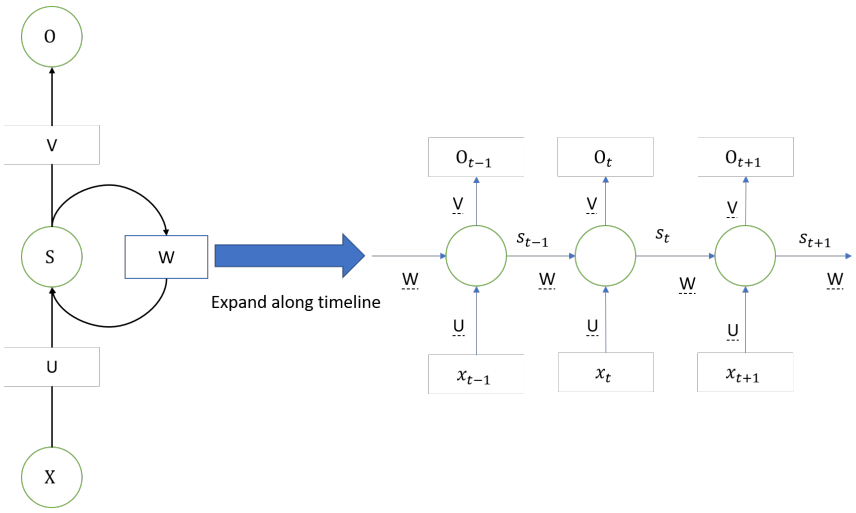


Fig.3.14 RNN structure

Given a passenger's historical trajectory $traj = q_1 q_2 ... q_t$, we have a set of spatiotemporal points q as inputs. RNN has the ability to produce a function to create a function below:

$$O_t = g(V \cdot S_t) \tag{3.5}$$

$$S_t = f(U \cdot X_t + W \cdot S_{t-1}) \tag{3.6}$$

to build up the relationship among those spatiotemporal points to infer the next location. In the equation (3.5), O_t is the output of RNN, which is the next location in our case. V is the weight matrix between hidden layer to output layer. U is the weight matrix between input layer and hidden layer. W is the weight matrix between last moment's hidden layer and current hidden layer. Now we can easily find how RNN capture the sequential information. At moment t, the output value is o_t, the hidden value is s_t, and the input value is q_t. Hidden value s_t is not only decided by q_t but also last hidden value s_{t-1}

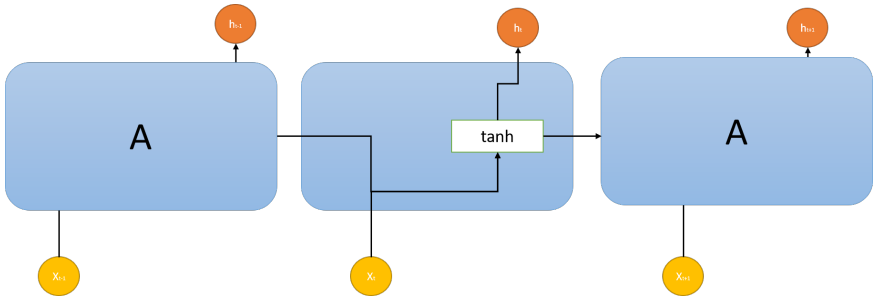
However, when the sequence is too long, i.e., in NLP field a long sentence with more than 20 words, its performance will go worse rapidly[1], which is directly caused by vanishing gradient problem. When we are optimizing the weight matrices, we calculate the gradient of the to update the values in the matrices. Unfortunately, because of the long length of sequence, it is relatively easy to get a extremely small value of gradient resulting the vanishing gradient problem. Thus, we can only use recurrent neural network to deal with limited length of trajectory with a short duration of one day or even shorter.

In our mobility dataset, the length of one-day trajectory has 24 timestamps, where we at least

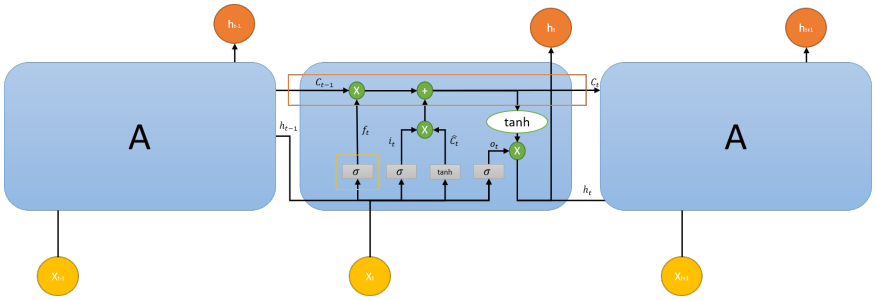
extract the spatiotemporal points in every 1 hour in a week, which obviously exceeds the ability of recurrent neural network. And the longer the trajectory is, the worse performance the prediction gives. In our experiment, we will compare RNN with other neural network structures to testify this problem.

3.7.1 LSTM

In order to solve the short-term memory problem of RNN, Sepp and Jurgen[17] designed Long-Short-Term Memory (LSTM) structure with a gate mechanism. To make the comparison of typical RNN and LSTM, we have Fig. 3.15 for demonstration. Even though RNN is a huge neural



(a) RNN unit structure



(b) LSTM unit structure

Fig.3.15 The unit structure of RNN and LSTM

network, if we look at the inside of RNN units, it is a quite simple structure, only consisting of one layer of activation function, here we denotes as a hyperbolic function *tanh*. LSTM consists of several units either but with a relatively complex structure. The horizon line in the top orange square is called cell state, which is a path sending the information from last unit to the next. To

explain how LSTM works, we first list out the equations, and talk about them one by one later.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.7}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.8}$$
$$\widehat{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \tag{3.9}$$
$$C_t = f_t * C_{t-1} + i_t * \widehat{C}_t \tag{3.10}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.11}$$
$$h_t = o_t * tanh(C_t) \tag{3.12}$$

And the most significant part is the gate structure in the yellow square, which can decide how much information from the last unit is considered and how much new information is considered in this unit with the equation (3.10). After the gate, LSTM inputs the information to the activation function *tanh* to update current unit with $\widehat{C}_t$ by equation (3.7), (3.8), (3.9). At last, LSTM decide how much information is going to be output as h_t by the equation (3.11) and (3.12) The explanation above is the standard process of LSTM in each step, which improve the performance of typical RNN.

3.7.2 GRU

Aiming to capture the strong relation when facing a relatively longer sequence, Kyunghyun Cho's team proposed another gate structure called Gated Recurrent Unit (GRU)[18]. In GRU

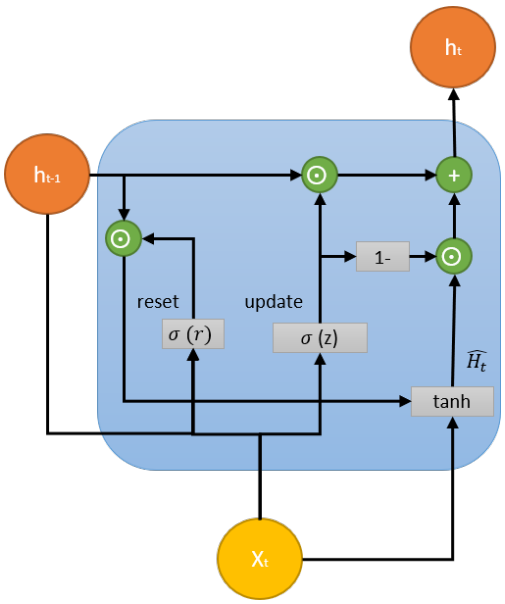


Fig.3.16 The unit structure of GRU

unit, the most significant components are reset gate r and update gate z. Reset gate is used from

the model to decide how much of the past information to forget, and update gate helps the model to determine how much of the past information needs to or is worthy to be passed along to the future. Like the clarification of LSTM, we list the related equations first and explain their roles in GRU unit.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) + b_r \tag{3.13}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) + b_z \tag{3.14}$$

$$\widehat{H}_t = tanh(W \cdot [h_{t-1} \odot r_t, x_t]) + b \tag{3.15}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widehat{H}_t \tag{3.16}$$

First of all, GRU unit calculates two gate states r_t and z_t by equation (3.15) and (3.16) to store and drop the past information from the last unit. The smartest part of GRU is that GRU only use one gate structure to determine how to deal with the past information simultaneously by $(1 - z) \odot h_{t-1}$. Here we can regard z_t as a 'forget' gate to ignore the unimportant messages from the past. And in $z_t \odot \widehat{H}_t$, GRU decide to store how much or what information from the past and add the information from current input. Eventually, combine the operation of $(1 - z_t) \odot h_{t-1} + z_t \odot \widehat{H}_t$, GRU updates the current content (state).

3.8 Convolutional neural network

Convolutional Neural Network(CNN) proposed by Yann LeCun et al.[19] is one of the typical model of deep learning for every textbook, which is mainly used to challenge and solve the tasks related to images, e.g., object detection, image segmentation, image classification and etc. And recently, more researches make use of CNN to tackle NLP[21] and some sequential problems. In multi-layer convolutional neural network, CNN has the ability to learn complex nonlinear mappings form a high to a low dimension feature space, where image classification is obvious[20]. Importantly, the simplified features in low dimension feature space can build up the abstract features in high dimension feature space with holding those evidential information. By using this significant feature of CNN, we assume it can capture the abstract features in long-term historical trajectories, where we believe there exists specific mobility pattern in daily life.

3.8.1 GLU

Skipping those complex CNN structures, we tend to utilize the fastest structure Gated linear Unit(GLU) proposed by Yann N. Dauphin et al.[21] firstly to tackle NLP task. GLU is basically a variation of basic CNN with a gate mechanism, consisting of two different Convolutional kernels

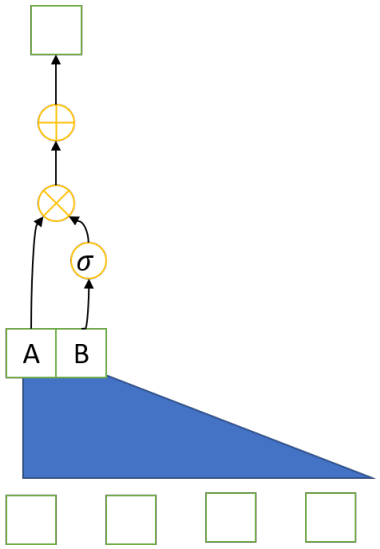


Fig.3.17 The unit structure of GLU

and a gate mechanism. Firstly, those two convolutional blocks extract different dimension features from feature spaces. The two outputs of the convolutional block A and B are then passed to the gate block which involves element-wise multiplying A by sigmoid(B): $A \otimes \sigma(B)$, equivalently:

$$h(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + b) \otimes (\mathbf{X} * \mathbf{V} + c) \tag{3.17}$$

where X is the input, W and V represents two different convolutional kernels which takes the responsibility to extract the features from the inputs. b and c indicates bias parameters which can be ignored, and σ works as gate mechanism.

第 4 章

Experiment

In this chapter, we first present the preliminaries to explain experiment settings, including dataset and the metrics we highly value. Then, in order to explore the performance in a long-term prediction, we deliver the experiment results of the baselines and our MetaInfo method to show the model variants. At last, we conduct several case studies to verify the performance in real situation. The results in this chapter are going to be used to answer the following four research questions:

- **RQ1**: Does our MetaInfo-based method improve the prediction results?
- **RQ2**: How do the hyper-parameters, e.g. length of trajectory, have impact on the model performance?
- **RQ3**: How is the performance of our MetaInfo-based method in real case comparing with other statistic methods?
- **RQ4**: Is the method able to be applied into real case?

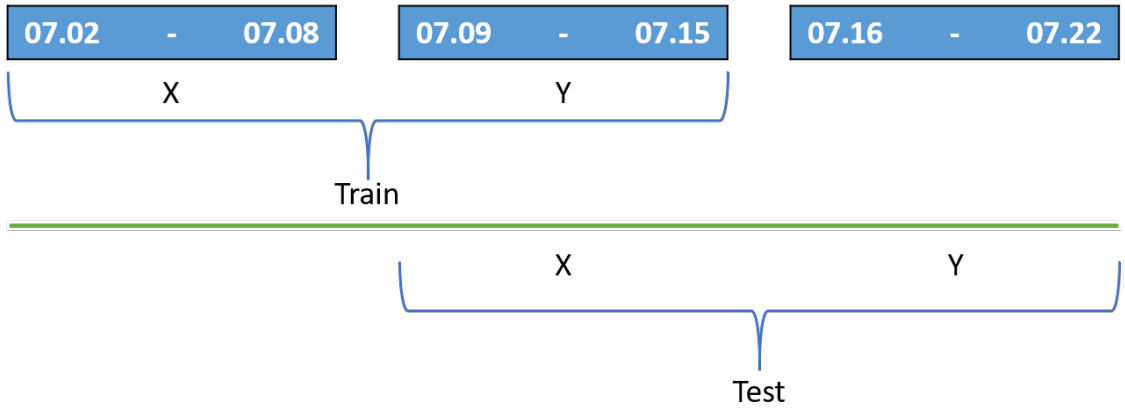
4.1 Preliminaries

In this section, we will give a description of data, experimental settings, and evaluation metrics.

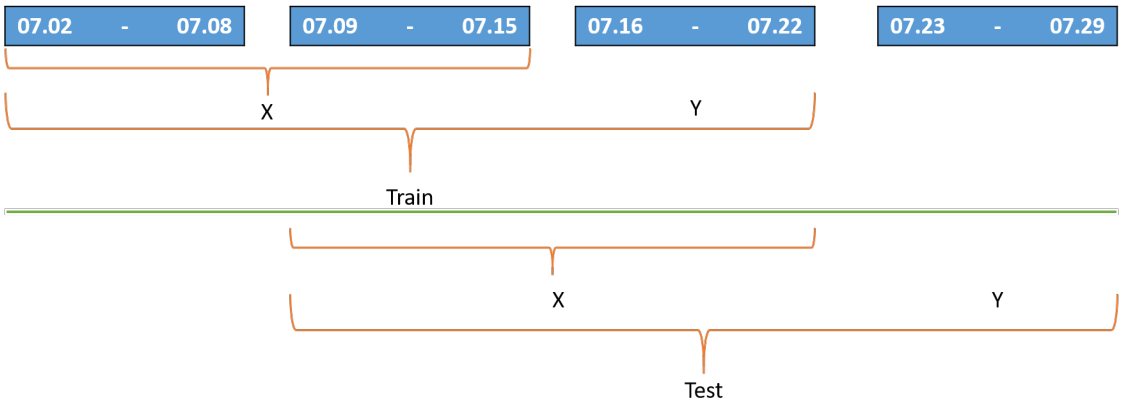
4.1.1 Dataset

First of all, we need filter the user id *uid* who has a 2-week-length trajectory for pattern analysis. Secondly, we assume that the mobility pattern usually won't change a lot, so the closed weeks have stronger similarity and connection than those that are not closed to.
For example, there are three sets of dataset: Dataset-1's time range is from July 2^{nd} to July 8^{th}. Dataset-2's time range is from July 9^{nd} to July 15^{th}. Dataset-3's time range is from July 16^{nd} to

July 22^{th}. Among these three datasets, Dataset-2 has strong similarity with both Dataset-1 and Dataset-3 but dataset Dataset-1 has weaker similarity with Dataset-3. That is to say, when doing the experiment, we need and must consider the similarity and connection of train dataset and test dataset. Otherwise, the prediction will be far way from the reality and be meaningless. Thus, in case study when we are doing the training and testing we use slide window method in Fig. 4.1 to select the datasets.



(a) Slide window selection on 1 week



(b) Slide window selection on 2 weeks

Fig.4.1 Slide window selection on different length of data

For better understanding of the datasets, we list the detail of the datasets in Table 4.1, and the calendar of our experiment periods in Fig. 4.2. In addition, these datasets are the raw datasets, we didn't do any sample balance to reduce the number of samples but filter the 2-week-constant *uid* out of the origin 200,000 *uid*. When we are training the model, we will conduct the random undersampling process to ensure the sample balance during training. Additionally, we won't use all of the datasets to train the model, some of the datasets are used to test the models, confirming

Table4.1 Dataset description

Training Length	Dataset	Time range	Number of trajectory
1 week	Dataset *A*	2012.07.02-2020.07.15	128643
	Dataset *B*	2012.07.09-2020.07.22	121058
	Dataset *C*	2012.07.16-2020.07.29	114403
	Dataset *D*	2012.07.30-2020.08.12	101151
	Dataset *E*	2012.08.06-2020.08.19	93848
	Dataset *F*	2012.08.13-2020.08.26	92926
2 weeks	Dataset *G*	2012.07.02-2020.07.22	101212
	Dataset *H*	2012.07.09-2020.07.29	99021
	Dataset *I*	2012.07.23-2020.08.12	82772
	Dataset *J*	2012.07.30-2020.08.19	71377
	Dataset *K*	2012.08.06-2020.08.26	75482

Mon.	Tue.	Wed.	Thur.	Fri.	Sat.	Sun.
07-02	07-03	07-04	07-05	07-06	07-07	07-08
07-09	07-10	07-11	07-12	07-13	07-14	07-15
07-16	07-17	07-18	07-19	07-20	07-21	07-22
07-23	07-24	07-25	07-26	07-27	07-28	07-29
07-30	07-31	08-01	08-02	08-03	08-04	08-06
08-06	08-07	08-08	08-09	08-10	08-11	08-12
08-13	08-14	08-15	08-16	08-17	08-18	08-19
08-20	08-21	08-22	08-23	08-24	08-25	08-26

Fig.4.2 Calendar of experiment dataset: The red dates are National holidays and weekends.

the actual performance when we are predicting the passenger number in case study.

4.1.2 Metrics

Before introduce all the metrics used to evaluate the models[30], we need to introduce another metric, confusion matrix, which is the basis of the following metrics. In binary case, confusion matrix is a matrix that has 4 metrics: TP (True Positive), FN (False Negative, TN (True Negative),

FP (False Positive). TP indicates how many positive samples are classified rightly. FN indicates

Ground truth / Prediction	Positive	Negative
True	TP	TN
False	FP	FN

Fig.4.3 Confusion matrix expression

how many negative samples are classified wrongly, which means the model classify a negative sample as positive. TN indicates how many negative samples are classified rightly. FP indicates how many negative samples are classified wrongly. Given a dataset U, positive samples Pos, negative samples Neg and the confusion matrix - TP, FN, TN, FP, we can have the following relationships in Fig. 4.3.

- Recall. Recall R represents the ratio of how many positive samples are classified correctly in all positive samples. Its formula is:

$$R = \frac{TP}{TP + FN} \tag{4.1}$$

- Precision. Precision P represents the ratio of how many positive samples are classified correctly in the samples which the model classifies as positive. The formula of precision is:

$$P = \frac{TP}{TP + FP} \tag{4.2}$$

- F1-score. In some situations, we might know that we want to maximize either recall or precision at the expense of the other metric. However, there are always cases where we want to find an optimal blend of precision and recall. For that situation, we can combine the two metrics using what is called the F1 score. The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following formula:

$$\frac{1}{F_1} = \frac{1}{2} * (\frac{1}{P} + \frac{1}{R}) \tag{4.3}$$

$$F_1 = \frac{2 * P * R}{P + R} \tag{4.4}$$

If we want to create a balanced classification model with the optimal balance of recall and precision, we need try to maximize the F1 score.

- Receiver Operating Characteristic (ROC) & Area Under the Curve (AUC)[31]. The ROC curve shows how the recall versus precision relationship changes as we vary the threshold for identifying a positive in our MetaInfo-based model. Finally, we can quantify a model' s ROC curve by calculating the total Area Under the Curve (AUC)
- Passenger number. Intuitively, this is just the metric indicating the number of passenger in the prediction.
- Error rate To indicate the difference between ground truth and prediction, we have the error ratio to indicates the actual performance. Ignoring the accuracy, we emphasize more on the difference between the actual number of passenger and the prediction result. Given the actual number of passenger Num_{real} and the prediction number Num_{pred}, we have equation (21) for error ratio:

$$Error_{ratio} = \frac{\mid Num_{pred} - Num_{real} \mid}{Num_{real}} \tag{4.5}$$

4.2 Analysis

In this section, we mainly have four parts to answer the first three research questions: **RQ1**, **RQ2**, and **RQ3** in the following sections. In the procedure, we divide each dataset into 3 parts: training set, validation set, and test set, and utilize them in different cases under the ratio of 6:2:2. Training set is used to train the internal weights in the model. Validation set is applied to test the model in each training epoch to avoid overfitting problem. And test set is used to test out the performance of the model after the training process in both model variants and hyper-parameter sections. Since we apply random undersampling method to guarantee the sample balance during training procedure, the data we used in the model training is merely part of the whole dataset, which leads to the consequence that the trained model is probably not be able to learn the general mobility pattern if only trained once. To tackle this adverse consequence, we don't merely train the model once instead we train the model repeatedly to achieve the global optimization. And to avoid the overfitting problem which is highly possible to be brought from repeat training, we set the early stop to terminate the training process depending on the result of validation result in each training epoch. To sum up:

1. Implement the repeat training with the same experimental settings, and set initial internal weights of model by the weights from last training.
2. According to the validation result, Execute the early stopping mechanism and other machine learning tricks to avoid overfitting problem.
3. Evaluate the model performance after training with the same test set in model variants and

hyper-parameter sections.

4. Do case studies in different time periods.

4.2.1 Model Variants

In this section, we train and test the variant models on several datasets and compare their performance with the metrics we introduced before. Specifically, we only take the tests on machine learning models to compare the influence of MetaInfo because for statistic methods we cannot merely apply test dataset, which is just part of the whole dataset, to evaluate the model performance, so we make the comparison with statistic models in *Case Study* section instead.
For evaluating model variants, we will conduct the experiments on the dataset *A* and dataset *B*, which represents 2 kinds of different normal situation:

- There is no special holiday in dataset *A*
- In dataset *B*, July 16^{th}, Monday is a special holiday called "Mizu No Hi".

After observing Table A.1 in *Appendix*, it is not hard to find that the MetaInfo method can improve the model performance by adding the condition to do the prediction (**RQ1**). Especially, it is more intuitive to find it out if we look at the values in *Error ratio* column. For every machine learning model with MetaInfo method, the error ratio is relatively lower than those which don't make use of MetaInfo, with approximately 2 3% improvement. And it is not hard to find that the Hidden Markov model cannot handle the long-term trajectory prediction task because of the mechanism itself, only considering the state right before.
Interpretation: As we input the pure trajectory into the model, the model merely consider the trajectory as a kind of time series related data. If we just predict a near future, only the trajectory is probably enough. However, we are challenging a long-term trajectory prediction, where the difficulty is the long-term trajectory is so long that we cannot ensure all the necessary information is learned by the model. Therefore, we complement the potentially necessary information by add meta information. As the result shows, this method does help the performance enhance.
In addition, their prediction speed varies dramatically from the different models. The longest time can even reach to 65 hours, which is obviously not worthy to make a prediction of a week. Among these models, the fastest model is GLU structure with binary output, which spent approximately 3 4 hours to do the calculation. And LightGBM placed the second with around 7 8 hours.

4.2.2 Hyper-parameter

In this section, we conducted the experiments around different trajectory length and discuss the impact brought from it. For evaluating the impact from different input length, we will conduct the experiments on the dataset *B* and dataset *G*, which represents 2 different time range but both predicting the situation of week (2012.07.16 - 2012.07.22).

- Dataset *A* has the input of 1 week (2012.07.09 - 2012.07.15).
- Dataset *G* has the input of 2 week (2012.07.02 - 2012.07.15).

From Table A.2 in *Appendix*, it is not hard to find that the increased input length have the ability to enhance the model performance, however, this improvement depends on the structure ability either, even though the performance doesn't descend dramatically (**RQ2**).
Interpretation: The hidden correlation of trajectory among different weeks does exist but it doesn't absolutely mean that the longer the trajectory is the better prediction we can give. And also the length of the trajectory can affect the efficiency heavily. By expanding the input trajectory, it does help to capture the correlation, however, we believe it is not worthy to enhance the result merely in this way. The final target of our method is to be applied in the real case, the long time cost could be one of the biggest obstacles. Hence, we prefer to use one week trajectory to do the prediction for the following two reasons:

- One week trajectory have the possibility to provide enough information to capture the mobility pattern to some extend.
- Training with one week trajectory cost much less time, and the performance does not fall back behind two week trajectory input.

4.3 Case study

In this section, in order to do further demonstration the performance of our MetaInfo-based method and answe **RQ3**, we conduct 3 case studies at Tokyo Station as follow:

- **Case study 1**: We are predicting the passenger number in a normal period which we assume that it appears regular mobility pattern in this dataset.
- **Case study 2**: The prediction is conducted in a special period - Obon Festival, which dramatically differs from the normal period.
- **Case study 3**: The prediction is conducted after a special period - Obon Festival, which

dramatically differs from the week before.

Since in section *Model Variants* and *Hyper-parameter*, we have found that the performance and efficiency of LightGBM and GLU models are the top ones, so specifically, in order to satisfy the efficiency demand, we only use LightGBM, GLU, and two statistic models (ARIMA and SARIMA). Like the previous section, the following metrics are applied in our case study: F1-score, recall, precision, passenger number, error ratio. In addition, we also visualize the points when the error ratio is over 10% and the passenger number is over 200, which is because when the passenger number is under a specific number no matter how big the error ratio is the passenger number won't change dramatically. Then, I choose 200 as the threshold to extract the error points we need pay more attention.

4.3.1 Case 1: Normal week

In case 1, we choose dataset *B* to train the models, and use dataset *C* to make the prediction of the whole week. Eventually, we predict the passenger number among 2012.07.23 - 2012.07.29 with 1-hour interval, which means we predict the passenger number in every hour among 2012.07.23 - 2012.07.29.
It is obvious that the statistic models can give a pretty accurate prediction in normal situations. However, when they meet the day, Monday, whose mobility pattern is different from last week, the prediction goes wrong seriously. Our assumption is that, last week's Monday (2012.07.16) was a holiday (Mizu No Hi) which has a relatively lower passenger number than the normal Mondays. It is because we fit the statistic models with this abnormal data so that the prediction from statistic is lower than the ground truth.
GLU+MetaInfo method shows accurate prediction at the time of 'rush hours' when there are at least more than 200 passengers, although the prediction goes bad when there are less passengers. We assume that it is because there are many negative samples in the training dataset, and we didn't input the all of negative samples in order to keep the sample balance. At last, comparing the former 3 methods LightGBM show a relatively better performance on weekdays but performs bad for weekend case.

4.3.2 Case 2: Predicting the week of Obon Festival

In case 2, we choose dataset *D* to train the models, and use dataset *E* to make the prediction of the whole week, where Obon Festival(8.13-8.15) is. Eventually, we predict the passenger number among 2012.08.13 - 2012.08.19 with 1-hour interval, which means we predict the passenger

number in every hour among 2012.08.13 - 2012.08.19. As the previous case, both statistic models present a good prediction in most time but they still have the same problem, the prediction strongly follows historical data. There are less passengers during 2012.08.13-2012.08.15 which is during Obon Festival but the prediction of all methods tends to stay at a higher passenger number. GLU+MetaInfo method shows pretty bad results during Obon Festival, giving a extremely high number in the whole week. And LightGBM+MetaInfo performs much better than GLU but generally less stable than statistic models. To sum up, all the methods tends to give a higher passenger number than ground truth, which follows the passenger trend of historical data, and among them the ARIMA model and LightGBM+MetaInfo performs best on weekdays but only ARIMA model performs best on weekends.

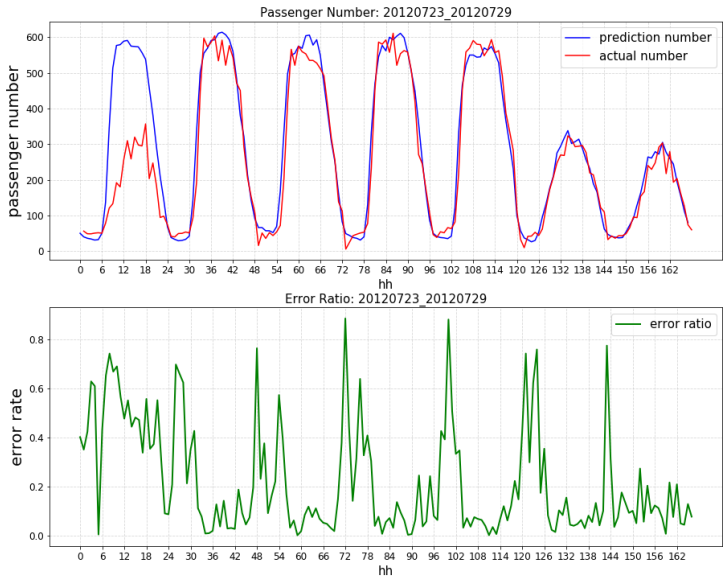
4.3.3 Case 3: Predicting the week after Obon Festival

In case 3, we choose dataset *E* to train the models, and use dataset *F* to make the prediction of the whole week.

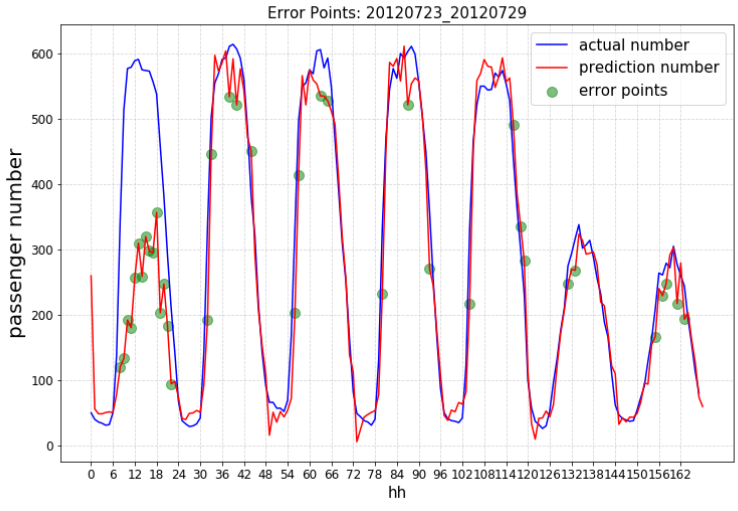
On the contrary to case 2, all the methods tend to give a lower passenger number of ground truth, which follows the passenger trend of historical data. In case 3, ARIMA model and LightGBM+MetaInfo perform nearly the same but ARIMA appears the more stable result.

Summary: From my perspective, when facing the situation that the passenger trend appears under a specific and stable pattern, the statistic methods seem to perform more stable than machine learning methods, and when the passenger trend appears different comparing with historical data the machine learning methods perform better than statistic methods.(**RQ3**) However, for both methods, if the future mobility pattern is globally different from the historical pattern instead of partially different, the prediction result cannot follow the actual mobility pattern well.

And all of the methods possess the same problem: the prediction results follows the historical data to some extend, when the passenger trend is under a specific and stable pattern. Even though our binary classifier method performs relatively well, it is not enough to be put into actual application.(**RQ4**)

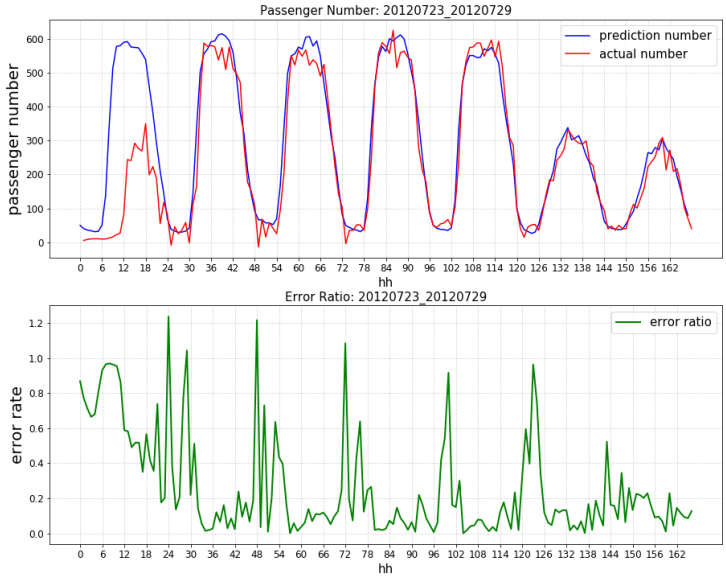


(a) Passenger number and error ratio

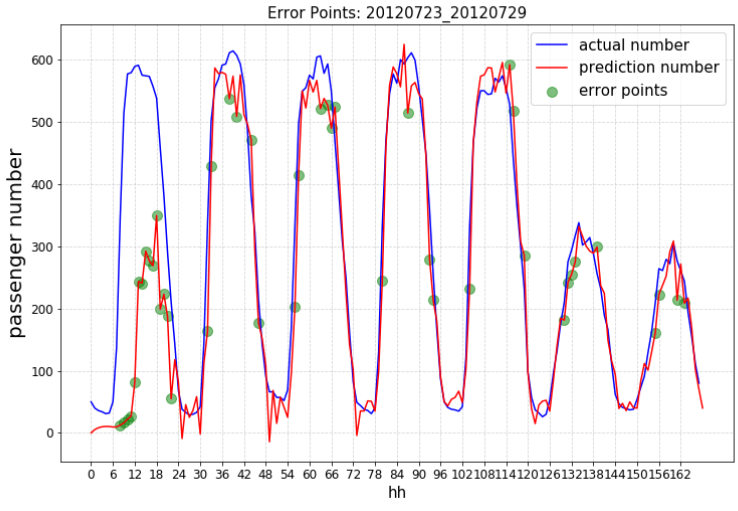


(b) Specific error points

Fig.4.4 Case 1: Predicted passenger number, error ratio, and specific error points of a week from ARIMA model

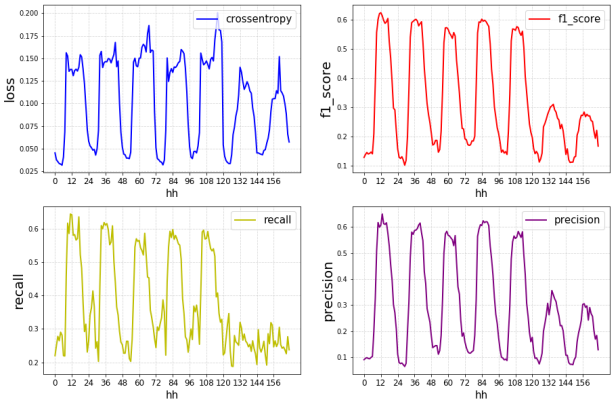


(a) Passenger number and error ratio

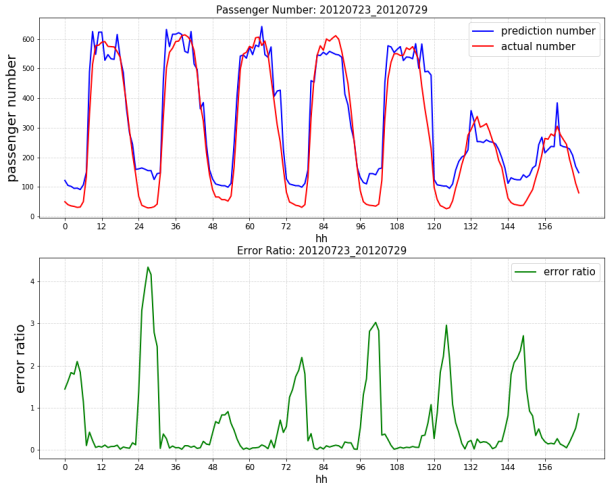


(b) Specific error points

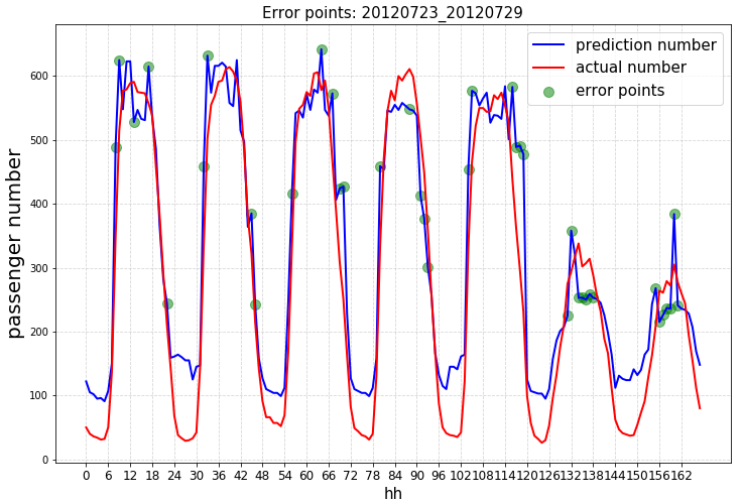
Fig.4.5 Case 1: Predicted passenger number, error ratio, and specific error points of a week from SARIMA model



(a) Metrics for binary classification

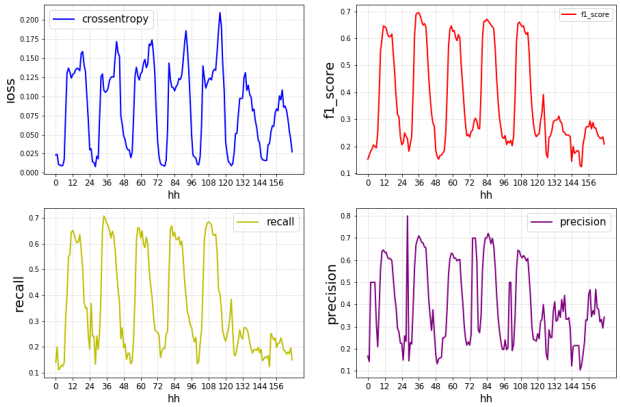


(b) Passenger number and error ratio

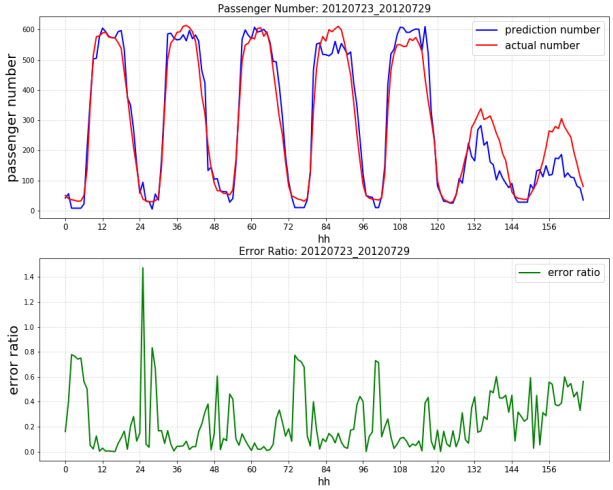


(c) Specific error points: when passenger number is over 200 and error ratio is over 10%

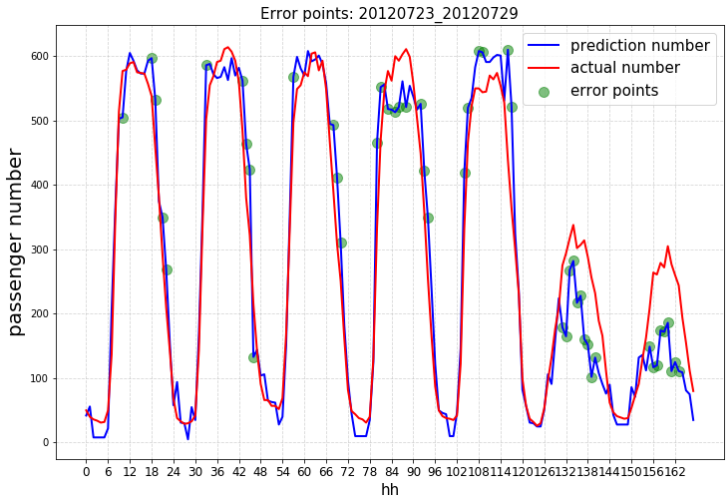
Fig.4.6 Case 1: Metrics, passenger number, error ratio and specific error points of GLU+MetaInfo model



(a) Metrics for binary classification

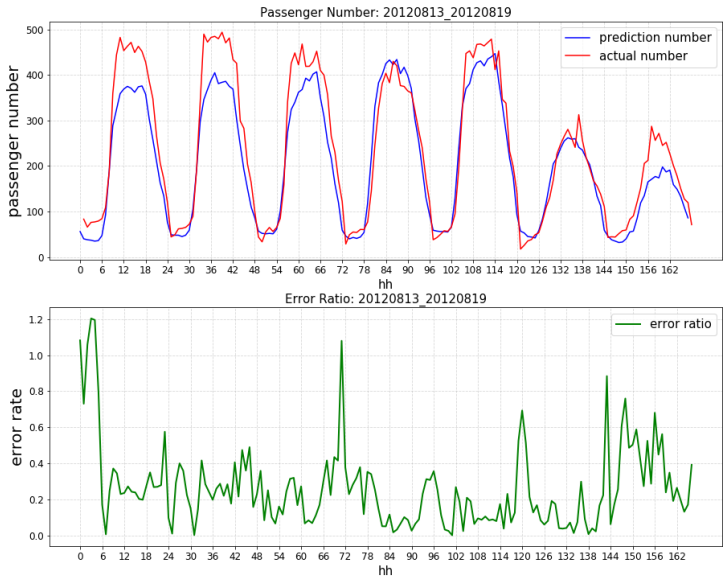


(b) Passenger number and error ratio

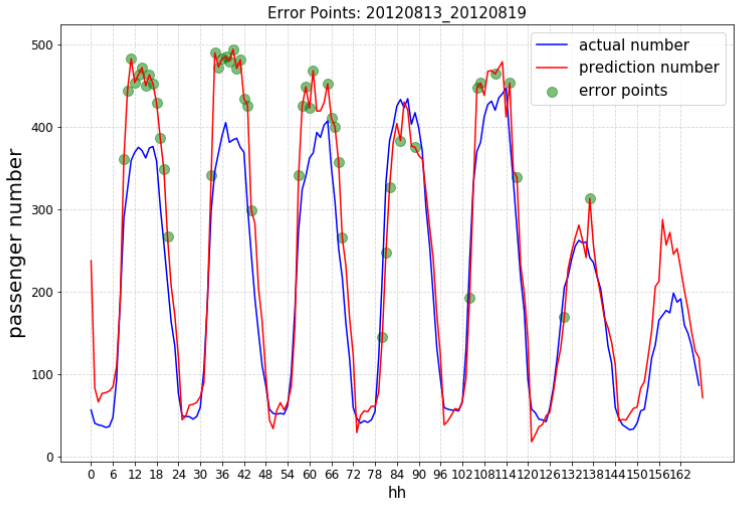


(c) Specific error points: when passenger number is over 200 and error ratio is over 10%

Fig.4.7 Case 1: Metrics, passenger number, error ratio and specific error points of LightGBM+MetaInfo model

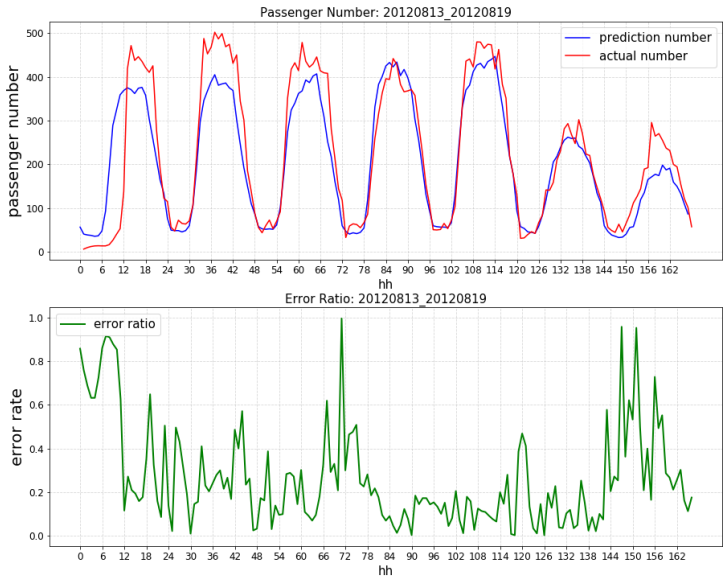


(a) Passenger number and error ratio

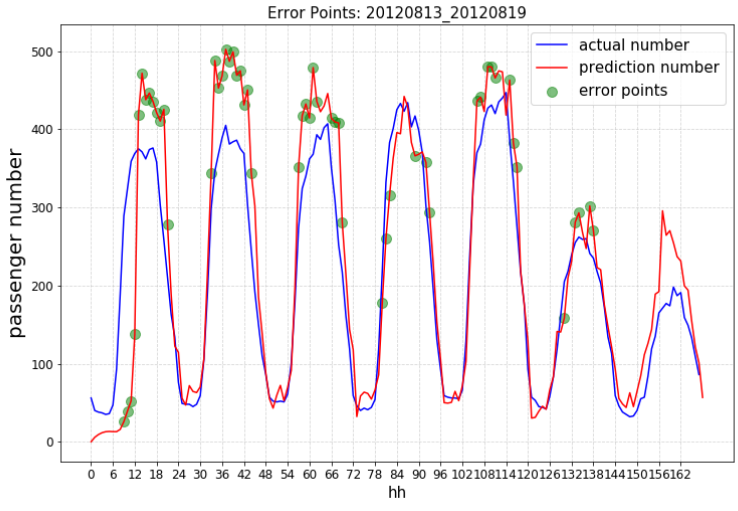


(b) Specific error points

Fig.4.8 Case 2: Predicted passenger number, error ratio, and specific error points of a week from ARIMA model

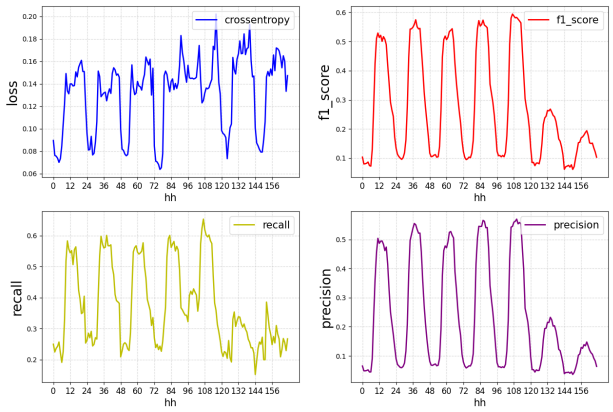


(a) Passenger number and error ratio

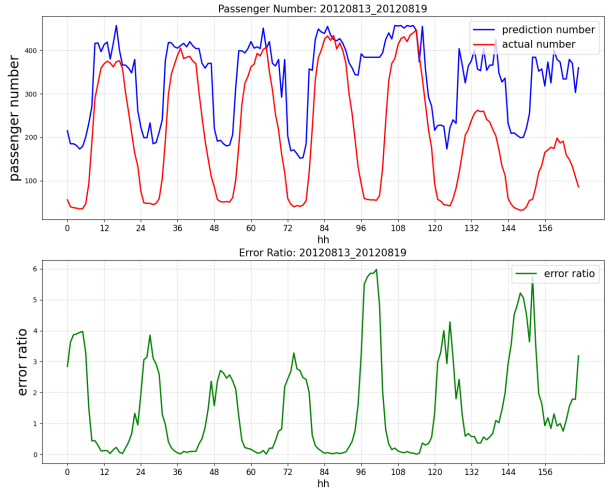


(b) Specific error points

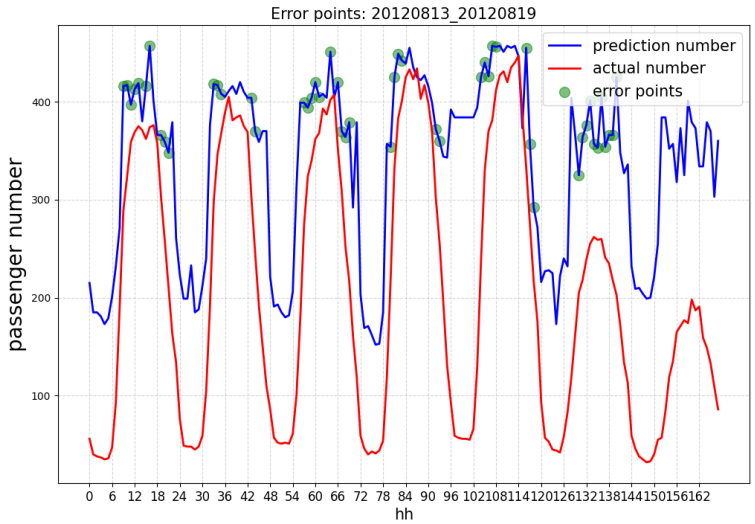
Fig.4.9 Case 2: Predicted passenger number, error ratio, and specific error points of a week from SARIMA model



(a) Metrics for binary classification

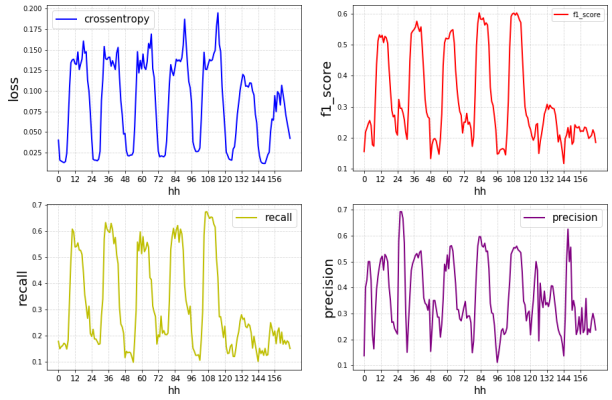


(b) Passenger number and error ratio

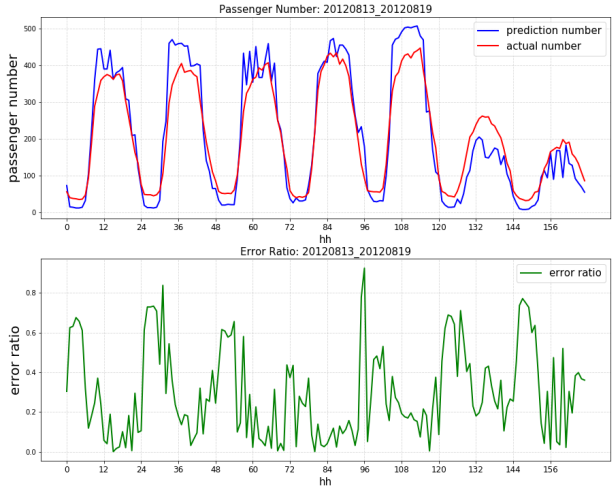


(c) Specific error points: when passenger number is over 200 and error ratio is over 10%

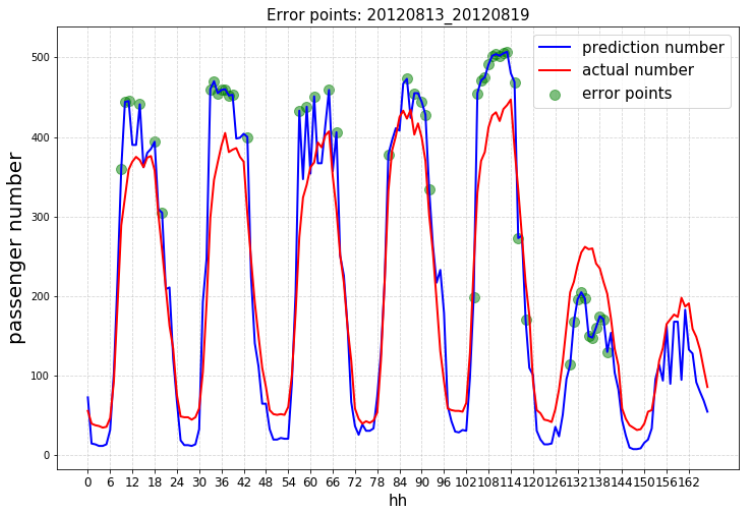
Fig.4.10 Case 2: Metrics, passenger number, error ratio and specific error points of GLU+MetaInfo model



(a) Metrics for binary classification

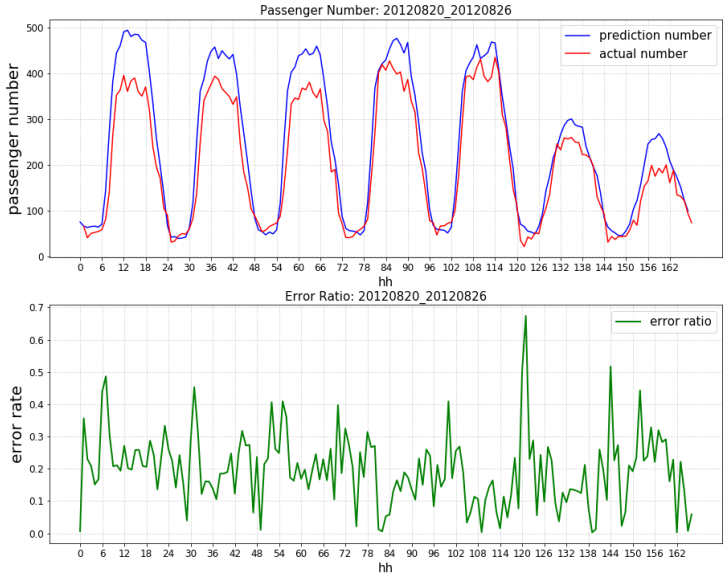


(b) Passenger number and error ratio

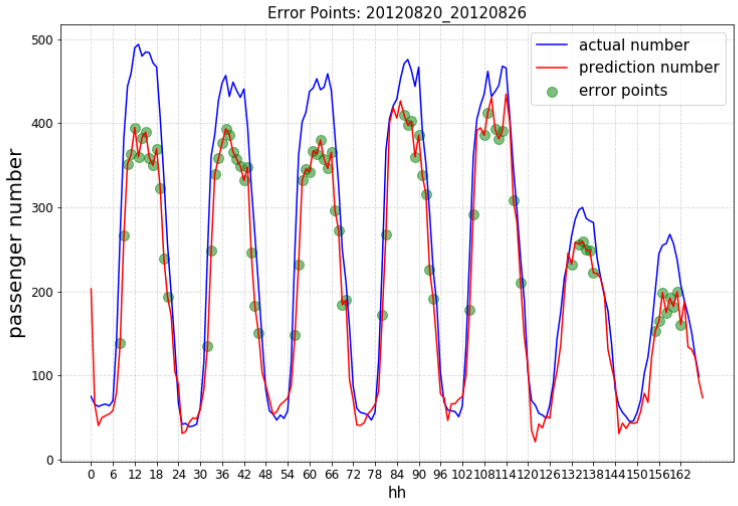


(c) Specific error points: when passenger number is over 200 and error ratio is over 10%

Fig.4.11 Case 2: Metrics, passenger number, error ratio and specific error points of LightGBM+MetaInfo model

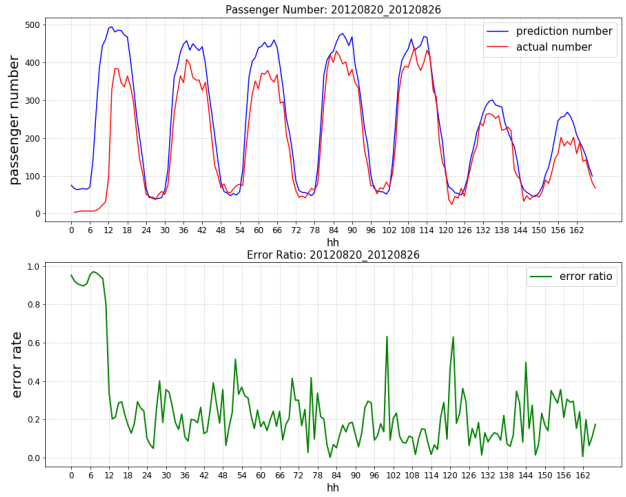


(a) Passenger number and error ratio

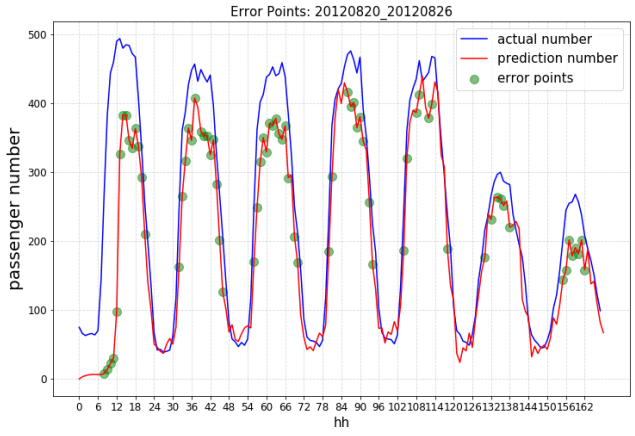


(b) Specific error points

Fig.4.12 Predicted passenger number, error ratio, and specific error points of a week from ARIMA model

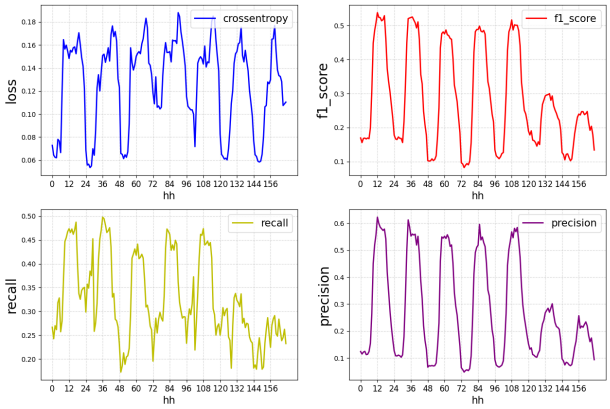


(a) Passenger number and error ratio

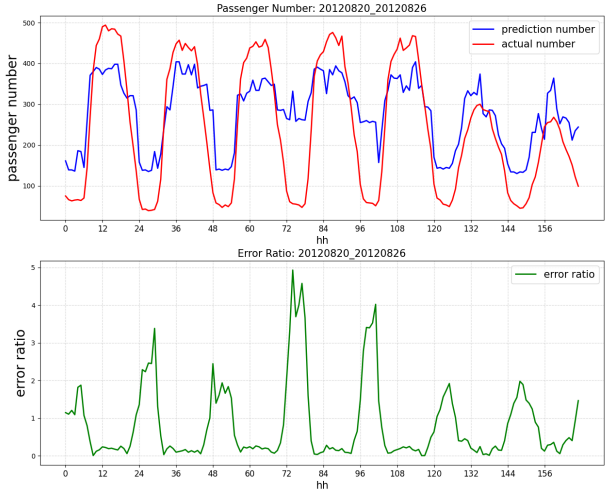


(b) Specific error points

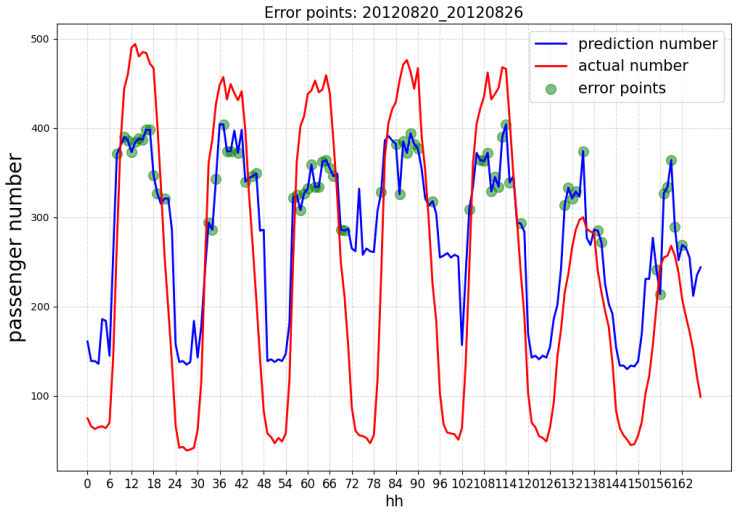
Fig.4.13 Case 3: Predicted passenger number, error ratio, and specific error points of a week from SARIMA model



(a) Metrics for binary classification

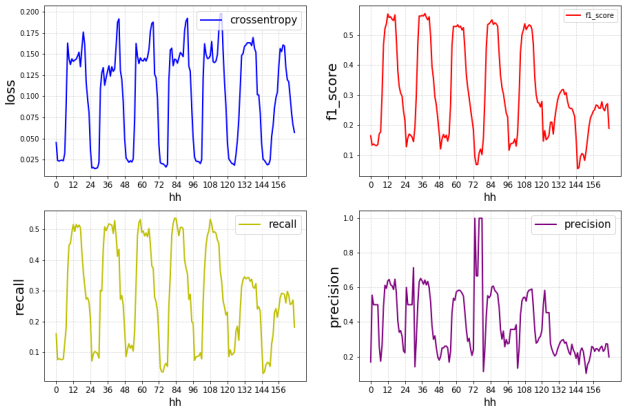


(b) Passenger number and error ratio

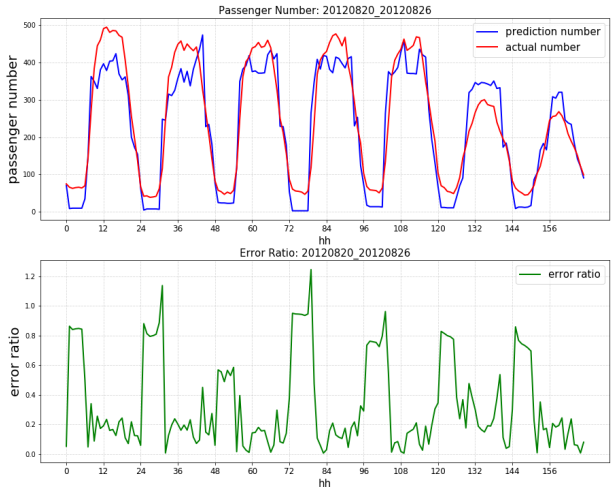


(c) Specific error points: when passenger number is over 200 and error ratio is over 10%

Fig.4.14 Case 3: Metrics, passenger number, error ratio and specific error points of GLU+MetaInfo model



(a) Metrics for binary classification



(b) Passenger number and error ratio

(c) Specific error points: when passenger number is over 200 and error ratio is over 10%

Fig.4.15 Case 3: Metrics, passenger number, error ratio and specific error points of LightGBM+MetaInfo model

第 5 章

Conclusion and future work

5.1 Conclusion

In this research, we proposed a binary classifier idea, aiming to predict the passenger number in the station by predicting whether a person will enter the station or not. In the experiment part, we conduct series of preprocess procedure to enhance the model performance. And we evaluate the machine model performance with different experiment settings. Eventually, considering the accuracy and efficiency we select ARIMA, SARIMA, GLU+MetaInfo and LightGBM+MetaInfo to conduct the case studies at Tokyo station during different time periods. The three case studies show the following conclusions:

1. When facing the situation that the passenger trend appears under a specific and stable pattern, the statistic methods perform more stable than machine learning methods.
2. When the passenger trend appears different comparing with historical data the machine learning methods perform better than statistic methods.
3. All of the methods possess the same problem: the prediction results follows the historical data slightly when the passenger trend is under a specific and stable pattern.
4. The LighGBM+MetaInfo and ARIMA model performs best in the case studies.

In summary, we believe that the binary classifier can capture the features in the long-term mobility pattern but there is still work can be done to improve the performance.

5.2 Future work

For the purpose of making the long-term mobility more reliable, there could be several future works for improving the performance of binary classifier based method. First, more feature engi-

neering could be done to explore the internal correlation of regular mobility pattern because a lot of data features are lost when we transform the trajectory into mesh code format. Secondly, there is still improvement space in the preprocess procedure. At the end of current research, we found that our preprocess method missed some potential passengers, leading to the information lost. Thirdly, in the previous data exploration, we found that the positive and negative sample ratio is extremely low. To figure the imbalanced data problem, we utilize sample balance method to ensure the model learns the positive and negative sample equally. However, even with repeat training, there are still unknown negative samples existing. To tackle this extreme imbalance situation, there could be researches regarding the positive samples as abnormal samples, and make the prediction by finding out the possible abnormal case in the long-term mobility pattern.

Acknowledgements

First of all, I'd like to appreciate my supervisor Prof.Shibasaki, co-advisor Prof. Sezaki and Prof. Song for giving me lots of guidance and advice for my research. Through the lab seminar, Prof. Shibasaki and Prof. Song help me cultivate my research sense and make me learn a lot of state-of-art techniques. Prof. Sezaki spent his time giving me precious advice to evaluate performance which I don't quite understand at the beginning but find it so essential right now.
In the same time, I want to give my thank to the lab members in Shibasaki Lab, especially Renhe Jiang, Zekun Cai, and Quanjun Chen. Jiang-san gave me precious ideas about how to construct the experiment settings to evaluate the different performance. Cai-san helped me out of the difficulty, when there is something going wrong on the server. Chen-san always take his time to answer my instant phone call to help me out of the difficulty. And both of them sacrifice their time to check coding progress and code correctness to prevent from unnecessary time cost. At last but not least, during the 2 years master student life, there are too many things that I need to say 'thank you' for. Even though I cannot point out the names who I need to thank, it is those people who make who I am right now. Without their help, there is no way I can make sure which work field and research field I am heading to.

参考文献

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. Computer Science (2014).

[2] Show-Jane Yen, Yue-Shi Lee. Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset.

[3] C. Wang, L. Ma, R. Li, T. S. Durrani and H. Zhang, "Exploring Trajectory Prediction Through Machine Learning Methods," in IEEE Access, vol. 7, pp. 101441-101452, 2019, doi: 10.1109/ACCESS.2019.2929430.

[4] A Rossi, G Barlacchi, M Bianchini, B Lepri. IEEE Transactions on Intelligent Transportation Systems

[5] Feng, Jie, et al. "Deepmove: Predicting human mobility with attentional recurrent networks." Proceedings of the 2018 world wide web conference. 2018.

[6] Liu, Qiang, et al. "Predicting the next location: A recurrent model with spatial and temporal contexts." Thirtieth AAAI conference on artificial intelligence. 2016.

[7] Gao, Qiang, et al. "Predicting human mobility via variational attention." The World Wide Web Conference. 2019.

[8] Fan, Zipei, et al. "Decentralized Attention-based Personalized Human Mobility Prediction." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 3.4 (2019): 1-26.

[9] Altman, Naomi S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879. hdl:1813/31637.

[10] Mani, Inderjeet, and I. Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction." Proceedings of workshop on learning from imbalanced datasets. Vol. 126. 2003.

[11] Ivan Tomek, "Two Modifications of CNN," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-6, no. 11, pp. 769-772, Nov. 1976, doi: 10.1109/TSMC.1976.4309452.

[12] D. L. Wilson, ”Asymptotic Properties of Nearest Neighbor Rules Using Edited Data,” in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-2, no. 3, pp. 408-421, July 1972, doi: 10.1109/TSMC.1972.4309137.

[13] Kubat, M., (2000). Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. Fourteenth International Conference on Machine Learning.

[14] Ke, Guolin, et al. ”Lightgbm: A highly efficient gradient boosting decision tree.” Advances in neural information processing systems. 2017.

[15] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, pages 1189–1232, 2001.

[16] Box, George EP, et al. Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[17] Sepp Hochreiter and J ¨ urgen Schmidhuber. Long short-term memory. Neural computation,9(8):1735–1780,1997.

[18] Kyunghyun Cho, Bart Van Merri ¨ enboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078,2014.

[19] Yann LeCun et al. Lenet-5, convolutional neural networks. URL: http://yann.lecun.com/exdb/lenet, 2015.

[20] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, Caroline Suen, Adam Coates, Andrew Maas, Awni Hannun, Brody Huval, Tao Wang, et al. Unsupervised feature learning and deep learning, 2013.

[21] Dauphin, Yann N., et al. ”Language modeling with gated convolutional networks.” International conference on machine learning. 2017.

[22] Vaswani, Ashish, et al. ”Attention is all you need.” Advances in neural information processing systems. 2017.

[23] B. Wang, Y. Wang, K. Qin and Q. Xia, ”Detecting Transportation Modes Based on LightGBM Classifier from GPS Trajectory Data,” 2018 26th International Conference on Geoinformatics, Kunming, 2018, pp. 1-7, doi: 10.1109/GEOINFORMATICS.2018.8557149.

[24] Li, J. Z. ”Monthly Housing Rent Forecast Based on LightGBM (Light Gradient Boosting) Model.” International Journal of Intelligent Information and Management Science 7.6 (2018).

[25] Z. Mei, F. Xiang and L. Zhen-hui, ”Short-Term Traffic Flow Prediction Based on Combination Model of Xgboost-Lightgbm,” 2018 International Conference on Sensor Networks and Signal Processing (SNSP), Xi’an, China, 2018, pp. 322-327, doi: 10.1109/SNSP.2018.00069.

[26] Zeng, Hong, et al. ”A lightGBM-based EEG analysis method for driver mental states classification.” Computational intelligence and neuroscience 2019 (2019).

[27] Zhou, Kaibo, et al. ”Fast prediction of reservoir permeability based on embedded feature selection and LightGBM using direct logging data.” Measurement Science and Technology 31.4 (2020): 045101.

[28] Machado, Marcos Roberto, Salma Karray, and Ivaldo Tributino de Sousa. ”LightGBM: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry.” 2019 14th International Conference on Computer Science & Education (ICCSE). IEEE, 2019.

[29] Weng, Tingyu, Wenyang Liu, and Jun Xiao. ”Supply chain sales forecasting based on lightGBM and LSTM combination model.” Industrial Management & Data Systems (2019).

[30] Tharwat, Alaa. ”Classification assessment methods.” Applied Computing and Informatics (2018).

[31] Balayla, Jacques. ”Prevalence Threshold and the Geometry of Screening Curves.” arXiv preprint arXiv:2006.00398 (2020).

[32] Hyndman, Rob J., and George Athanasopoulos. Forecasting: principles and practice. OTexts, 2018.

[33] Furman, Alex. ”Random walks on groups and random transformations.” Handbook of dynamical systems. Vol. 1. Elsevier Science, 2002. 931-1014.

[34] Karatzas, Ioannis, and Steven E. Shreve. ”Brownian motion.” Brownian Motion and Stochastic Calculus. Springer, New York, NY, 1998. 47-127.

[35] Barmak, Daniel Hernan, Claudio Oscar Dorso, and Marcelo Otero. ”Modelling dengue epidemic spreading with human mobility.” Physica A: Statistical Mechanics and its Applications 447 (2016): 129-140.

[36] Ramos, Raul, and Jordi Suriñach. ”A gravity model of migration between the ENC and the EU.” Tijdschrift voor economische en sociale geografie 108.1 (2017): 21-35.

[37] Beiró, Mariano G., et al. ”Predicting human mobility through the assimilation of social media traces into mobility models.” EPJ Data Science 5.1 (2016): 30.

[38] Pereida, Karime, Mohamed K. Helwa, and Angela P. Schoellig. ”Data-efficient multirobot, multitask transfer learning for trajectory tracking.” IEEE Robotics and Automation Letters 3.2 (2018): 1260-1267.

付録 A

Experiment results

TableA.1 Comparison among variant machine learning models: dataset *A* with 22378 passengers in total and 8898 passengers in the station; dataset *B* with 17911 passengers in total and 9050 passengers in the station

Method	Datasets	F1-score	Recall	Precision	AUC	Passenger number	Error ratio
LightGBM+MetaInfo	A	0.9059	0.8793	0.9343	0.9727	8517	0.0589
	B	0.9090	0.8880	0.9309	0.9730	8546	0.0557
LightGBM	A	0.8977	0.8609	0.9378	0.9665	8168	0.0820
	B	0.9048	0.8735	0.9384	0.9687	8424	0.0692
GRU+MetaInfo	A	0.8847	0.8587	0.9139	0.9511	8353	0.0612
	B	0.8823	0.8456	0.9239	0.9468	8284	0.0846
GRU	A	0.8616	0.8230	0.9061	0.9308	8082	0.0917
	B	0.8768	0.8718	0.8819	0.9398	8946	0.0115
LSTM+MetaInfo	A	0.8900	0.8848	0.8966	0.9564	8821	0.0086
	B	0.8872	0.8854	0.8908	0.9502	8998	0.0057
LSTM	A	0.8652	0.8538	0.8790	0.9327	8642	0.0287
	B	0.8798	0.8578	0.9030	0.9430	8598	0.0500
GLU+MetaInfo	A	0.8798	0.8565	0.9060	0.9575	8278	0.0697
	B	0.8810	0.8561	0.9090	0.9574	8527	0.0578
GLU	A	0.8512	0.8053	0.9051	0.9367	7920	0.1098
	B	0.8662	0.8335	0.9038	0.9472	8348	0.0776
HMM	A	0.3583	0.2716	0.4699	0.4848	6442	0.2760
	B	0.3353	0.3071	0.3692	0.3946	7402	0.1821

TableA.2 Comparison among variant machine learning models: dataset *B* with 22378 *uid* in total and 8898 passengers in the station; dataset *G* with 14838 *uid* in total and 7450 passengers in the station

Method	Datasets	F1-score	Recall	Precision	AUC	Passenger number	Error ratio
LightGBM+MetaInfo	B	0.9090	0.8880	0.9309	0.9730	8546	0.0557
	G	0.9199	0.9020	0.9385	0.9797	7160	0.0389
LightGBM	B	0.9048	0.8735	0.9384	0.9687	8424	0.0692
	G	0.9128	0.8862	0.9410	0.9747	7016	0.0583
GRU+MetaInfo	B	0.8823	0.8456	0.9239	0.9468	8284	0.0846
	G	0.8774	0.8388	0.9199	0.9418	6794	0.0881
GRU	B	0.8768	0.8718	0.8819	0.9398	8946	0.0115
	G	0.8762	0.8377	0.9185	0.9403	6795	0.0879
LSTM+MetaInfo	B	0.8872	0.8854	0.8908	0.9502	8998	0.0057
	G	0.7601	0.6590	0.8980	0.8519	5468	0.2661
LSTM	B	0.8798	0.8578	0.9030	0.9430	8598	0.0500
	G	0.8048	0.6940	0.9575	0.8761	5400	0.2752
GLU+MetaInfo	B	0.8810	0.8561	0.9090	0.9574	8527	0.0578
	G	0.8705	0.8492	0.8971	0.9489	7052	0.0534
GLU	B	0.8662	0.8335	0.9038	0.9472	8348	0.0776
	G	0.8651	0.8273	0.9087	0.9485	6788	0.0888
HMM	B	0.3353	0.3071	0.3692	0.3946	7402	0.1821
	G	0.3237	0.2923	0.3499	0.3874	6104	0.1806

２０２０年度　修士論文　二項分類器による乗客数の予測

陳　柏嘉