Implementing AI as Cyber IoT Devices: The House Valuation Example
Yun-Wei Lin, Yi-Bing Lin, Fellow, IEEE, Chun-You Liu, Jiun-Yi Lin, and Yu-Lin Shih

Abstract—Internet of Things (IoT) has been widely utilized with Artificial Intelligence (AI). However, it requires substantial effort to integrate AI and big data with IoT. To mitigate this problem, AItalk was proposed. By treating AI as a cyber IoT device, we do not need to write code of the AI mechanism in the network applications, as traditional AI-based IoT applications did. This paper describes how the AI tools such as scikit-learn and TensorLayer are accommodated as cyber IoT devices in AItalk, and extends AItalk for non-IoT applications. We use house valuation as an example to show how AItalk can flexibly include the factors that have significant impact on the house price. We show that by adding extra features other than housing profile features, the accuracy for the prediction (valuation) can be improved by 38%. We also investigated the communication overhead of the distributed AItalk structure, which is 3.7% for the computation of one house price valuation.

Index Terms—artificial intelligence, cyber actuator, cyber sensor, house valuation, Internet of Things, machine learning

I. INTRODUCTION

Internet of Things (IoT) has been widely utilized in various fields (e.g., money flow, traffic management, logistics flow, smart grid, people flow, health monitoring, interactive art design, and smart home). Many smart applications have been deployed by integrating IoT, big data and Artificial Intelligence (AI). However, such integration results in significant programming effort.

To simplify this integration issue, AItalk [1] was proposed to conveniently include a machine learning open-source tool [2] in the existing IoT applications. The IoT platform considered in AItalk is called IoTtalk [3], and the integration has shown that AItalk can rapidly extend existing IoT applications into AI-based smart applications [1] [4]. An important concept of AItalk is the introduction of “cyber IoT device” that can be manipulated in the IoT service platform (i.e., IoTtalk). A major contribution of this paper is to design and implement both the house/environment factors and the AI model as cyber IoT devices such as HdataBank and HouseValuation for the house valuation application.

Specifically, we demonstrate that every AI application can be transformed to an “IoT application” in AItalk even if no real IoT devices are involved. The idea is to treat the input features as “sensors”, and the output predictions and statistics as “actuators”. The AI mechanism itself is considered as an IoT device that consists of both inputs and outputs. Through the concept of IoT, an AI application is decomposed into distributed components interacting through the IoT technology, which can be built with much better efficiency. In our approach, real-time data are conveniently accommodated without extra programming effort.

In this paper, Sections II and III describe how the AI tools such as scikit-learn [5] and TensorLayer [6] are accommodated as cyber IoT devices in AItalk. Sections III-VI show how the IoTTalk engine is used to build an AI application for house valuation, which involves non-real IoT (cyber) devices. Section VII discusses the overhead of AItalk.

II. THE AITALK PLATFORM

Fig. 1 shows the software blocks of IoT-based machine learning for an intelligent farm application, where a farmer (Fig. 1 (b)) uses a controller (e.g., a smartphone) to control actuators (sprayers for water, fertilizer or pesticide; Fig. 1 (c)). The sprayers are intelligently controlled by sensors, e.g., for humidity, CO2, temperature and so on [4] (see Feature, in Fig. 1 (a)) that produce real-time data.

Fig. 1. Machine learning for intelligent farm application

When the sensor data (Fig. 1 (1)) arrive, the machine learning device first performs feature extraction (Fig. 1 (5)). Specific extraction methods are selected to extract the sensor data characteristics to form a feature vector (Fig. 1 (6)). The feature vectors as well as the farmer’s labels (Fig. 1 (2)) are used for training (Fig. 1 (7)). The hyper-parameters (Fig. 1 (8)) are set up to generate prediction results for multiple machine learning algorithms. Through an appropriate ensemble method...
(Fig. 1 (9)), the best prediction result (Fig. 1 (3)) is used to instruct the actuators (e.g., water spraying). The results are also used to improve the accuracy of prediction by conducting validation (Fig. 1 (11)) that provides better setup for the hyper-parameters (Fig. 1 (12)). Useful statistics (Fig. 1 (10)) are displayed (Fig. 1 (d)) for the application developer to adjust the machine learning model.

By implementing the open source tools including scikit-learn [5], TensorFlow [6] and Flux [28] as cyber IoT devices, Fig. 2 shows that in Altalk, the above machine learning mechanism can be transparently included in an existing IoT application (Fig. 2 (f)) created with IoTtalk (Fig. 2 (h)) [3, 4].

To interact with the IoTtalk engine (Fig. 2 (h)), Altalk installs every IoT device a software module called device application (DA; see Fig. 2 (a)-(e)). The DA sends and/or receives the IoT information to/from the IoTtalk engine. The web-based GUI easily sets up the machine learning mechanism, and establishes AI-based interactions between the sensors and the actuators.

The developer can access the Altalk GUI (Fig. 2 (g)) through any computing device with web browser (e.g., a smart phone; see Figs. 4 and 5) so that he/she can conveniently configure an AI application by creating, and setting up device features, functions and connection. IoTtalk introduces the concept of “cyber” IoT device (such as animation) besides the “physical” IoT devices (such as water spray). Following this concept, Altalk implements the machine learning mechanism in Fig. 1 as a cyber IoT device called ML_device (Fig. 2 (i)) that consists of two parts. The DA (Fig. 2 (e)) is responsible for interaction with the IoTtalk engine, and the machine learning module (Fig. 2 (j)) includes the components in Fig. 1 (5)-(12).

The DA implements the DFs to interact with the IoTtalk engine. For example, ML_device has both IDFs and ODFs, and is therefore represented by both an input and an output devices. In the input part of the DA, the prediction produced by Fig. 1 (9)) is sent to Result-IDF (Fig. 2 (3)) to control the actuators (Fig. 2 (c)) through the IoTtalk engine. The input device of ML_device also produces statistics that are sent to Stats-I (Fig. 2 (4)). The IoTtalk engine then sends these statistics to the Statistics device (Fig. 2 (d)), which are typically displayed for tuning the hyper-parameters. In the output part of the DA for ML_device, the Label-O ODF (Fig. 2 (2)) receives the labels from the remote control (Fig. 2 (b)), and the Feature-O ODFs (Fig. 2 (1)) receive the data from the sensors (Fig. 2 (a)) through the IoTtalk engine, and pass the data to feature extraction (Fig. 1 (5)).

III. THE DEVICE APPLICATION FOR MACHINE LEARNING

This section shows how to port AI tools into Altalk. Fig. 3 illustrates the device application (DA) and ML Module that implement AI tools as cyber IoT devices, where ML_device (Fig. 2 (e), (i) and (j)) is illustrated in Fig. 3 (e), (i), and (j). In this figure, every arrow line represents a step where the arrow indicates the direction of the step. The crossing lines without bullets (Steps 13 and 17, Steps 14 and 17, and Steps 15 and 17) represent independent steps.

Steps 1-3: After the user has set up the machine learning configuration through the ML setup webpage (to be elaborated in Fig. 8), the ML Web Server (Fig. 3 (a)) saves the configuration in the database DB (Fig. 3 (b)). Then the server invokes init() of MsgHandler (Fig. 3 (c)) to retrieve the ML configuration information from DB.

Steps 4-6: init() invokes register() that connects ML_device to IoTtalk engine (Fig. 3 (h)) through HTTP.

Steps 7 and 8: init() invokes historical_data() to read the historical data that are sent to feature_extract().
Steps 28-30: The user exits AItalk through the ML setup.

Steps 25: Through model fitting with enhanced hyper-parameters,

Steps 22-24: Both the historical data and the real-time data are sent to feature_extract() and then train() for training. Based on the setup saved in DB, train() performs, for example, k-fold cross-validation by randomly partitioning the raw data set into k groups of approximately equal size. One group is treated as a validation set, and the k-1 groups are used as the training set. According to the feature extraction setup stored in DB, feature_extract() performs the task in Fig. 1 (5) on both the training and the validation data sets by invoking feature extraction functions in the decomposition class of the scikit-learn module (Fig. 3 (f)), including PCA(), Sparse_coding(), Standardize(), etc.

Steps 14 and 15: After the feature sets are generated at Step 13, train() uses the training feature set to perform the task in Fig. 1 (7), which invokes the fit functions in scikit-learn. For discrete features, the discrete fit functions are executed with inputs “features” (came from Feature-O) and “labels” (came from Label-O). The discrete fit functions in scikit-learn include SVC.fit(), kNN.fit(), DecisionTree.fit(), and so on. At the end of this step, the machine learning model is fit by the training feature set.

Steps 16-18: The fit machine learning model is used by predict() to invoke the predict functions in scikit-learn including SVC.predict(), kNN.predict(), DecisionTree.predict(), and so on. These functions use the validation feature set as input to produce the predicted results. Then the predicted results and the corresponding labels are sent to validate() for comparison to produce the statistics (Fig. 1 (11)).

Steps 19-21: The validated results are used to find the better hyper-parameters of the machine learning algorithms from a set of hyper-parameter combination. To perform this task (Fig. 1 (12) and (13)), tune_hyper_param() invokes the functions obtained from the hyper-parameter turning classes in scikit-learn, for example, GridSearchCV() from class model_selection (for grid search), bayesian_optimization() from class Gaussian_process (for Bayesian optimization), and so on. In this way, we obtain better hyper-parameters that are sent to train() to fit the machine learning models better.

Steps 22-24: Through model fitting with enhanced hyper-parameters, validate() finds the best prediction results and the statistics that are sent to the IoTtalk engine through the Stats-I IDF.

Step 25: After the machine learning models have been trained and tested by the training and the validation feature sets, we can receive the data without labels (i.e., the step path (26)->(10)->(11)->(12)->(16), and predict() produces the predicted results that are sent to the IoTtalk engine through the Result-I IDF (i.e., the step path (25)->(23)->(24)->(27)).

Steps 28-30: The user exits AItalk through the ML setup webpage by invoking deregister().

Note that both train() at Step 16 and predict() at Step 17 can run multiple machine learning models simultaneously, and use the ensemble method to find the best results. If ML_device connects to TensorLayer, then the Convolutional Neural Network (CNN) is used, and the following steps are affected. At Step 15, utils.fit() of the tensorlayer class is used, which includes several hyper-parameters where session is the declared tensorflow session, network is the neural network to be trained, cost is the cost function of CNN (e.g., cost.cross_entropy() of the tensorlayer class), features are the input features, labels are target labels, batch_size is the number of features that will be propagated through the neural network in each training process, and n_epoch is the iteration number that CNN will work through the entire training dataset. The hyper-parameter train_op is the optimizer for training (e.g. train.AdamOptimizer() of the tensorflow class), which automatically performs Steps 19-21. At Step 17, predict() invokes both utils.predict() and utils.test() in the tensorlayer class, where utils.test() automatically performs validation in Step 18.

In Fig. 3 (f) and (g), the APIs/libraries for TensorLayer and scikit-learn are hidden from the developer and he/she can transparently use these tools without touching their tedious APIs/libraries. From the above description, it is clear that AItalk can seamlessly include AI capability to the existing IoT applications. Examples include smart home [1] and smart farm [4]. The AI mechanism for an application without involving real IoT devices can also be effectively created in AItalk. The idea is to treat the features and the label of the application as the cyber sensors, and the predicted results and the statistics for machine learning performance are sent to a display actuator (a cyber output device). One advantage of this “IoT approach” is that a complicated AI application can be decomposed into simple modules, where the implementation is easy to understand, modify and debug.

IV. AITALK FOR NON-IOT APPLICATION: HOUSE VALUATION AND PRICING

This section shows how AItalk is used to develop a non-IoT application, specifically, a house valuation system. Our study indicates that this valuation system is comparable with that of proprietary state-of-the-art real estate appraisal systems worldwide [7]. AItalk can easily accommodate various features beyond basic housing features [9] as well as IoT data (e.g., cyber sensors, and the predicted results and the statistics for machine learning performance are sent to a display actuator (a cyber output device). One advantage of this “IoT approach” is that a complicated AI application can be decomposed into simple modules, where the implementation is easy to understand, modify and debug.

IV. AITALK FOR NON-IOT APPLICATION: HOUSE VALUATION AND PRICING

This section shows how AItalk is used to develop a non-IoT application, specifically, a house valuation system. Our study indicates that this valuation system is comparable with that of proprietary state-of-the-art real estate appraisal systems worldwide [7]. AItalk can easily accommodate various features beyond basic housing features [9] as well as IoT data (e.g., cyber sensors, and the predicted results and the statistics for machine learning performance are sent to a display actuator (a cyber output device). One advantage of this “IoT approach” is that a complicated AI application can be decomposed into simple modules, where the implementation is easy to understand, modify and debug.

IV. AITALK FOR NON-IOT APPLICATION: HOUSE VALUATION AND PRICING

This section shows how AItalk is used to develop a non-IoT application, specifically, a house valuation system. Our study indicates that this valuation system is comparable with that of proprietary state-of-the-art real estate appraisal systems worldwide [7]. AItalk can easily accommodate various features beyond basic housing features [9] as well as IoT data (e.g., cyber sensors, and the predicted results and the statistics for machine learning performance are sent to a display actuator (a cyber output device). One advantage of this “IoT approach” is that a complicated AI application can be decomposed into simple modules, where the implementation is easy to understand, modify and debug.
and offline house related information. We use the Altalk GUI to build HdataBank and ML_device called “HouseValuation”. By connecting real sensor devices with HdataBank, Altalk seamlessly integrates IoT data into the AI applications. When the ML_device item (Fig. 4 (2)) is clicked from the “Model” pulldown list (Fig. 4 (1)), the device feature setup window pops up (Fig. 4 (3)). We first select three IDFs including Result-I, FIA-I and MER-I (Fig. 4 (4)). Then we select ODFs for 7 Features and Label-O (Fig. 4 (5)). After saving the setups (Fig. 4 (6)), the input and the output device icons for ML_device are displayed in the GUI window with the name HouseValuation (Fig. 5 (b) and (c)). This setup process automatically creates the DA of ML_device.

The input device of HouseValuation is an icon placed on the left-hand side (Fig. 5 (c)). The HouseValuation output device is another icon placed on the right-hand side (Fig. 5 (b)). One or more small icons are placed inside a device icon to represent the DFs. The HouseValuation device has 3 IDFs and 8 ODFs. When a device icon is placed in the GUI window, a network application corresponding to the physical or cyber device is automatically created by the IoTtalk engine.

The input and the output devices for “HdataBank” (Fig. 5 (a) and (d)) are also created following the same selection procedure (Fig. 4 (7)). The HdataBank input device has 57 feature IDFs and one label IDF to be connected to the ODFs of ML_device. The IDFs of the HdataBank input device are city and house related information that will be elaborated in Section V. For better readability, Altalk provides a compound mechanism that allows input devices to be grouped within a compound device. Therefore, 57 feature IDFs are grouped in the HdataBank compound device with 7 input devices, which are represented as “IDFs” in Fig. 5 (1)-(7). The 58-th IDF is HousePrice-I that provides actual house prices to serve as the labels for the prediction (Fig. 5 (8)).

To connect an IDF to an ODF, we only need to drag a line between the corresponding IDF icon and the ODF icon in the window, and IoTtalk automatically creates the network application to handle the interaction between them. For example, the first 7 IDFs of the HdataBank output device are connected to the feature ODFs of HouseValuation through Joins 1-7. Join 8 connects HousePrice-I with Label-O. The HdataBank output device displays the predicted house price sent from HouseValuation to HousePrice-O (Fig. 5 (9)) through Join 9, the feature importance analysis (FIA-O; see Fig. 5 (10)) through Join 10 (to be elaborated in Fig. 9), and the median error rate (MER-O; see Fig. 5 (11)) through Join 11. Detailed setups of HouseValuation will be elaborated in Section VI.

V. CREATION OF THE HDATA_BANK IDFS

This section describes the IDFs of the HdataBank device based on hedonic pricing, the theoretical foundation of our house valuation application. The hedonic house pricing model identifies both internal or structure factors (i.e., house profile features) and external or environmental factors affecting it (i.e., non-house profile features). The reader is referred to [19] for hedonic house pricing, and [22], [23] for environmental effect on house prices. Choice of function forms of hedonic pricing can be found in [20], [21], and [24]. Several empirical studies have been conducted, including effects of locations on house prices [25], effects of neighborhood attributes on house prices [26] and school quality effect [27]. In our study, there are two groups of internal factors: Basic Estate Information (Fig. 6 (1)) and House Layout Information (Fig. 6 (2)). There are five groups of external factors: Basic City/County Information (Fig. 6 (3)), Economy Information (Fig. 6 (4)), Geography and Weather Information (Fig. 6 (5)), Transportation Information device (Fig. 6 (6)) and Life Function Information (Fig. 6 (7)). Through the Altalk GUI, we can easily implement hedonic pricing model to accommodate various HdataBank IDFs that may affect housing valuation including temperatures, humidity, rainfall and PM2.5 data produced from the government databases or real-time IoT devices [3], occurrences of sensible earthquakes, the distribution of geographical faults [11] and the distribution of soil liquefaction [12]. In the external factor groups, Economy Information, Geography and Weather Information, Transportation Information device, and Life Function Information are neighborhood features. By considering these neighborhood features, we estimate house...
prices with the comparable transactions, i.e., similar house sales recently transacted in the close neighborhood. These IDFs are especially important for countries facing relatively high economic risks from multiple hazards (e.g., Japan and Taiwan). The data sources for the IDFs of HdataBank consist of the government open data [9] and the real-time crawl data that cannot be downloaded directly. All data are grouped in 7 input devices in Altalk. The DAs for these devices are the same (Fig. 2 (a)), and have already been implemented in Altalk. To create any of the seven devices in Fig. 6, we only need to plug in the corresponding data sources (government databases, historical files, or real sensors) into the DA.

A major data source is the online database of Ministry of the Interior (MoI), Taiwan [13]. This public database provides detailed information for every property, including the land area, the construction area and the parking areas of the property. The basic housing data also includes the county/city (where the house is located), the housing type, the address, the floor space, the lowest and the highest floors, the total number of floors, the number of bedrooms, the number of living rooms, the number of bathrooms, the age of the house, and the latitude and longitude of the house.

The MoI database provides month information. Real estate economists usually include year of sale to control the unobserved effect in a particular year. In addition, house sales have seasonal effect. We found that in Taiwan, the month feature does not affect the house price. Instead, it affects the sale transactions. The highest transaction months are March, April, and December, and the lowest transaction months are August and September. Most important of all, the database provides the price information and the transaction details that will serve as the labels in our system. The online database contains 523,166 transaction records collected from July 2016 to September 2019. The data cover all counties and cities in Taiwan, which are accessed by the first two of the following seven input devices of HdataBank described below.

- The Basic City/County Information device (Fig. 6 (1)) includes 9 IDFs, where the most important two IDFs are the house age and the land area.
- The House Layout Information device (Fig. 6 (2)) includes 9 IDFs, and the most important two IDFs are the bedroom number and the lowest floor.
- The Basic City/County Information device (Fig. 6 (3)) includes 7 IDFs. The most important two IDFs are the number of empty houses and the number of new buildings.
- The Economy Information device (Fig. 6 (4)) includes 8 IDFs, where the most important two IDFs are the house price to income ratio and the income.
- The Geography and Weather Information device (Fig. 6 (5)) includes 9 IDFs, and the most important two IDFs are Type 1 fault and Type 2 fault.
- The Transportation Information device (Fig. 6 (6)) includes 3 IDFs. The most important two IDFs are Taiwan railway stations and the Taipei MRT stations.
- The Life Function Information device (Fig. 6 (7)) includes 4 IDFs, where the most important two IDFs are the locations for post offices and high schools. It is interesting to note that in Taiwan, the post offices also serve as the banks, and therefore become an important life function.

To obtain the distances between the house and locations for fault, transportation and life function, Altalk uses Google API to obtain the latitudes and the longitudes of the locations for distance calculation. The above computation is achieved at the “circles” of Joins 5, 6 and 7. IoTtalk allows creation of function at a Join circle. The data sent from the IDFs connected to the circle are the inputs to the function. In each circle of Joins 5-7, both the locations of the house and locations of specific targets are sent to the function to compute the distances based on the Haversine formula. Details of the Join function creation can be found in [3]. Note that in every Join circle of Altalk, there is a default function to automatically remove outliers and to fill missing data.

In HdataBank, HousePrice-I is attached to a “sensor” that produces the actual prices of property transactions collected from the database of MoI. The HdataBank includes house sales data from all counties and cities. The prediction process is conducted separately for each single city.

VI. HOUSEVALUATION SETUP

This section describes HouseValuation setup. The configuration of ML_device in Fig. 7 is set up by filling the ML setup webpage illustrated in Fig. 8. Note that same parameters of different AI tools are mapped to the same graphical icon representation in the ML setup webpage. Therefore, the developer can set up the applications with different AI tools in the same way through this webpage without worrying about the tedious setup details for these tools. In this example, by clicking the %a icon at the top left corner of the HouseValuation icon (Fig. 5 (b)), the ML setup window (Fig. 8) pops up. For the description purpose, we use the 7 input devices to represent the IDFs in Fig. 6, and we say “7 IDFs” instead of “57 IDFs”. The functional blocks in Fig. 7 are labeled to indicate the same setup steps in Fig. 8.
Steps 2-4. Feature Extraction: Two stages are selected for feature extraction. The stage number is “2” in the ML setup webpage (Fig. 8 (2)). Stage 0 includes seven IDFs (Fig. 8 (3)), which serve as the inputs of Stage 1 (Fig. 8 (4)). Stage 1 is a $7 \times 7$ matrix with the “standardization” extraction function. The diagonal boxes are checked, which means that Feature $i$ of Stage 0 is connected to Feature $i$ of Stage 1 (for $1 \leq i \leq 7$). Note that the scales of these input features are different, and must be standardized. Currently, Altalk implements all feature extraction algorithms in [5] including PCA, MFCC, FFT, Bypass, Scaling, and so on.

Step 5. Algorithm Selection: There are three machine learning categories in Altalk: regression, classification, and user defined. Like the feature extraction stages, the machine learning algorithms can be configured as a multi-stage network. In the HouseValuation example, we initially selected several machine learning algorithms in the regression category to build the models, including conventional methods such as Linear Regression, Lasso, Ridge, ElasticNet, K-Nearest Neighbors, Decision Tree, and the advanced methods such as Random Forests, Extra Trees [14], XGBoost [15], LightGBM [16] and CatBoost [17]. In the training process, the median error rates (MERs) for these algorithms are calculated, which are listed in Table 2. For every algorithm, the MER is computed from $n$ samples $(X_1, y_1), (X_2, y_2), ..., (X_n, y_n)$. Specifically, $X_i$ is used by the algorithm under test to produce the predicted house price $\hat{y}_i$, where $y_i$ is the ground truth of $y_i$. Therefore,

$$MER = \frac{\text{median}_{i = 1, 2, \ldots, n} \left(\frac{|y_i - \hat{y}_i|}{y_i} \times 100\right)}$$

Altalk produces the MERs (Fig. 7 (c)) and lists them in Table 2. In this table, MER1s are the median error rates for the models using all features listed in Fig. 6 (1)-(7), and MER2s are the median error rates for those using only housing profile (Fig. 6 (1) and (2)). The developer selects top five algorithms to be used in the final implementation for HouseValuation. Among them, Random Forests are well-known tree-based ensemble learning approaches. Extra Trees are a variant of random forests. XGBoost, LightGBM, and CatBoost are state-of-the-art algorithms modified from Gradient Boosting Decision Tree. These algorithms have performed well in recent competitions on Kaggle [10]. Finally, Altalk configures one-stage matrix (Fig. 8 (5)) with the best 5 regression algorithms selected from Table 2.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MER1</th>
<th>MER2</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>33.170%</td>
<td>33.170%</td>
<td>0.000%</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>18.328%</td>
<td>22.311%</td>
<td>17.852%</td>
</tr>
<tr>
<td>Lasso</td>
<td>19.375%</td>
<td>22.392%</td>
<td>13.474%</td>
</tr>
<tr>
<td>Ridge</td>
<td>18.349%</td>
<td>22.314%</td>
<td>17.769%</td>
</tr>
<tr>
<td>ElasticNet</td>
<td>19.274%</td>
<td>22.327%</td>
<td>13.674%</td>
</tr>
<tr>
<td>K-Nearest Neighbors</td>
<td>13.541%</td>
<td>14.004%</td>
<td>3.306%</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>9.114%</td>
<td>11.939%</td>
<td>23.662%</td>
</tr>
<tr>
<td>Random Forests</td>
<td>6.623%</td>
<td>9.225%</td>
<td>28.206%</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>6.777%</td>
<td>9.780%</td>
<td>30.706%</td>
</tr>
<tr>
<td>XGBoost</td>
<td>6.964%</td>
<td>10.544%</td>
<td>33.953%</td>
</tr>
<tr>
<td>LightGBM</td>
<td>7.040%</td>
<td>10.279%</td>
<td>31.511%</td>
</tr>
<tr>
<td>CatBoost</td>
<td>7.754%</td>
<td>12.657%</td>
<td>38.737%</td>
</tr>
<tr>
<td>Average Ensemble</td>
<td>6.461%</td>
<td>9.536%</td>
<td>32.246%</td>
</tr>
</tbody>
</table>

Step 6. Ensemble: Since we use multiple machine learning algorithms at Step 5, the ensemble step must be set up to find the best result based on the predictions of these algorithms. Ensemble learning [18] achieves better prediction result as compared to any of the individual algorithms. In Fig. 8 (6), the average ensemble method is used to compute the mean of the results for all selected machine learning algorithms, which “averages out” various errors of the models. Through Join 9 in Fig. 5, this result is sent from Result-I of HouseValuation (Fig. 7 (a)) to Result-O of HdataBank for display. ML_device also produces the F scores of the feature importance analysis (FIA; Fig. 7 (b)) and the MERs (Fig. 7 (c)). These statistics are sent for display through Joins 10 and 11 in Fig. 5. The developer uses the F scores (Fig. 9) to remove the insignificant IDFs. For example, Fig. 9 shows that the GDP and the add-on room features are not important features. The figure also indicates that online (real-time) rainfall, humidity, PM2.5 and temperature data obtained from the sensor devices of IoTtalk are useful. In our study, we do not ignore any feature in prediction. Instead, ranking of all features in the importance analysis is used to provide useful information for the housing agency to sell the houses. In Altalk, the AI algorithms from different AI tools can be selected for ensemble learning. Without Altalk, the developer needs to accommodate these algorithms by using...
tedious APIs/libraries.

Step 7. Validation: In our example, the k-fold cross-validation method is selected for validation (Fig. 8 (7)). This method randomly splits 399,995 transaction records into the training data (80%, 319,996 transaction records) and the testing data (20%, 79,999 transaction records).

Step 8. Automatic Hyper-parameter Tuning (AHT): For the selected machine learning algorithms, the effect of the hyper-parameters is obtained through the calculation of the validation errors that are used by the hyper-parameter optimizer to tune the models. In the automatic hyper-parameter tuning step of our example, Grid Search (Fig. 8 (8)) exhaustively searches through a specified subset of the hyper-parameter space. That is, based on the validation errors, Grid Search identifies better choices to update the hyper-parameter settings. This training process repeats until the stop condition of the hyper-parameter optimizer is met. Without AItalk, the developer needs to implement control settings for various hyper-parameters. For example, TensorLayer does not provide hyper-parameter tuning. Such functions for various AI tools have been implemented in AItalk with uniform simple input interface. When the developer selects the name e.g., Grid Search, AItalk automatically conducts hyper-parameter tuning based on Grid Search.

Step 9. Training Data Selection: When we click the “Training data” button (Fig. 8 (9)), the ML setup window pops up a webpage to select the real-time data or the historical data for training (Fig. 10 (1)). We may further specify the start and the end dates (Fig. 10 (2)), the days in a week (Fig. 10 (3)) and the hours in a day (Fig. 10 (4)).

Step 10. Model Training: By clicking the “Train Model” button (Fig. 8 (10)), HouseValuation follows the setup of Steps 2-9 to collect data and train the model. This process uses k-fold cross-validation to tune the hyper-parameters for HouseValuation. The accuracy, precision, and recall for the 5 machine learning algorithms are obtained based on the current hyper-parameters. Then we use Grid Search to adjust the models.

We use the median error rate to evaluate the prediction (valuation) performance of different methods and compare them with Zestimate (which is Zillow’s estimated market value for real estate). As we mentioned, two types of models are used in our study: the all-features models (with MER1 error) and the housing-profile model (with MER2 error). Table 2 shows that the improvement of the all-features models over the corresponding housing-profile models ranges from 3.306% to 38.737%. The MER of the average ensemble model using all features is 6.461%. Note that our study considers both MERs for total house price by using the total-house-price label and the price-per-square-foot label, respectively. The accuracy of MER based on price-per-square-foot is better than that for the total-house-price. In Table 2 we list the results of MERs based on price-per-square-foot. We have also considered the non-linear relation between some features and the price by using the log-linear models [8], [20], [21], and [24]. The average error with the log-linear models (6.834%) is worse than the error without the log-linear models (6.461%). This result is due to the fact that the state-of-the-art algorithms like XGBoost or LightGBM have already optimized the mapping between the input and output features by considering the non-linear relation of these features. Therefore, there is no need to accommodate extra log-linear models. Although our error rate is higher than Zillow’s Zestimate that has a median error rate of 4.3%, our experiments only used 399,999 transaction records, which is about 0.4% of those used in Zillow (about 100 million records).

IoT sensors have not been directly used for house valuation. In this study, we show that by implementing real estate application in AItalk with IoT sensors, house valuation performance can be improved. As indicated in the feature importance analysis (Fig. 9), some features obtained from the IoT sensors (such as earthquake information used to derive types 1 and 2 distances) are major features. Other weather sensor data such as those for temperature, PM 2.5 are feathers that cannot be ignored.
VII. AITALK COMMUNICATIONS OVERHEAD

With the distributed IoT structure of AItalk, we can flexibly add and remove input data sources in the house data bank, and configure ML_device. However, such flexibility is offered with extra communication overhead.

The components of an AItalk application are categorized into three groups: the input devices (Fig. 2 (a), (b), and (i)), the IoTtalk engine (Fig. 2 (h)), and the output devices (Fig. 2 (c), (d) and (i)). We consider three network scenarios to locate the components of three groups with different communication delays. In the first scenario, all components are placed in one building connected with one-hop WLAN links. In the second scenario, the IoTtalk engine is located in NCTU’s private cloud, and the input and output devices are located at different places on campus. In the third scenario, the input and output devices are distributed on NCTU campus and the IoTtalk engine is installed in a commercial cloud. Through the communication protocols of IoT, two types of AItalk communication delays are incurred in the i-th scenario (1 ≤ i ≤ 3). The first delay \( d_{1,i} \) is incurred from an input device to the IoTtalk engine (i.e., paths (a)→(h), (b)→(h) or (i)→(3)→(4)→(e)→(h) in Fig. 2) and the second delay \( d_{2,i} \) is incurred from the IoTtalk engine to an output device (i.e., paths (h)→(e), (h)→(d), (h)→(e)→(1)/(2)→(i)).

Experiments are conducted to measure \( d_i \). The IoT devices and the IoTtalk server are synchronized with the Network Time Protocol (NTP) by installing the timer modules. Measurement experiments repeated 1200 times under commercial background traffic [3] are conducted in each of three scenarios. Fig. 11 (a) illustrates the histograms of \( d_i = d_{1,i} + d_{2,i} \), where \( E[d_1] = 13.2 \text{ ms} \), \( E[d_2] = 21.5 \text{ ms} \), and \( E[d_3] = 28.4 \text{ ms} \). The variances are \( V[d_1] = 0.028E[d_1]^2 \), \( V[d_2] = 0.012E[d_2]^2 \), and \( V[d_3] = 0.001E[d_3]^2 \). These variances are small, which are good. The expected delay times at a virtual machine in the cloud are slightly larger than that of a local server.

The computing times for one house price valuation excluding \( d_i \) are measured through 1000 experiments, which are \( t_a \) for the all-features model and \( t_h \) for the housing-profile model, respectively.

Fig. 11 (b) illustrates the histograms of \( t_a \) and \( t_h \), where \( E[t_a] = 1187.9 \text{ ms} \) , and \( E[t_h] = 771.9 \text{ ms} \). The variances are \( V[t_a] = 5.915E[d_1]^2 \) and \( V[t_h] = 0.008E[d_3]^2 \). Note that \( V[t_a] \) is larger than \( V[t_h] \) because an all-features model will invoke Google API for distance computation, which contributes to both \( E[t_h] \) and \( V[t_a] \). To produce one house price valuation, the AItalk’s communication overhead over the machine learning calculation times are \( E[d_1]/E[t_a] = 0.011 \), \( E[d_2]/E[t_a] = 0.018 \) and \( E[d_3]/E[t_a] = 0.024 \) for all-features modeling, and \( E[d_1]/E[t_h] = 0.017 \), \( E[d_2]/E[t_h] = 0.028 \), and \( E[d_3]/E[t_h] = 0.037 \) for housing-profile modeling. In all cases, the overheads are less than 3.7%, which are insignificant.

VIII. CONCLUSION

Through the concept of “cyber IoT device”, this paper describes how various AI tools such as scikit-learn, TensorFlow and Flux are integrated in AItalk, and shows that AItalk can also be used to develop any AI applications that do not involve real IoT devices. In our approach, the developer can transparently use different AI tools to develop their applications without worrying about the details of their APIs/libraries.

We used house valuation and pricing as an example to show how AItalk can flexibly include the factors that have significant impact on the house price. We showed that by adding extra features other than housing profile features (including the online IoT sensor data), the accuracy for the valuation can be improved by 38%. We also investigated the communication overhead of the distributed AItalk structure, which is 3.7% for the computation of one house price valuation. Our study shows that AItalk is convenient for developing AI application with negligible extra execution overhead.

REFERENCES


Yun-Wei Lin received Ph.D. degrees in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2011, respectively. He has been an Assistant Professor in the College of Artificial Intelligence, National Chiao Tung University, Taiwan since 2019. His current research interests include mobile ad hoc network, wireless sensor network, vehicular ad hoc networks, and IoT/M2M communications.

Jyun-Yi Lin received the B.S. in information management and finance from National Chiao Tung University, HsinChu, Taiwan, in 2016, and the M.S. in information management from National Chiao Tung University, HsinChu, Taiwan, in 2018. He is currently working towards the Ph.D. in computer science at National Chiao Tung University. His current research interests include machine learning and IoT platforms.

Yu-Lin Shih received the B.S. degree in industrial and information management from National Cheng Kung University, Taiwan, R.O.C., in 2012 and the M.S. degree in statistics from National Chiao Tung University, Taiwan, R.O.C., in 2018. Since 2018, he has been an engineer in the Wistron Corporation. His research interests include big data, machine learning, deep learning, computer vision, medical Image analysis, time series, statistics and algorithmic trading.