ML-based Telegram bot for real estate price prediction

Hamza Salem, Manuel Mazzara
Software and Service Engineering lab, Innopolis University, Republic of Tatarstan, Russia
E-mail: h.salem@innopolis.university, m.mazzara@innopolis.ru

Abstract. Telegram chatbots are one of the main sections in Telegram applications that get more attention and provide huge business opportunities for companies nowadays. Also, Machine learning algorithms have been used in numerous fields and industries and one of the main growing markets is real estate. It is already helping real estate agents to respond more quickly to clients’ questions and develop landing pages that fit customers’ needs. However, one of the main time-consuming stages for a real estate agent is the process of answering questions about price per location and knowing which locations clients are interested in. In order to solve these problems, we have built a telegram chatbot to answer customers of any real estate agent with the price of a real estate property based on their current Geolocation. This bot can predict the price of a real estate property using room numbers, Geolocation, and surface area in square meters. These are the inputs for machine learning algorithms to give an approximate price. The bot can predict based on the source of the data-set, for example, we have collected our data from classified ads website for real estate in Amman, Jordan. We have used python and web scraping library for data extraction, cleaning, and transformation. As a result, by using Geolocation we have increased the model accuracy to 1.3X and our bot can be replicated in any markets based on the dataset. We believe our work can be taken to different markets and make real estate agent’s job easy and more profitable by changing leads to potential customers thought answering their questions.

Keywords— Machine Learning, Real Estate, web scrapping, Telegram Bot, Geolocation

1. Introduction

Bots are just a Telegram account operated by software and most of them have AI features based on the developer. Bots can do a lot of things such as messaging, teaching, answering questions, and integrating with other services [1][2]. AI systems for real estate pricing are recent and becoming more integrated with other platforms such as website bots and other platforms like Messenger, WhatsApp, and telegram [3]. The traditional real estate property pricing is based on the cost and the age of the building, however, most of the current pricing model is trying to remove these parameters and focus only on the main features for a house and Geolocation [4]. For example, two houses in Amman with the same number of rooms, baths, and surface area in square meters are different based on Geolocation (longitude and latitude) and the difference is not just a couple of dollars sometimes it can be double of the other price.

We have used Scikit-learn supervised learning algorithms such as KNeighborsClassifier, KNeighborsRegressor, LinearRegression, and DecisionTreeClassifier to predict prices for a real
estate property [5]. The whole application can be divided into four parts: First, Web scraping script to scrap the dataset. Second, Data cleaning and converting text to numbers. Third, Build a model and test several algorithms and compare results. Finally, integrate the model in the Telegram Bot Platform and try it with real user input. We believe that our research value is the primary combination of Telegram bot platform with machine learning algorithms in real estate markets and provides a new way to interact with potential customers with an automated process and convert leads to potential customers based on the input. The significance of this research lies in 1) providing a platform for price prediction based on Geolocation; 2) Help real estate agents to convert potential customers to customers based on their interaction with our Bot; 3) realizing the preliminary combination of Telegram Bot platform and machine learning. This research can be an initial step into creating a general platform for all real estate agents to create their own Bot using easy steps without any programming knowledge or machine learning experience, more on that later in the future work section. This paper is organized as follows, The first section gives a brief overview of the significant results of the paper. The second section analyses the literature review and other related work done before our solution. The third section presents our methodology for building the solution and show the main components of it. The Forth section shows the implantation part for each part of the system. The fifth section demonstrates the results and shows our model accuracy and evaluations. The sixth, seventh sections show limitations and conclusions of the paper and discuss the future work for such a solution.

2. Related work & literature Review
For real estate agents, time is money: so when they close more deals with less time they gain more money. In addition, in real estate software converting users to paying customers is not a simple process; they need more time to invest for these users to answer questions and provide them with answers and suggestions [6].

In the literature there few examples of integrating machine learning in the real estate market. For example, in [7] the authors identify opportunities in real estate market in real-time so investors interested in the housing market will know the prices of houses when it will go under specific prices. Moreover, another solution [8] has provided a comprehensive model for estimating the price of new housing in any city at the design phase or beginning of the construction based on construction companies to gauge the sale market. In [9] the authors proposed a housing price prediction model based on machine learning algorithms such as Naïve Bayesian, and AdaBoost and compared their classification accuracy performance. However, they [9] draw our attention to continue their own work and create our own model and integrate it into a Telegram bot to generate a final product to help customers and real estate agents.

Our approach proposes chatbot to interact with customers; bots cost less than applications and other solutions, and chatbot gives customers better experiences [10]. Also, we have focused to solve the issue of replaying on customers with prices per Geolocation and make real estate agents reply faster by giving prices directly and create a list of customers who are interested in locations based on their questions. Our model can be generalized based on the source of the dataset, offers a fair market price based on Geolocation, and provides an initial step to building a platform for real estate agents from any country to put their own data and create their chatbot without knowledge of machine learning and bots development.

3. Method & Data
The Bot has been built using Python as the main language and python-telegram-bot library to integrate with Telegram API [11]. In the data collection, a custom scraper has been built using Python request library and Beautifulsoup library to extract information to a CSV file. The dataset comes from a website called opensooq and select Jordan as a country and search about apartments in Amman only [16]. In the data transformation and processing, we have used
Python pandas library to convert the CSV file into a data frame format, and convert Address to Geolocation parameters (longitude, latitude).

In the price prediction, we have used different Machine Learning algorithms [5], such as KNeighbors, LinearRegression and DecisionTree. In the Bot implementation, we have built the bot to interact with the price prediction model and provide an approximated price based on the input of the Bot user.

4. Design & implementation
This section begins by examining the overflow for the process of building our bot. We started with Bot Customers Interaction to the stages of collecting the dataset from scraping, cleaning, and transformation. Finally, we present the chosen machine learning model and our implementation and integration with the bot.

4.1. Bot Customers Interaction
Python Telegram Bot library provides an interface for the Telegram Bot API and it is compatible with Python versions 3.6+ [11]. By using this library we had implemented the steps for the interaction with potential customers. The overflow for the steps are presented in Figure 1.

![Bot Customer interaction Steps](image)

**Figure 1.** Bot Customer interaction Steps

When the customers enter our chatbot it will ask the first question about the apartment type categorized by size and the question will be “what are you looking for?”, it can be seen in Figure 2 and it will show two options:

(i) Regular apartment (include large size)
(ii) One room apartment (Small Size)

After the customer will choose the type, the bot will ask four different questions (Figure 3-5):

(i) Send me your location or the location that you want to see the apartment price there?
(ii) How many bathrooms do you want?
(iii) How many sleeping rooms do you want?
(iv) How much area in square meters you want?
Based on these questions the bot will send the input to the machine learning model to measure the price and send you back the output result as an approximate price for an apartment with these features like you see in figure 6.

4.2. Data Scraping and Cleaning
This section is divided to two main subsections. The first subsection is showing the process of data scraping and why we have used these technique. The second subsection is demonstrate the process of of data cleaning after scraping.

4.2.1. Data Scraping
Our dataset is extracted using a custom scraper built by BeautifulSoup [12]. The scraper works as an automation request for apartments listed on a website. For example, imagine you
will hire 10 employees to extract the data seen in Figure 7 for hundreds of pages. What is the time and cost for that process? That’s why we are using python-requests with Beautiful Soup library; to make it easier and faster. The script views the listing by request each URL based on the page number and save the required data (Price, Address as text, sleeping room number, bathroom number, area in square meters) and save to CSV file.

4.2.2. Data Cleaning

Because the scraped data has some (labels or unwanted texts) and this can cause problems in the model implementation part so cleaning data is our next step. As seen in Figure 7 when data has been extracted the price had comma ‘,’ and the currency too. both should be removed to be numbers only. As you can see in the code below we used replace() function in python to replace the comma with nothing.

```python
price.text.replace(',', '')
```

The second step for data cleaning is to remove labels and keep only numbers. For example, in the sleeping room, bathroom, and area in square meters attribute we have unwanted labels. We have created an if-condition to replace labels with the right number such as (3 bathrooms =3). Finally, most of the columns in CSV are numbers ready for the next step.

4.3. Data Transformation

In this part of the research, we had cleaned our data but still, we have two main problems. Some unwanted attributes such as (address as text) should be converted to numbers. Also, there are some outlier values in the price column; and the reason is sellers try to attract people with fake small numbers will be paid monthly and they write these values in the final price value. See in Figure 8 price is 1 JOD and it is not accurate of a price for an apartment. To solve the first problem, We have used Geopy library as a Python client for several popular geocoding web

![Figure 7. website apartments listing](image)
services. It makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks [13]. In the code below you can see the location variable is created to get longitude and latitude for each address.

```python
location = geolocator.geocode("Address text")
data['latitude'] = location.latitude
data['longitude'] = location.longitude
```

For the second problem, we just add if-condition to filter data less than 5000 JOD and we have chosen this number after we analyzed the data manually and most of the outlier has been deleted after applying the condition and data now makes sense to start with the model implementation.

### 4.4. Model implementation

In model implementation, we have used Sklearn different algorithms measure the price such as KNeighborsClassifier, KNeighborsRegressor, LinearRegression, and DecisionTreeClassifier as shown in the code below [5].

```python
m_tree = tree.DecisionTreeClassifier()
m_KN = KNeighborsClassifier(n_neighbors=3)
m_KNR = KNeighborsRegressor(n_neighbors=3)
m_reg=LinearRegression()
```

Also, we split our dataset set to test data and training data to measure the accuracy for each algorithm. By using "train test split" method our dataset will be split into 75% training data, 25% test Data.

```python
x=df.loc[:, ['m', 'latitude','longitude','bathroom','sleepingroom']].to_numpy()
y =df.loc[:, ['price']].to_numpy()
trainX, testX, trainy, testy = train_test_split( x, y, test_size=0.25, random_state=40)
x: represent the inputs
y: represent the output
trainX, trainy : for training function.
testX,testy : for testing function.
```

In Sklearn as seen the code below, fit() take two inputs X,Y, and create the model with training dataset.

```python
m_KN = KN.fit(trainX,trainy )
m_KNR = KNR.fit(trainX,trainy )
```
**Figure 9.** train test split

```python
m_reg = LinearRegression().fit(trainX, trainy)
m_tree = tree.fit(trainX, trainy)

The next process is to execute prediction based on values, for example, the code below we defined a test array with inputs [area square meter, latitude, longitude, bathroom number, sleeping room number].

test_array = [360, 31.9880592, 35.8113021, 5, 5]
CLF_tree_prediction = m_tree.predict([[test_array]])
KN_prediction = m_KN.predict([[test_array]])
KNR_prediction = m_KNR.predict([[test_array]])
reg_prediction = m_reg.predict([[test_array]])

And the model will give different results based on each algorithm. After the training stage, the code below shows a Pickle file for the chosen model. Pickle files are used to save and load your machine learning model in Python for production environments [14].

joblib.dump(clf_tree, 'clf_tree.pkl')
#Load the model from the file
clf_tree_pickle = joblib.load('clf_tree.pkl')
```

5. Result
5.1. Model evaluation
In order to evaluate each model to decide which model we should use for this bot, we have used score(testX, testY) that returns the mean accuracy on the given test data and labels. Below in Table 1 shows the score for each model based on their Sklearn algorithms for our dataset with and without Geolocation (latitude, longitude) [15].

The best score came from KNeighborsRegressor. Adding Geolocation to the inputs improved the score 1.3X and that’s why we used Telegram Bot to make it easy to get Geolocation coordinates.
### Table 1. sklearn algorithms scores with Geolocation and without

| Sklearn Algorithms       | Score Without Geolocation | Score With Geolocation |
|--------------------------|----------------------------|------------------------|
| KNeighborsClassifier     | 0.0495611770779556        | 0.10944759938048529    |
| KNeighborsRegressor      | 0.38236470340881673      | 0.506903497326561      |
| LinearRegression         | 0.3456337979698737       | 0.3428184446205108     |
| DecisionTreeClassifier   | 0.08879710893133712      | 0.14713474445018068    |

6. Limitations

The shortcomings of the project are:

(i) Model accuracy depends on the popularity of addresses on the selected area. For example, the bot will work only in Amman (the capital of Jordan) and this area has been converted into Coordinates easily using Geopy. Other areas will have problems preparing a dataset on the step of data transformation.

(ii) Telegram Bot can take inaccurate current location based on the internet connection and that’s why sometimes the bot will retrieve different results for the same neighborhood.

(iii) Some items in the website listing can provide more parameters to increase accuracy. However, these parameters will not be used by our model until it will exist in all websites listing items. For example, some items in our dataset had extra information such as (floor number, extra yard) and these extra parameters will influence the price prediction.

7. Conclusion

In conclusion, our work is an attempt to combine real estate applications with machine learning and provide user-friendly bot to interact with potential customers for real estate agents and convert these users to buying customers. The model use data comes from a website for classified ads scraped by custom scraper build using Python and BeautifulSoup. These data are cleaned and transformed to be a dataset for our machine learning model to predict the price for real estate apartments in Amman. Our bot gives an approximate price based on parameters such as (Address, sleeping room number, bathroom number, area in square meters). We are aware that our research may have limitations too. The first one is Model accuracy based on the popularity of addresses in any selected area, it can be solved in the future by providing more data about areas around the world. The second problem is an inaccurate location in telegram and this one can be avoided by a good internet connection. The third problem is increasing accuracy still questionable based on the dataset, with good details we will have more accurate results. In the future, we are planning to create a full system to make real estate agents choose their own dataset and create their own bots without any knowledge in development only UI elements and drag-drop components.

References

[1] Klopfenstein, L. C., Delpriori, S., Malatini, S., Bogliolo, A. (2017, June). The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In Proceedings of the 2017 conference on designing interactive systems (pp. 555-565).

[2] Setiaji, H., Paputungan, I. V. (2018, March). Design of telegram bots for campus information sharing. In IOP Conference Series: Materials Science and Engineering (Vol. 325, No. 1, p. 012005). Institute of Physics Publishing.

[3] Park, B., Bae, J. K. (2015). Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data. Expert Systems with Applications, 42(6), 2928-2934.
[4] Northcraft, G. B., Neale, M. A. (1987). Experts, amateurs, and real estate: An anchoring-and-adjustment perspective on property pricing decisions. Organizational behavior and human decision processes, 39(1), 84-97.
[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.
[6] Poon, J., Hoxley, M., Fuchs, W. (2011). Real estate education: an investigation of multiple stakeholders. Property Management.
[7] Baldomínicos, A., Blanco, I., Moreno, A. J., Iturrarte, R., Bernárdez, Ó., Afonso, C. (2018). Identifying real estate opportunities using machine learning. Applied sciences, 8(11), 2321.
[8] Rafiei, M. H., Adeli, H. (2016). A novel machine learning model for estimation of sale prices of real estate units. Journal of Construction Engineering and Management, 142(2), 04015066.
[9] Park, B., Bae, J. K. (2015). Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data. Expert Systems with Applications, 42(6), 2928-2934.
[10] Klopfenstein, L. C., Delpriori, S., Malatini, S., Bogliolo, A. (2017, June). The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In Proceedings of the 2017 conference on designing interactive systems (pp. 555-565).
[11] Python-Telegram-Bot. (n.d.). Python-telegram-bot/python-telegram-bot. Retrieved October 15, 2020, from https://github.com/python-telegram-bot/python-telegram-bot.
[12] Richardson, L. (2007). Beautiful soup documentation. Dosetano: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018].
[13] Welcome to GeoPy’s documentation. (n.d.). Retrieved October 16, 2020, from https://geopy.readthedocs.io/en/stable/.
[14] Brownlee, J. (2020, August 27). Save and Load Machine Learning Models in Python with scikit-learn. Retrieved October 16, 2020, from https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/.
[15] Bisong, E. (2019). More supervised machine learning techniques with scikit-learn. In Building Machine Learning and Deep Learning Models on Google Cloud Platform (pp. 287-308). Apress, Berkeley, CA.
[16] Opensouq. (n.d.). Retrieved October 18, 2020, from https://jo.opensooq.com/ar