Predicting the Price of Second-Hand Housing Based on Lambda Architecture and KD Tree

Qinghe Pan[1], Zeguo Qiu[2], Yaoqun Xu[2] and Guilin Yao[3]

Abstract—**In this paper a system is designed and implemented to predict the price of second-hand housing. This system based on Lambda architecture can execute prediction in both real-time and batch modes so it can give two kinds of different price predictions that reflect current and historical conditions respectively. The *kNN* related algorithms are used for price prediction. By comparing the performance of brute *kNN, kd tree* and *ball tree, kd tree* is selected as the price prediction model of the system. In system implementation the kd tree model is chosen to predict prices in both real-time and batch services. The kd tree model can also recommend housings to user besides price prediction. The experiment shows the effectiveness of our system. Time and space performance of brute *kNN, kd tree* and *ball tree* are compared by experiments. And the evaluation metrics of other available maching learning models are compared. The reason of choosing the kd tree model is also explained by the experimental results.**

Index Terms—**Lambda Architecture, real-time system, batch system, kd tree.**

I. INTRODUCTION

IN recent years the techniques of big data have been developing in industrial and commercial areas. Emerging business requirements continually bring new challenges to software and hardware architectures in the big data industry. One of the most important fields is prediction which can greatly help individuals and enterprises to make decisions. Lambda architecture [1], [2], [3], [4] can help us achieve this goal with its special construction. This architecture consists of three layers: speed, service and batch layers. The speed layer is responsible for real-time computation on streaming data and the batch layer is in charge of batch computation on historical big data. The service layer can provide services to users so one can get computation results based on the latest and historical data at the same time.

Lambda architecture has been deployed in various applications such as recommendation system [5], [6], anomaly detection [7], [8], monitoring system [9] and so on. In this paper it is used to predict the second-hand housing prices in China where the second-hand housing transactions are usually processed in trading agents. The sellers provide their housing information to realtors of the agents who publish the second-hand housing information on websites. The buyers can search on websites to choose the interesting housings. The price of the second-hand housing is a more attracting attribute than other attributes. This paper designs and implements a system to predict the second-hand housing price based on Lambda Architecture. The system can help realtors to publish reasonable housing price and buyers to learn the current market price. And it can also be used to study the price trend by research institutions, and be used by bankers to evaluate the price of mortgaged housing.

In [1] the similar problem has been studied and an architecture named Alarea has been designed and applied successfully in price prediction of real estate. In [1] the data of real estate transactions on different categories from Spanish Ministry of Development (2004-2016) is used in batch layer. And data from Twitter real-time API is used in real-time layer. The influence of sentiment(like tweets) on housing prediction has been studied in [10], [11], [12]. In [13] and [14], real-time sentiment analysis is particularly studied. Based on big data technology, multiple data sources can be associated together, and more accurate prediction can be provided from both historical and real-time perspectives. Our research is very similar to [1], but in the batch layer and real-time layer, our system uses the same data source. In future research we will consider to absorb other real-time factors and ingredients into our existing system to improve the prediction effect.

The representative models of housing price prediction are AVMs (Automated Valuation Models) [15]. The traditional benchmark for AVMs is the hedonic model based on the theory that the price of an asset is a function of its quantifiable characteristics. Now instead of hedonic models most AVMs use some type of ML technique [16], such as *neural network*, *decision tree*, *random forest*, *SVM regression* and so on. In this paper, *kd(k dimensional) tree* [17], [18] is used as the prediction model in batch and real-time layers. Because *kd tree* resides in memory after it is trained, it can also be used in service layer to provide querying service. Compared to other kNN(K Nearest Neighbor) related algorithms such as *brute kNN* and *ball tree* [19], [20], *kd tree* has its advantages in performance. Other models such as *linear regression*, *neural network*, *SVM* and so on can also be used in in batch and real-time layers. In experiment the performances of these models are compared with *kd tree*.

Besides Lambda architecture, other architectures or systems with reasonable techniques stack can achieve the same prediction goal. Based on Kafka [21] and Flink [22], Kappa architecture [23] can unify batch processing and real-time computing. Alarea [1] can handle source data in different formats. Compared with Lambda and Kappa architectures, it has equivalent or better quality attributes, so it is very attractive.

The rest of this paper is arranged by the following way. In Section II the characteristics of Lambda architecture are

[1,2,3] School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China.
[1]corresponding author (e-mail: panqh@hrbcu.edu.cn)

DOI: 10.36244/ICJ.2022.1.1

described, and Kappa and Alarea architectures are briefly introduced and analyzed. In Section III kNN-related models are discussed and the housing data is briefly explored. In Section IV-A the results of experiment are analyzed. It describes the details of components in our system, discusses experiment results and compares the performance and metrics of different models. Lastly, the paper is concluded in Section VI.

II. RELATED ARCHITECTURES

A. Lambda architecture

Lambda Architecture is initiated by Nathan Marz [24]. There are three layers in this architecture. The names and functions of the layers are described in Table I. The batch layer is responsible for providing batch views based on the master database. The speed layer executes fast and timely computation that will compensate for high latency to serving layer. This schema can also be depicted in Fig. 1 [24]. In Fig. 1 the new data flows into two places. One direction is to batch layer where the new data will be appended to the master database so when next update the batch views will contain the new data. The other direction is to speed layer where the same new data will be accumulated in this layer to produce real-time views before the next iteration of batch views in serving layer. The functions of Lambda Architecture in full can be summarized by these three equations:

batch view = function(all data)
real-time view = function(real-time view, new data)
query = function(batch view, real-time view)

TABLE I
THE NAMES AND FUNCTIONS OF LAYERS IN LAMBDA ARCHITECTURE.

Layers	Function Descriptions
Speed layer	1. Compensate for high latency of updates to serving layer. 2. Fast, incremental algorithms. 3. Batch layer eventually overrides speed.
Serving layer	1. Provide access services to batch views. 2. Updated by batch layers.
Batch layer	1. Store growing master dataset. 2. Compute functions on the dataset. 3. Provide batch views.

B. Kappa architecture

Kappa architecture [23] is a simplification of Lambda architecture. Compared to Lambda architecture Kappa unifies the batch and real-time layers. The structure of Kappa is shown is Fig. 2. The main reason for the popularity of Kappa is the Apache Kafka [21] and Apache Flink [22] frameworks. Kafka not only acts as a message queue, but also can save historical data for a longer time to replace the batch layer in Lambda architecture. Flink takes an earlier time as the starting point and plays the role of batch processing. At the same time, Flink solves the problem of accuracy of calculation results under the disorder of events. If batch processing is consistent with real-time processing, Kappa is more appropriate. However, in some other scenarios, the whole historical data set needs to be processed in batch, so Lambda architecture is more appropriate.

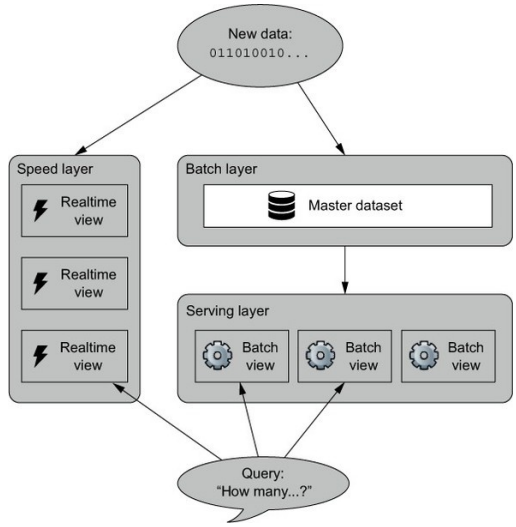


Fig. 1. The illustration of Lambda Architecture.

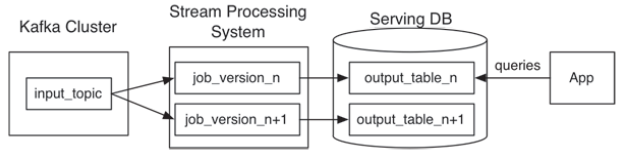


Fig. 2. The illustration of Kappa Architecture.

C. Alarea architecture

Alarea [1] is an architecture that combines batch processing and real-time processing in two different layers and has been deployed to deal with big data and real-time data in the real estate domain. Compared with Lambda and Kappa, Alarea has the following three advantages.

1. Alarea mixes and integrates heterogeneous data sources.
2. Alarea gives developers the opportunity to decide which layer is better for their purposes.
3. Alarea copes with two kind of data processing and is capable of treating it no matter the timing that they present.

In [1] Lambda, Kappa, and Alarea are also compared based on the four quality attributes, including recoverability, fault tolerance, new data gap and hardware consumption.

III. MODELS

A. KNN Models

The most effective method to evaluate the price of a house in a building is to obverse the prices of the nearest floors above or beneath it. So *kNN*(K Nearest Neighbors) model is an intuitive approach to predict the housing price. Compared to other prediction models *kNN* has two advantages in our application scenario.

The first advantage is that *kNN* can get more accurate result when the number of sample data is few. *KNN* only needs to find out the *k* nearest neighbors from the sample data but other prediction models may have to get more data to execute the training. If the dataset only contains two samples(*k=2*), for example the housing information of the 4th and 6th floors. If the task is to predict the price of housing on the 5th floor that between 4th and 6th floors in the same building, then *kNN* model will get the 4th and 6th floors as its 2 nearest neighbors and predict the 5th housing price by averaging the prices of the two neighbors. But for others models two samples are too small to execute the training.

The second advantage is that *kNN* can also generate a recommending housings list by the *k* nearest neighbors themselves besides predicting housing price.

Assume that the data set $X = \{x_1, \ldots, x_m\}$, where m represents the number of housing records. $x_i = (x_{i1}, \ldots, x_{in})$, $1 \leq i \leq m$, where *n* is the number of fields in each record and x_{in} is the *price* field. Suppose the data of a house is $h = (h_1, \ldots, h_{n-1}, h_{price})$, where $h_1, \ldots, h_{n-1}$ are known and h_{price} is the targeting value that will be predicted by *kNN*. The following steps describe the details of prediction by *kNN* method.

Step 1. For each data record *x* in *X*, its *1st* to *(n-1)th* fields are used to calculate the Euclidean distance from *h* and the results are collected and sorted to find the *k* nearest data records. Let $X_k = (x_{s_1}, \ldots, x_{s_k})$ be the vector representing the *k* nearest data records by distance from near to far, and the corresponding distance vector is $Dist_k = (dist_{s_1}, \ldots, dist_{s_k})$.

Step 2. Use the *nth* field of the *k* data records to estimate the value of h_{price}, as shown in (1).

$$h_{price} = \frac{\sum_{i=1}^{k} x_{s_i n}}{k} \tag{1}$$

Where $x_{s_i n}$ is the *nth* field of record x_{s_i}, i.e., the *price* field.

The variation of *kNN* is *Weighted kNN* that considers the influences of different distances on h_{price} instead of directly averaging the *nth* field of the *k* data records in (1). The following (2) calculates the weight for each data record based on the distance, where δ is the appropriate positive constant.

$$w_{s_i} = e^{-\frac{dist_{s_i}{}^2}{2\delta^2}} \tag{2}$$

The value range of w_{s_i} is (0, 1]. The larger the value of $dist_{s_i}$, the smaller the value of w_{s_i}, and vice versa. These characteristics determine that w_{s_i} is a better choice for representing weight. After the introduction of weights, the calculation of h_{price} is shown in (3).

$$h_{price} = \frac{\sum_{i=1}^{k} w_{s_i} x_{s_i n}}{\sum_{i=1}^{k} w_{s_i}} \tag{3}$$

B. Kd tree and ball tree

The naive or brute *kNN* is to choose the *k* nearest neighbors from the *n* samples by sample-wise comparison. The time complexity to predict price of one housing is $O(n^2)$. When n is large the brute *kNN* is not a practicable method. Many improved methods are designed to solve this problem such as the *kd tree* [17], [18], *ball tree* [19], [20], *Hybrid Spill Tree* [25], [26] and so on. The time complexity of these tree-based methods are $O(log(n))$.

The process of building *kd tree* is recursive process that includes two main steps shown in Fig. 3.

Step1. Computes the variance of the dimensions and splits the data at median based on the dimension *Root* that has the highest variance. Let the two split datasets be *Left* and *Right*.

Step2. Repeat the step 1 for *Left* and *Right* until *Left* and *Right* cannot be split.

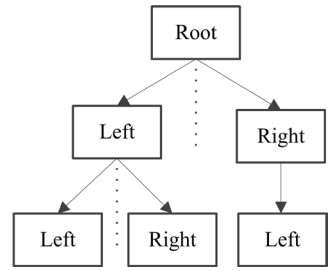


Fig. 3. The construction of *kd tree*.

The process of building ball tree is also a recursive process that includes two main steps shown in Fig. 4.

Step1. All data is split into two almost equal sized balls $ball_A$ and $ball_B$.

Step2. Repeat the step 1 for $ball_A$ and $ball_B$ until $ball_A$ and $ball_B$ cannot be split.

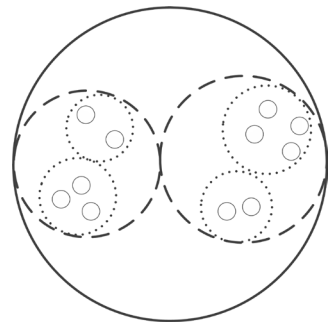


Fig. 4. The construction of ball tree.

In this paper the *kd tree* is used in both batch and speed layers in Lambda architecture. The reason will be explained in detail in Section V. After trained by data the *kd tree* will reside in memory and provide prediction service until next iteration. For instance, in batch layer the *kd tree* is trained every day and in speed layer the training frequency is decided by the size of time windows.

IV. EXPERIMENT

A. Data

Usually, the second-hand housing data is crawled from the official websites on the Internet. A distributed crawling system is designed and deployed on cloud to gather data. In such

system each node has its IP address and crawlers on each node run as daemons to crawl data from the websites that own large real-time and historical housing information. By the research purpose the second-hand housing data used in this paper is from one public dataset [27]. This dataset includes 318851 housing information records in Beijing from 2011 to 2017. The data in this dataset will be used as input to our system. It means that the crawlers in our system can directly read records in this dataset as if the data obtained from websites on Internet. This way will save us the time to clean data and help us focus on the system implementation itself. One notation is that housing price in this dataset is the final price but we will ignore this attribute. So the price can be used as the listing price or final price. Each record in dataset consists of 26 fields but only 11 fields are used. The names and meanings of these fields are listed in Table II.

TABLE II
THE NAMES AND MEANINGS OF FIELDS IN DATASET.

Field names	Meanings
bedroom	the number of bedroom
living room	the number of living room
bathroom	the number of bathroom
kitchen	the number of kitchen
floor	the height of the house
ladder ratio	the number of ladders a resident have on average
square	the square of house
subway	not near to subway(0) and near to subway(1)
lat	the latitude of house
lng	the longitude of house
price	the average price by square

For example the list [3, 2, 2, 1, 6, 0.5, 102.66, 0, 39.873867, 116.66589] consists of data items corresponding to the fields in table. For training *kd tree* the data must be processed further. Firstly, the dataset should be split into *X* matrix and *Y* targeting vector. The *Y* consists of all housing prices in *price* field and the *X* matrix consists of data in other fields except for *price* field. Secondly, the *X* matrix should be normalized before it is used to train *kd tree*. For example the *subway* field should be one-hot encoding and other nine fields in the *X* matrix should be standardized by using max-min or std-dev methods. The data is explored and the results are shown in Fig. 5.

The Fig. 6 shows the relationship between the *price* field and the *longitude* and *latitude* fields. It can be seen that the closer to the central urban area, the higher the average housing price, the more housings for sale, and the more frequent transactions.

The Pearson correlation coefficient between data fields is shown in the Fig. 7. It can be seen that the *bedroom*, *living room* and *bathroom* fields are positively correlated and have high correlation coefficients. At the same time, there are also strong positive correlations between the *square* field and the *bedroom*, *living room* and *bathroom* fields. The *subway* field has a strong positive correlation with the *price* field. Because the data comes from Beijing and it belongs to the economic center, whether there is a subway has a great impact on the housing price. The *bedroom*, *living room* and *bathroom* fields have negative correlations with the *price* field, because usually the total price of houses with large area is higher, which makes it more difficult to sell, resulting in lower the *price* field.

B. System structure

In this experiment the main frameworks used in our system are described in Table III. The concise experiment topology is shown in Fig. 8. Both Apache Flume and Apache Kafka are distributed, reliable, and available components. They are widely used in big data processing. In this research they are combined together to implement Lambda architecture. There are four reasons for this combination.

1. In production environment because there are many real-time data sources, it is not convenient to build many Kafka clients to publish data to central topics with one topic per activity type. So, usually in implementation the real-time data is delivered to Flume firstly.

2. There are many interceptors in Flume that can be used to filter and clean data. It is more convenient to process data in Flume than in Kafka.

3. When Flume is connected to Spark streaming directly without Kafka, if the speed of data flowing in is faster than the speed of data processing in Spark streaming then the data that cannot be processed in time will be lost. It can be solved by using Kafka between Flume and Spark streaming. Kafka likes a cache that can store data over a period of time.

4. Besides connecting Flume to Spark streaming, Kafka can also provide data to Cassandra for persistent storage. So, Kafka is the core component in our implementation of Lambda architecture.

There are two crawlers *c1* and *c2* for housing data collecting. As described in IV-A the *c1* and *c2* read records in dataset as if they crawl them from websites on Internet. The *c1* and *c2* put their data into *f1* and *f2* respectively. The *f1* and *f2* put their data into *f3*. The *f3* directly connects to Kafka. Spark streaming system fetch data from Kafka based on the time window. The fetched data flows into two directions. In one direction the data is pushed to speed layer and in this layer the *kd tree* model is trained by the data. In the other direction the same data is appended to the existing distributed Cassandra database and in the batch layer the *kd tree* is trained by all or partial data in Cassandra. For speed layer the model training frequency is decided by the size of time window which can be set half an hour, one hour, or two hours and so on. The model training frequency for batch layer can be set one day, two days and so on. Based on speed layer the real-time service can be built which can provide price prediction and house recommending based on the coming data in the time window. The batch service built on batch layer can do the same task based on the historical data. The results from real-time and batch services can support decision making from different perspectives.

C. Experimental Results

We can test the effectiveness of price prediction and housing recommendation by submitting some data to the real-time and batch service respectively. For example, after running a

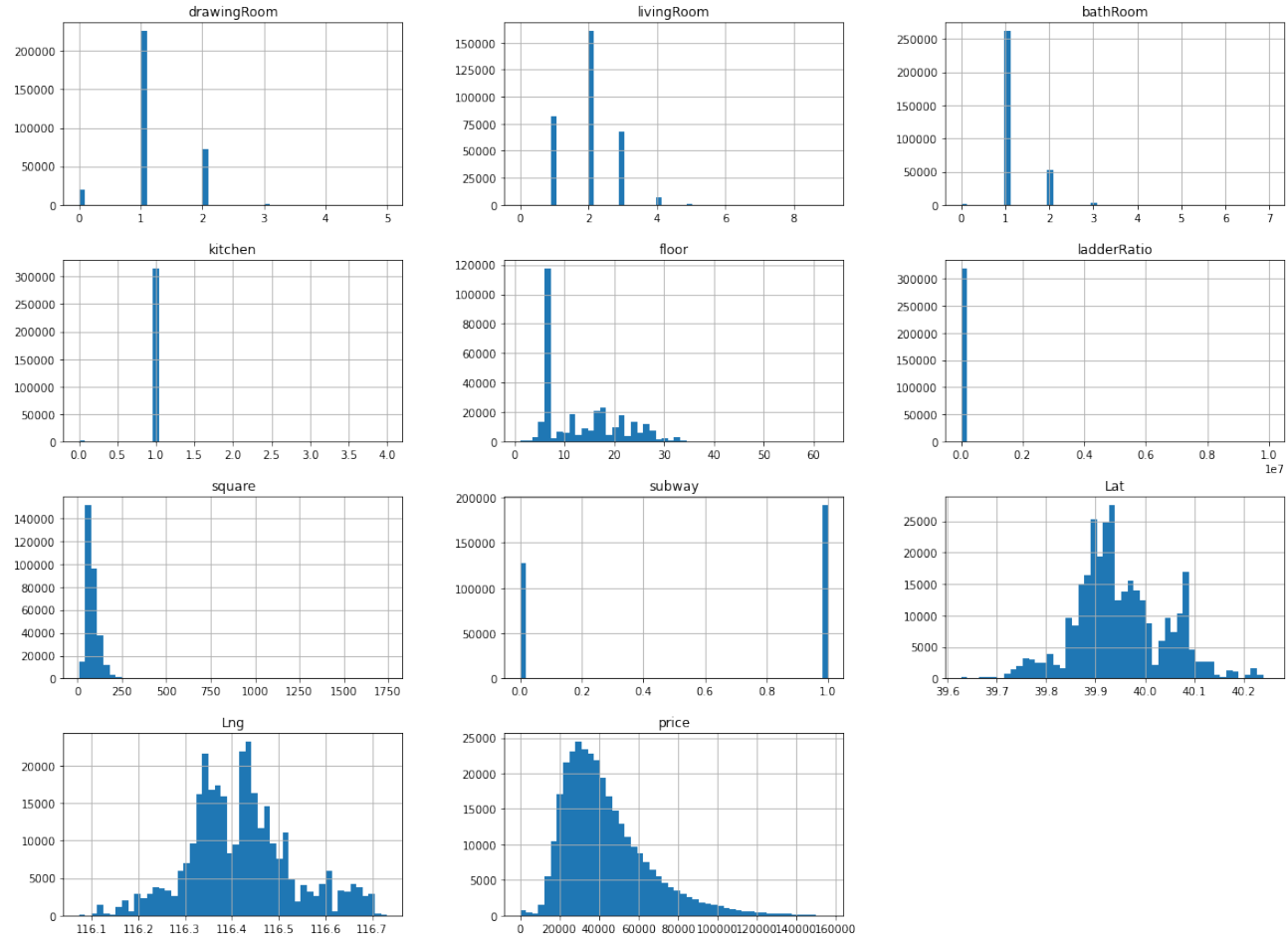


Fig. 5. The exploration of all data fields.

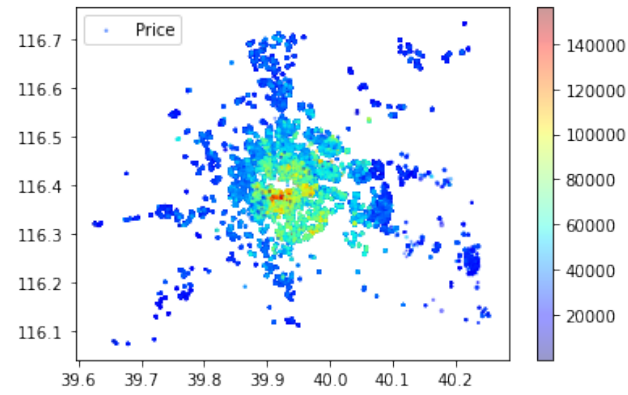


Fig. 6. The housing price distribution of different latitude and longitude.

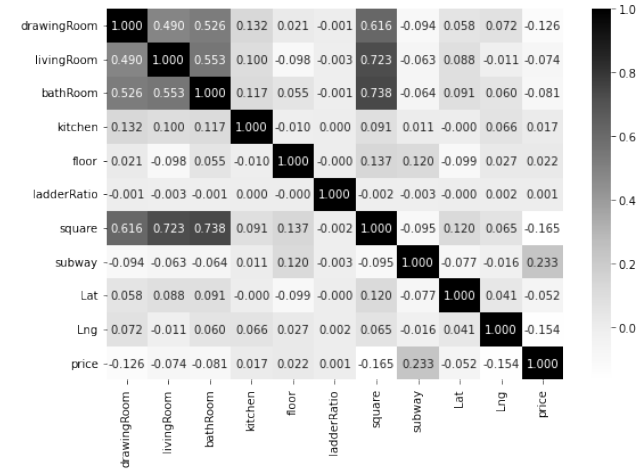


Fig. 7. The corralation of data fields.

period of time, there are 153626 records in Cassandra and the *kd tree* has been trained by data in batch layer. The time window of Spark streaming is set to 30s and the *kd tree* in speed layer has been trained by data in this time span. The housing information [3, 2, 2, 1, 6, 0.5, 102.66, 0, 39.873867, 116.66589] is submitted to the real time and batch service respectively. Both services are deployed in Flask and the housing information is post to corresponding service in JSON format.

Table IV shows the results returned by batch service when $k = 10$. The predicted price is 33194.3 that is the average price of the 10 nearest neighbors. We can also compute the price by *weighted kNN*. The response time of service is 0.001753s that is short enough. The *id* field gives the neighbor index in the 153626 records. So one whole housing information with 26 fields can be obtained by each *id*.

TABLE III
THE COMPONENTS AND FRAMEWORKS IN THE SYSTEM

Frameworks	Functions	Office websites
Flume	Gather and redirect streaming data.	http://flume.apache.org/
Kafka	Cache the data flow from Flume and provide streaming data to Spark streaming system.	http://kafka.apache.org/
Spark streaming	1. Be used in speed layer to get streaming data for real-time computation. 2. Store streaming data as real-time views. 3. Append streaming data to distributed Cassandra database.	https://spark.apache.org/streaming/
Cassandra	Be used in batch layer to store the streaming data and provide batch views.	https://cassandra.apache.org
Flask	1. Train kd trees based on real-time and batch views respectively. 2. Provide web interfaces for both real-time and batch services.	https://flask.palletsprojects.com/

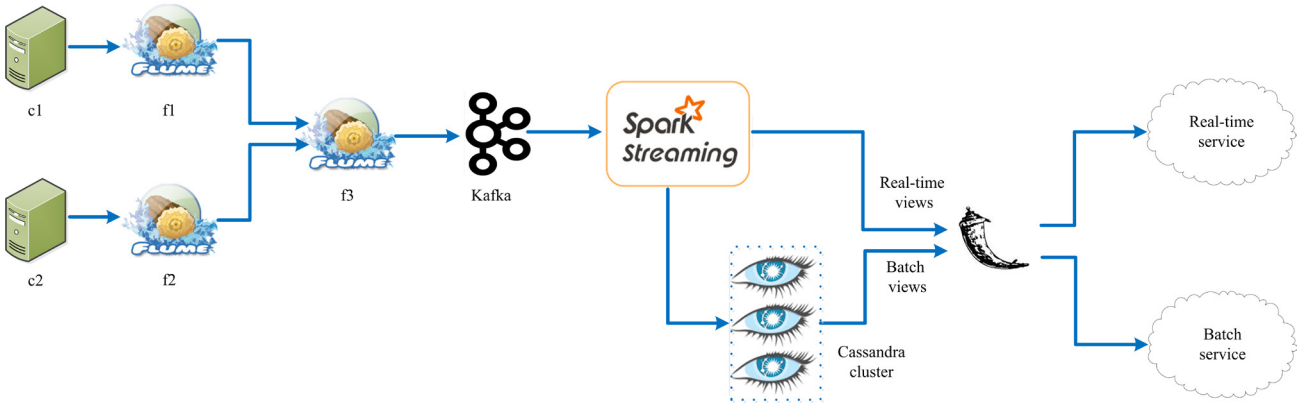


Fig. 8. The structure of price prediction system.

Table V shows the results returned by real-time service when $k = 10$. The predicted price is 34910.6 and the numbers in *id* field are the housing indices in the records of the corresponding time window.

For the same input data the batch and real-time services give different price predictions. The reason is that the *kd tree* in batch service is trained by the all or partial historical data, but the *kd tree* in real-time service is trained by the data in time window. In our example the more training data the greater chance to find out the most nearest neighbors. It can be seen from the distance values in the *distance* column in Table IV and V. It is clear that the 10 neighbors got in batch service are closer to input data than in the real-time service.

So in Lambda architecture we can get more accurate prediction in batch service and get more timely prediction in real-time service that can reflect the trend and fluctuation of price in short term. These two kinds of prices can give us the different perspectives to make decision.

V. RESULTS AND DISCUSSION

In this section the time and space performance of *brute kNN*, *kd tree* and *ball tree* are compared. The results show that *kd tree* is better than other two models in our application. We also compare some other machine learning models with the three *kNN* related models.

Figure 9 shows the memory space of different models after training for different n, here n is the number of records in training data. The values of n are set 20000, 50000, 100000 and 150000. It can be seen that with the growth of n the memory space of different models increases accordingly. And the memory space of *kd tree* and *ball tree* has more significant increase than *brute kNN*. The reason is that besides the basic data information both the *kd tree* and *ball tree* need additional space to store tree structure information.

Figure 10 shows time cost of different models for different n during the training process. It can be seen that the *brute kNN* uses less training time than both *kd tree* and *ball tree*. The time cost of *kd tree* and *ball tree* has significant increase as the growth of n because both tree-based models need more time to construct the tree structures.

So from Fig. 9 and 10 it can be seen that both *kd tree* and *ball tree* need more memory space and training time. But after trained, the tree-based models have shorter prediction time. It can be proved by the following experiment. The testing dataset includes 1000 samples and all models are trained by the same training dataset. When *k=3, 5, 10, 15, 20, 25* we compute the averaging prediction time of 1000 samples on *brute kNN*, *ball tree* and *kd tree* respectively. In Fig. 11 it can be seen that *kd tree* have less prediction time than both *ball tree* and *brute kNN*.

Besides the comparison of space and time cost the evaluation metrics RMSE(Root Mean Squared Error), MAE(Mean Absolute Error) ,R-Squared and MAPE(Mean Absolute Percentage Error) are chosen to evaluate the models. The smaller these metrics the better performance of models. In this paper the k-fold cross-validation technique is used in order to

TABLE IV
THE BATCH SERVICE RESULTS WHEN $k = 10$.

id	bedroom	livingroom	bathroom	kitchen	floor	ladderratio	square	subway	lat	lng	distances
65707	3	2	2	1	6	0.5	105.97	0	39.879143	116.652458	0.158833
151840	3	2	2	1	6	0.5	108.38	0	39.882812	116.661682	0.186459
151820	3	2	2	1	6	0.5	108.70	0	39.882812	116.661682	0.193895
14604	3	2	2	1	6	0.5	110.00	0	39.876873	116.655457	0.222846
14596	3	2	2	1	6	0.333	110.00	0	39.876873	116.655457	0.222846
65726	3	2	2	1	6	0.333	109.92	0	39.879143	116.652458	0.237487
14603	3	2	2	1	6	0.5	110.81	0	39.876873	116.655457	0.242931
112687	3	2	2	1	6	0.5	111.62	0	39.882767	116.669685	0.264404
142464	3	2	2	1	6	0.5	111.65	0	39.883930	116.655563	0.282764
112650	3	2	2	1	6	0.5	92.90	0	39.882767	116.669685	0.284707

TABLE V
THE REAL-TIME SERVICE RESULTS WHEN $k = 10$.

id	bedroom	livingroom	bathroom	kitchen	floor	ladderratio	square	subway	lat	lng	distances
91	3	2	2	1	15	0.3	114.60	0	39.910679	116.594887	1.666115
73	3	3	2	1	4	0.5	144.70	0	39.970848	116.552177	2.611373
59	2	2	1	1	6	0.5	79.88	0	39.934643	116.695198	2.942007
82	2	2	2	1	4	0.028	52.41	0	39.774361	116.512253	3.288918
78	3	2	2	1	6	0.5	129.56	0	40.092632	116.381378	3.437851
41	3	2	2	1	6	0.5	187.16	0	40.110054	116.560165	3.452464
64	3	1	1	1	9	0.5	95.00	0	39.971687	116.544952	3.491309
44	4	2	2	1	6	0.5	158.00	0	40.162060	116.555733	3.619236
84	3	2	2	1	13	0.5	158.93	1	40.082584	116.416153	3.679577
96	3	2	2	1	25	0.375	129.83	0	40.045547	116.430565	3.754350

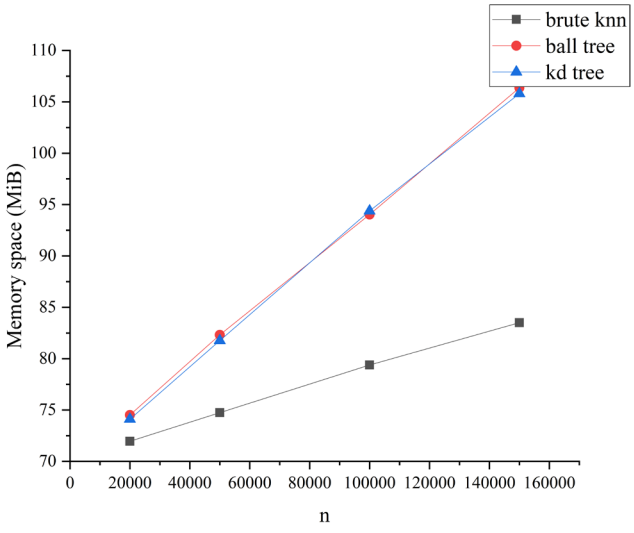


Fig. 9. The memory space of *brute kNN*, *kd tree* and *ball tree* models.

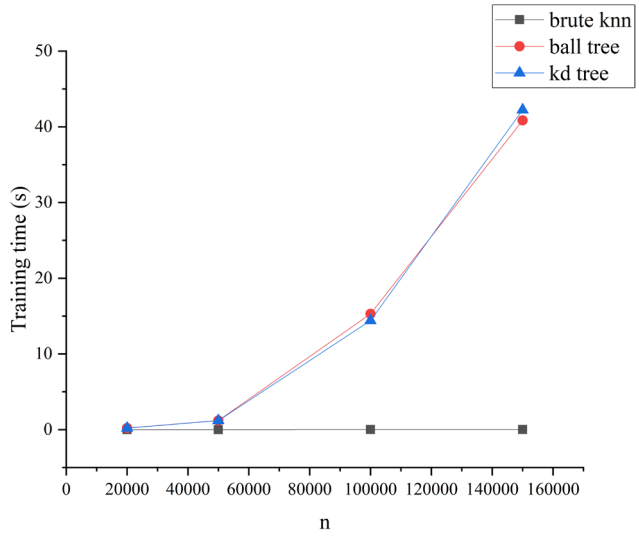


Fig. 10. Training time of of *brute kNN*, *kd tree* and *ball tree* models.

compute the average values of RMSE, MAE, R-Squared and MAPE. The k is set to 5. We compare these metrics between the *brute kNN*, *ball tree* and *kd tree*. Table VI shows the computed results of RMSE, MAE, R-Squared and MAPE of three *kNN* related models for *k=3, 5, 10, 15, 20, 25*. It can be seen that there is no obvious difference of performance in all cases.

In the k-fold cross-validation above, other machine learning models are chosen to complete the same prediction task, including *regression*, *neural network*, *decision tree*, *random forest* and *SVM regression*. The description and parameter information and the average values of RMSE, MAE, R-Squared and MAPE of these models are shown in Table VII. From Table VI and VII we can see that *kNN*-related models have better performance in our application.

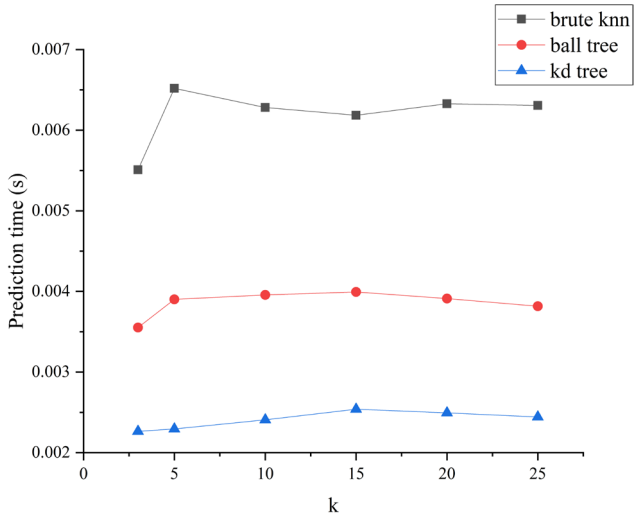


Fig. 11. Predicting time of *brute kNN*, *kd tree* and *ball tree* models.

TABLE VI
THE EVALUATION METRICS FOR DIFFERENT k ON THE *brute kNN*, *ball tree* AND *kd tree* MODELS.

k	Models	RMSE	MAE	R-Squared	MAPE
3	brute kNN	18206.3527	13926.4134	0.4257	364.7942
	ball tree	18202.1706	13926.7102	0.4260	364.7693
	kd tree	18200.8848	13926.0901	0.4260	358.1100
5	brute kNN	17562.3638	13584.7236	0.4656	350.8004
	ball tree	17548.6531	13577.2199	0.4664	346.4767
	kd tree	17553.8938	13579.8747	0.4661	346.4880
10	brute kNN	17129.0225	13382.8892	0.4916	337.2047
	ball tree	17122.4624	13377.2459	0.4920	336.3491
	kd tree	17126.3306	13377.2684	0.4918	335.6577
15	brute kNN	17062.6534	13373.7111	0.4956	347.9658
	ball tree	17061.8768	13377.6418	0.4956	345.4252
	kd tree	17064.7471	13376.5118	0.4955	345.9193
20	brute kNN	17062.9813	13407.4079	0.4956	351.2006
	ball tree	17062.5500	13403.7100	0.4956	352.4062
	kd tree	17067.8417	13410.3385	0.4953	352.5939
25	brute kNN	17086.2347	13437.6975	0.4942	364.1835
	ball tree	17086.8381	13436.8339	0.4942	364.5805
	kd tree	17086.4431	13437.4806	0.4942	364.1689

From these experiments we can see:

1. The *brute kNN*, *kd tree* and *ball tree* have similar evaluation metrics that are better than other machine learning models in our system.

2. Although the *ball tree* and *kd tree* need more memory space and training time than *brute kNN*, the prediction time is less.

3. In our application the *kd tree* has less prediction time than *ball tree*. So we choose *kd tree* as the final model.

VI. CONCLUSION

The accurate and timely housing price prediction can help individuals and enterprises make reasonable decisions. In this paper we design and implement a system for the prediction of second-hand housing price based on Lambda Architecture. This system can provide housing price prediction by both historical and real-time data, so it can provide two different perspectives on prediction. By analysis of the space and time performance and metrics comparison with other models, the *kd tree* is used as prediction model in both batch layer and speed layer. Besides price prediction, the other benefit of using the *kd tree* is that the nearest k neighbors can be used as a housing recommending list. The system can also be used in other applications to provide prediction services.

Other architectures, such as Kappa and Alarea, can also complete the same task. In the future, we will consider porting the existing system to these two frameworks and compare their implementation performance with that of Lambda architecture. Alarea is very attractive because its techniques stack is very similar to that of our system, and it combines the advantages of Lambda and Kappa architectures.

ACKNOWLEDGMENT

This research was funded by the School-level Scientific Research Project of Harbin University of Commerce (grant number 18XN021), the Heilongjiang Provincial Natural Science Foundation of China (grant number LH2019F044) and the Heilongjiang Provincial Key Laboratory of Electronic Commerce and Information Processing.

REFERENCES

[1] H. García-González, D. Fernández-Álvarez, J. E. Labra-Gayo, and P. Ordóñez de Pablos, "Applying big data and stream processing to the real estate domain," *Behaviour & Information Technology*, vol. 38, no. 9, pp. 950–958, 2019, DOI: 10.1109/TMC.2019.2944829.

[2] A. A. Munshi and Y. A.-R. I. Mohamed, "Data lake lambda architecture for smart grids big data analytics," *IEEE Access*, vol. 6, pp. 40463– 40 471, 2018, DOI: 10.1109/access.2018.2858256.

TABLE VII
THE EVALUATION METRICS OF OTHER MODELS.

Models	Description	RMSE	MAE	R-Squared	MAPE
regression	Ordinary least squares linear regression with the intercept term.	22572.6282	17639.8669	0.1172	373.0176
neural network	Multilayer perceptron. It has two hidden layers. The first hidden layer contains 5 neurons. The second hidden layer contains 3 neurons. It uses L2 regularization.	22212.5016	17321.2284	0.1451	374.2383
decision tree	It uses mean squared error as feature selection criterion.	20340.7861	15163.8469	0.2831	343.9726
random forest	It uses mean squared error as feature selection criterion. The number of trees in the forest is set to 100.	17193.4087	13176.5261	0.4878	356.1482
SVM regression	Support vector regression with linear kernel.	25372.5650	18480.4802	-0.1154	294.4954

[3] M. Gribaudo, M. Iacono, and M. Kiran, "A performance model- ing framework for lambda architecture based applications," *Future Generation Computer Systems*, vol. 86, pp. 1032–1041, 2018, DOI: 10.1016/j.future.2017.07.033.

[4] O. Debauche, S. A. Mahmoudi, N. De Cock, S. Mahmoudi, P. Manneback, and F. Lebeau, "Cloud architecture for plant phenotyping research," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 17, p. e5661, 2020, DOI: 10.1002/cpe.5661.

[5] T. Numnonda, "Areal-time recommendation engine using lambda architecture," *Artificial Life and Robotics*, vol. 23, no. 2, pp. 249–254, 2018, DOI: 10.1007/s10015-017-0424-8.

[6] I. Ganchev, Z. Ji, and M. O'Droma, "A conceptual framework for building a mobile services' recommendation engine," in *2016 IEEE 8th International Conference on Intelligent Systems (IS)*. IEEE, 2016, pp. 285–289, DOI: 10.1109/IS.2016.7737435.

[7] P. Krajsic and B. Franczyk, "Lambda architecture for anomaly detection in online process mining using autoencoders," in *International Conference on Computational Collective Intelligence*. Springer, 2020, pp. 579–589, DOI: 10.1007/978-3-030-63119-2-47.

[8] D. Vajda, A. Pekár, and K. Farkas, "Towards machine learning-based anomaly detection on time-series data," *INFOCOMMUNICATIONS JOURNAL*, vol. 13, no. 1, pp. 35–44, 2021, DOI: 10.36244/ICJ.2021.1.5.

[9] U. Suthakar, L. Magnoni, D. R. Smith, and A. Khan, "Optimised lambda architecture for monitoring scientific infrastructure," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1395–1408, 2018, DOI: 10.1109/tpds.2017.2772241.

[10] R. M. Croce and D. R. Haurin, "Predicting turning points in the housing market," *Journal of Housing Economics*, vol. 18, no. 4, pp. 281–293, 2009, DOI: 10.1016/j.jhe.2009.09.001.

[11] P. Dua, "Analysis of consumers' perceptions of buying conditions for houses," *The Journal of Real Estate Finance and Economics*, vol. 37, no. 4, pp. 335–350, 2008, DOI: 10.1007/s11146-007-9084-0.

[12] C. Jin, G. Soydemir, and A. Tidwell, "The u.s. housing market and the pricing of risk: Fundamental analysis and market sentiment," *Journal of Real Estate Research*, vol. 36, no. 2, pp. 187–220, 2014, DOI: 10.1080/10835547.2014.12091390.

[13] L. Zhang, J. Zhao, and K. Xu, "Emotion-based social computing platform for streaming big-data: Architecture and application," in *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*. IEEE, 2016, pp.1–6, DOI: 10.1109/ICSSSM.2016.7538620.

[14] J. Smailović, M. Grčar, N. Lavrač, and M. Žnidaršič, "Stream-based active learning for sentiment analysis in the financial domain," *Information sciences*, vol. 285, pp. 181–203, 2014, DOI: 10.1016/j.ins.2014.04.034.

[15] R. Schulz, M. Wersing, and A. Werwatz, "Automated valuation modelling: a specification exercise," *Journal of Property Research*, vol. 31, no. 2, pp. 131–153, 2014, DOI: 10.1080/09599916.2013.846930.

[16] M. Steurer, R. J. Hill, and N. Pfeifer, "Metrics for evaluating the performance of machine learning based automated valuation models," *Journal of Property Research*, pp. 1–31, 2021, DOI: 10.1080/09599916.2020.1858937 .

[17] Y. Chen, L. Zhou, Y. Tang, J. P. Singh, N. Bouguila, C. Wang, H. Wang, and J. Du, "Fast neighbor search by using revised kd tree," *Information Sciences*, vol. 472, pp. 145–162, 2019, DOI: 10.1016/j.ins.2018.09.012.

[18] L. Huand S. Nooshabadi, "High-dimensional image descript or matching using highly parallel kd-tree construction and approximate nearest neighbor search," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 127–140, 2019, DOI: 10.1016/j.jpdc.2019.06.003.

[19] L. Wang, B. Yang, Y. Chen, X. Zhang, and J. Orchard, "Improving neural-network classifiers using nearest neighbor partitioning," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2255–2267, 2016, DOI: 10.1109/TNNLS.2016.2580570.

[20] A. A. Neloy, M. S. Oshman, M. M. Islam, M. J. Hossain, and Z. B. Zahir, "Content-based health recommender system for icu patient," in *International Conference on Multi-disciplinary Trends in Artificial Intelligence*. Springer, 2019, pp. 229–237, DOI: 10.1007/978-3-030-33709-4_20.

[21] B. R. Hiraman et al., "A study of apache kafka in big data stream processing," in *2018 International Conference on Information, Commu- nication, Engineering and Technology (ICICET)*. IEEE, 2018, pp. 1–3, DOI: 10.1109/ICICET.2018.8533771.

[22] D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, "A comparison on scalability for batch big data processing on apache spark and apache flink," *Big Data Analytics*, vol. 2, no. 1, pp. 1–11, 2017, DOI: 10.1186/s41044-016-0020-2.

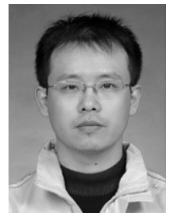
[23] J. Kreps, "Questioning the lambda architecture." 2014. [Online]. Available: https://www.oreilly.com/radar/questioning-the-lambda-architecture/

[24] J. Warren and N. Marz, *Big Data: Principles and best practices of scalable realtime data system*s. Simon and Schuster, 2015.

[25] J. Maillo, S. García, J. Luengo, F. Herrera, and I. Triguero, "Fast and scalable approaches to accelerate the fuzzy k-nearest neighbors classifier for big data," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 5, pp. 874–886, 2019, DOI: 10.1109/TFUZZ.2019.2936356.

[26] J. Maillo, J. Luengo, S. García, F. Herrera, and I. Triguero, "A preliminary study on hybrid spill-tree fuzzy k-nearest neighbors for big data classification," in *2018 IEEE international conference on fuzzy systems (fuzz-IEEE). IEEE, 2018*, pp. 1–8, DOI: 10.1109/FUZZ-IEEE.2018.8491595.

[27] Q. Qiu, "Housing price in beijing." 2018. [Online]. Available: https://www.kaggle.com/ruiqurm/lianjia/version/2



Qinghe Pan is currently an associate professor at the School of Computer and Information Engineering of Harbin University of Commerce. His current research interests focus on big data techniques, data mining and machine learning algorithms. He currently teaches in many areas such as Hadoop and Spark architectures, distributed systems and data mining methods.

Zeguo Qiu is currently a professor at Harbin University of Commerce. He received PhD in Management Science and Engineering from the Dongbei University of Finance and Economics, Liaoning, China, in 2013. In 2015, he joined the Northeast Asia Service Outsourcing Postdoctoral Workstation and worked on the topic of management decision, information system re-engineering, enterprise technology innovation and e-commerce. His research interests include management decision making, information systems and e-commerce.

Yaoqun Xu received the B.S. degree in mathematics from Jilin University, Changchun, China, in 1993, the M.S. degree in mathematics from the Harbin Institute of Technology, Harbin, China, in 1997, and the Ph.D. degree in navigation, guidance, and control from Harbin Engineering University, Harbin, in 2002. He is currently a Professor with the College of Computer and Information Engineering, Harbin University of Commerce. His current research interests include chaotic dynamics, neural networks, and intelligent optimization and decision.

Guilin Yao is currently an associate professor in the School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China. His research interests include image processing, computer vision, and machine learning.