TOPIC: HOSTEL PRICE FORECAST IN VIETNAM

FINAL PROJECT REPORT MACHINE LEARNING IN BUSINESS ANALYTICS

Lecturer: Truong Hoai Phan, M.S.
Group: Ian

06 THÁNG NĂM 2023

UNIVERSITY OF ECONOMICS AND LAW
FACULTY OF INFORMATION SYSTEMS

Members of Group Lan

NO.	NAME	STUDENT ID	ROLE	PHONE	EMAIL
1	Nguyen Hoang Minh Nhat	K204161996	Leader	0346982464	nhatnhm20416c @st.uel.edu.vn
2	Bui Van Toan	K204162002	Member	0354570471	toanbv20416c @st.uel.edu.vn
3	Nguyen Quang Tuan	K204162005	Member	0794160602	tuannq20416c @st.uel.edu.vn
4	Do Quang Huy	K204160662	Member	0936217867	huydq20416c @st.uel.edu.vn

Content

Image Catalog

1. Project Overview

1.1 Reasons for implementation

House prices are one of the important issues in the real estate market. Predicting home prices can help us understand market trends and make smart business decisions. In addition, this issue also has high applicability in practice, especially in the field of real estate and finance.

1.2 Objectives

The main goal of this topic is to be able to be applied in practice to predict house prices for subjects such as investors, real estate companies, homeowners and home buyers. Home price prediction results can help the subject make the right decision about buying, selling, or investing in real estate.

1.3 Objects and scope of the project

The subjects of application of this topic can be experts and analysts in the field of real estate, as well as students and trainees in universities and colleges training in this field.

2. Prepare the data

2.1 Data collection and cleaning

Crawling Sites

After identifying the problem, the goal and the scope of the problem to be implemented, we decided to choose mogi.vn website to collect data. Currently, there are many websites that post information about motel room rentals, but many motel owners post their motel rooms on many different websites, so if we collect many websites at the same time, there will be data duplication. Therefore, we decided to choose the most popular website that is posted by many inn room owners, which is the mogi.vn website.

Data collection tools

We use the Octoparse application to collect data on the website to the machine, Octoparse is suitable software for people who don't know how to code, it automatically identifies patterns on the website, from which we only need drag-and-drop operations, basic data input to get the data about, however, users need to understand Octoparse's process to get the right data. We use its free version, which allows the collection of up to 10,000 data lines, the maximum speed for data collection is 5 lines per second. After collecting the data using Octoparse, we get the data table as follows.

	title	Title_URL	address	area	bedroom	wc	Price	Text	date_subr	urgent_re	Deatail_a	id
2	Cho thuê	https://m	Quận 3, TPHCM	18 m2	0 PN	0 WC	2 triệu 70	Còn 3 phòng cho thuê như	17/03/202	Cho thuê	Cách Mạn	21823331
3	P 2,3 tr giả	https://m	Quận 7, TPHCM	12 m2	0 PN	0 WC	2 triệu	- phòng siêu đẹp trong căn hộ	17/03/202	Cho thuê	Nguyễn Lu	22015945
4	Phòng Trọ	https://m	Quận Tân Bình, TPHCM	25 m2	0 PN	0 WC	2 triệu 70	Phòng Trọ Gía sinh viên chỉ 2t	17/03/202		Cống Lở, F	22041403
5	1 PN Ban	https://m	Quận Gò Vấp, TPHCM	35 m2	0 PN	0 WC	4 triệu 60	Địa chỉ: 120 đường 59, Phư	17/03/202		120/69 Đư	22050586
6	PHÒNG SIÍ	https://m	Quận Gò Vấp, TPHCM	50 m2	0 PN	0 WC	5 triệu	Phòng Siêu Rộng 50m2 - Full N	17/03/202		120/69 Đư	22050575
7	Phòng siê	https://m	Quận Gò Vấp, TPHCM	20 m2	0 PN	0 WC	2 triệu 80	Phòng cho thuê đường Phạm '	17/03/202		Phạm Văn	21991190
8	Duplex mo	https://m	Quận 7, TPHCM	45 m2	0 PN	0 WC	6 triệu	Căn hộ dịch vụ full nội thất - E	17/03/202		Đường số	22041964
9	CHDV cho	https://m	Quận 7, TPHCM	40 m2	0 PN	0 WC	5 triệu	CHDV full nội thất phòng rộng	17/03/202		Nguyễn Hi	22056290
10	Chỉ còn 2	https://m	Quận 2 (TP. Thủ Đức), T	35 m2	0 PN	0 WC	3 triệu 50	Vị trí: Nguyễn Thị Định, quận 2	17/03/202		Nguyễn Th	22044497
11	New 100%	https://m	Quận Tân Phú, TPHCM	25 m2	0 PN	0 WC	4 triệu	- Dự án mới tinh 100% ngay Tr	17/03/202		Lũy Bán Bi	22061379
12	Phòng Trọ	https://m	Quận Gò Vấp, TPHCM	25 m2	0 PN	0 WC	3 triệu 80	Phòng Trọ FULL nội Thất có gá	17/03/202		Nguyễn Vă	22004575
13	Phòng cc H	https://m	Quận 7, TPHCM	25 m2	0 PN	0 WC	3 triệu 20	Phòng master 5tr giảm còn 4tr	17/03/202		Trần Xuân	21892483
14	Cho thuê	https://m	Quận Bình Thạnh, TPH	18 m2	0 PN	0 WC	4 triệu 50	Chung cư chính chủ cho thuê	17/03/202		Ung Văn K	22017936
15	Cho thuê	https://m	Quận 7, TPHCM	40 m2	0 PN	0 WC	5 triệu	Cho thuê CHDV giờ giấc tự do	17/03/202		Nguyễn Hi	22057858
16	KHAI T	https://m	Quận Tân Bình, TPHCM	38 m2	0 PN	0 WC	4 triệu 50	- KHAI TRUONG Căn hộ full nộ	17/03/202		Bạch Đằng	22054389
17	Căn Hộ St	https://m	Quận Tân Bình, TPHCM	22 m2	0 PN	0 WC	5 triệu 90	TOÀ NHÀ 28 CĂN HỘ DỊCH	17/03/202		31 Tân Hả	21545429
18	Cho thuê	https://m	Quận Tân Bình, TPHCM	16 m2	0 PN	0 WC	3 triệu	Chính chủ: cho thuê phòng trọ	17/03/202		Nguyễn Th	22054437
19	Cho Thuê	https://m	Quận 9 (TP. Thủ Đức), T	40 m2	0 PN	0 WC	3 triệu 50	KHAI TRƯƠNG CĂN HỘ CÓ GÁ	17/03/202		Lã Xuân O	22048273
20	phòng trọ	https://m	Quận 7, TPHCM	16 m2	0 PN	0 WC	2 triệu 20	cho thuê phòng- phòng 1tr7 (	17/03/202		Chuyên D	21827597
21	Duplex ful	https://m	Quận 7, TPHCM	25 m2	0 PN	0 WC	3 triệu 50	Căn hộ dịch vụ full nội thất- C	17/03/202		Huỳnh Tấ	22053381
22	PHÒNG ST	https://m	Quận Tân Bình, TPHCM	22 m2	0 PN	0 WC	6 triệu 20	Phòng Studio Full Nội Thấ	17/03/202		31 Tân Hả	21548645

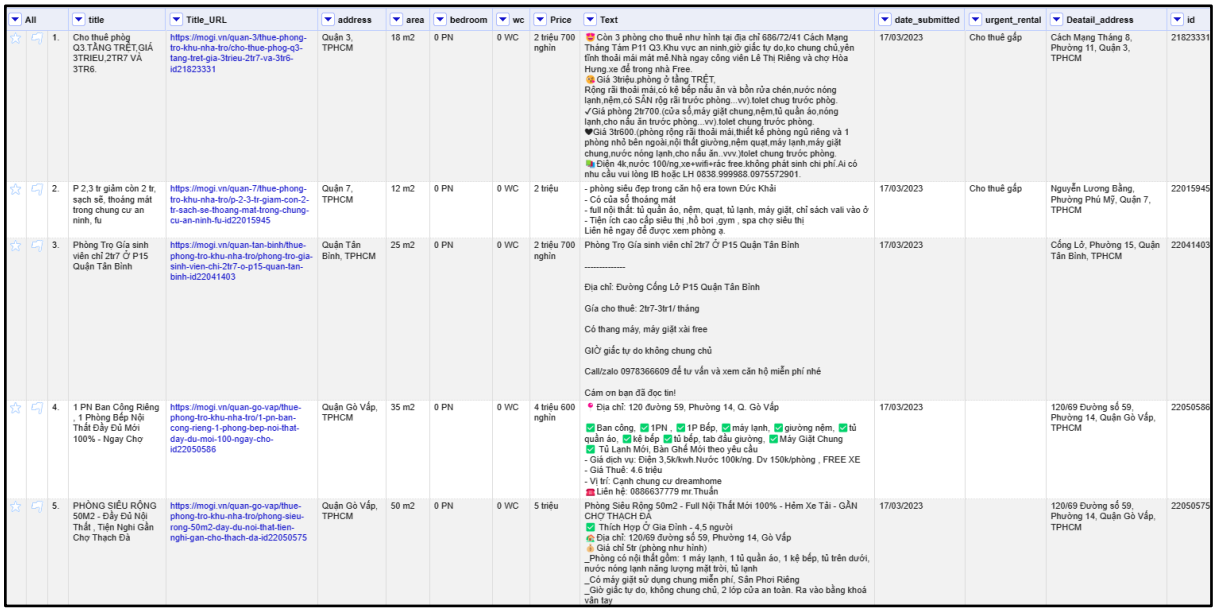
Photo 2. 1: Data after collection using Octoparse

Data Processing

Once we have the dataset, we will proceed to process and normalize the dataset, which we do using the Openrefine tool. Openrefine is a completely free application that

helps clean and transform data. We proceed with the data processing in Openrefine according to the following steps:

- **Delete unnecessary columns**: the data we collect with Octoparse has a lot of columns, including those that are not related to the problem such as: bedroom, wc, title_url. So the first step is to delete these columns, in Openrefine to delete a column simply point to that column and delete it.



All		title	Title_URL	address	area	bedroom	wc	Price	Text	date_submitted	urgent_rental	Deatail_address	id
1.		Cho thuê phòng Q3.TẦNG TRỆT,GIÁ 3TRIEU,2TR7 VÀ 3TR6.	https://mogi.vn/quan-3/thue-phong-tro-khu-nha-tro/cho-thue-phog-q3-tang-tret-gia-3trieu-2tr7-va-3tr6-id21823331	Quận 3, TPHCM	18 m2	0 PN	0 WC	2 triệu 700 nghìn	❤️Còn 3 phòng cho thuê như hình tại địa chỉ 686/72/41 Cách Mạng Tháng Tám P11 Q3.Khu vực an ninh,giờ giấc tự do,ko chung chủ,yên tĩnh thoải mái mát mẻ.Nhà ngay công viên Lê Thị Riêng và chợ Hòa Hưng,xe để trong nhà Free. 🍊Giá 3triệu,phòng ở tầng TRỆT, Rộng rãi thoải mái,có kệ bếp nấu ăn và bồn rửa chén,nước nóng lạnh,nệm,có SÂN rộng rãi trước phòng...vv).toiet chug trước phòg. ✓Giá phòng 2tr700 (cửa sổ,máy giặt chung,nệm,tủ quần áo,nóng lạnh,cho nấu ăn trước phòng...vv).tolet chung trước phòng. ❤Giá 3tr600.(phòng rộng rãi thoải mái,thiết kế phòng ngủ riêng và 1 phòng nhỏ bên ngoài,nội thất giường,nệm quạt,máy lạnh,máy giặt chung,nước nóng lạnh,cho nấu ăn..vvv,)toilet chung trước phòng. 🍉Điện 4k,nước 100/ng,xe+wifi+rác free.không phát sinh chi phí.Ai có nhu cầu vui lòng IB hoặc LH 0838.999988.0975572901.	17/03/2023	Cho thuê gấp	Cách Mạng Tháng 8, Phường 11, Quận 3, TPHCM	21823331
2.		P 2,3 tr giảm còn 2 tr, sạch sẽ, thoáng mát trong chung cư an ninh, fu	https://mogi.vn/quan-7/thue-phong-tro-khu-nha-tro/p-2-3-tr-giam-con-2-tr-sach-se-thoang-mat-trong-chung-cu-an-ninh-fu-id22015945	Quận 7, TPHCM	12 m2	0 PN	0 WC	2 triệu	- phòng siêu đẹp trong căn hộ era town Đức Khải - Có cửa sổ thoáng mát - full nội thất: tủ quần áo, nệm, quạt, tủ lạnh, máy giặt, chỉ sách vali vào ở - Tiện ích cao cấp siêu thị ,hồ bơi ,gym , spa chợ siêu thị Liên hệ ngay để được xem phòng ạ.	17/03/2023	Cho thuê gấp	Nguyễn Lương Bằng, Phường Phú Mỹ, Quận 7, TPHCM	22015945
3.		Phòng Trọ Gía sinh viên chỉ 2tr7 Ở P15 Quận Tân Bình	https://mogi.vn/quan-tan-binh/thue-phong-tro-khu-nha-tro/phong-tro-gia-sinh-vien-chi-2tr7-o-p15-quan-tan-binh-id22041403	Quận Tân Bình, TPHCM	25 m2	0 PN	0 WC	2 triệu 700 nghìn	Phòng Trọ Gía sinh viên chỉ 2tr7 Ở P15 Quận Tân Bình ---------- Địa chỉ: Đường Cống Lò P15 Quận Tân Bình Gía cho thuê: 2tr7-3tr1/ tháng Có thang máy, máy giặt xài free GIỜ giấc tự do không chung chủ Call/zalo 0978366609 để tư vấn và xem căn hộ miễn phí nhé Cảm on bạn đã đọc tin!	17/03/2023		Cống Lở, Phường 15, Quận Tân Bình, TPHCM	22041403
4.		1 PN Ban Công Riêng , 1 Phòng Bếp Nội Thất Đầy Đủ Mới 100% - Ngay Chợ	https://mogi.vn/quan-go-vap/thue-phong-tro-khu-nha-tro/1-pn-ban-cong-rieng-1-phong-bep-noi-that-day-du-moi-100-ngay-cho-id22050586	Quận Gò Vấp, TPHCM	35 m2	0 PN	0 WC	4 triệu 600 nghìn	📍Địa chỉ: 120 đường 59, Phường 14, Q. Gò Vấp ✅Ban công, ✅1PN , ✅1P Bếp, ✅máy lạnh, ✅giường nệm, ✅tủ quần áo, ✅kệ bếp ✅tủ bếp, tab đầu giường, ✅Máy Giặt Chung ✅Tủ Lạnh Mới, Bàn Ghế Mới theo yêu cầu - Giá dịch vụ: Điện 3,5k/kwh.Nước 100k/ng. Dv 150k/phòng , FREE XE - Giá Thuê: 4.6 triệu - Vị trí: Cạnh chung cư dreamhome ☎️Liên hệ: 0886637779 mr.Thuần	17/03/2023		120/69 Đường số 59, Phường 14, Quận Gò Vấp, TPHCM	22050586
5.		PHÒNG SIÊU RỘNG 50M2 - Đầy Đủ Nội Thất , Tiện Nghi Gần Chợ Thạch Đà	https://mogi.vn/quan-go-vap/thue-phong-tro-khu-nha-tro/phong-sieu-rong-50m2-day-du-noi-that-tien-nghi-gan-cho-thach-da-id22050575	Quận Gò Vấp, TPHCM	50 m2	0 PN	0 WC	5 triệu	Phòng Siêu Rộng 50m2 - Full Nội Thất Mới 100% - Hẻm Xe Tải - GẦN CHỢ THẠCH ĐÀ ✅ Thích Hợp Ở Gia Đình - 4,5 người 🏠Địa chỉ: 120/69 đường số 59, Phường 14, Gò Vấp 🔥Giá chỉ 5tr (phòng như hình) _Phòng có nội thất gồm: 1 máy lạnh, 1 tủ quần áo, 1 kệ bếp, tủ trên dưới, nước nóng lạnh năng lượng mặt trời, tủ lạnh _Có máy giặt sử dụng chung miễn phí, Sân Phơi Riêng _Giờ giấc tự do, không chung chủ, 2 lớp cửa an toàn. Ra vào bằng khoá vân tay	17/03/2023		120/69 Đường số 59, Phường 14, Quận Gò Vấp, TPHCM	22050575

Photo 2. 2: Data after being loaded into Openrefine

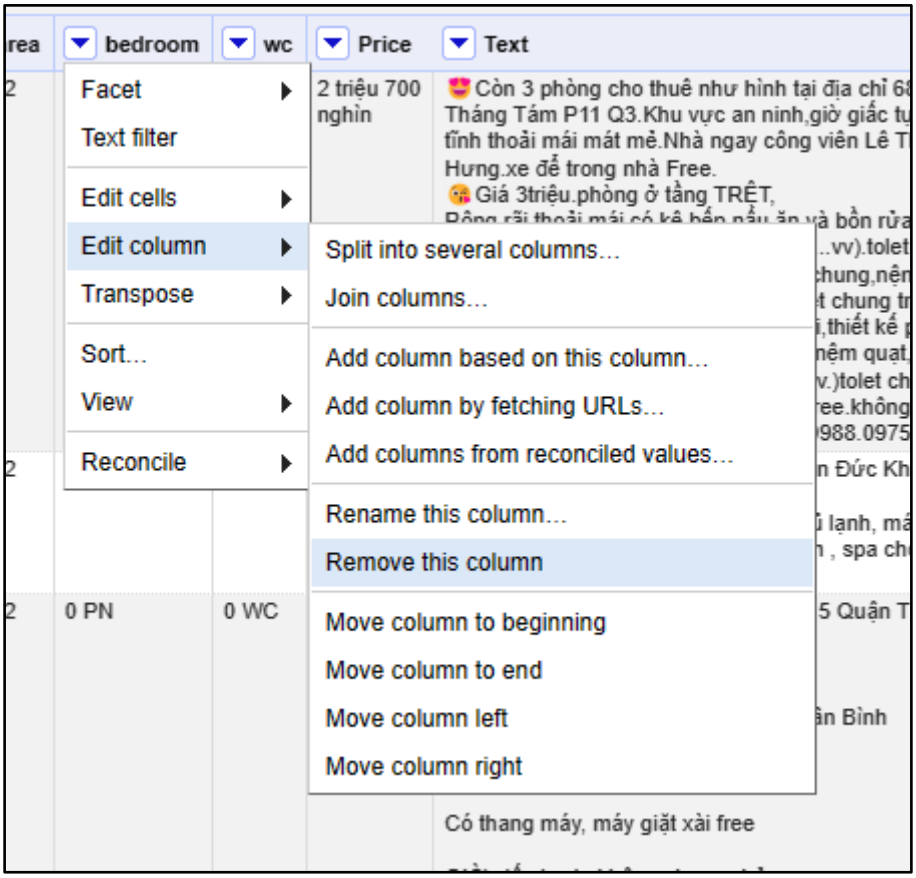


Photo 2. 3: Delete a column in the dataset

- **Data normalization:** in the dataset there may be heterogeneous values (e.g., District 2 ⇔ District 2), values we don't want (e.g., 2 million 300 thousand => 2.3 m) so we need to normalize all the values in the dataset.
    - Openrefine supports normalization operations in an extremely intuitive and convenient way, to identify values with the same meaning but present different forms in the same style using the "Text Facet" function.
    - Next is the change in the type of value we want. For example, for the Price column, we want to convert it back to a numeric type and convert it to a decimal form (e.g. 2 million 500 thousand => 2.5), in Openrefine we use the transform function. To transform data, Openrefine provides us with 3 programming languages: GREL, python, Clojure. We will perform the conversion using the Python language.

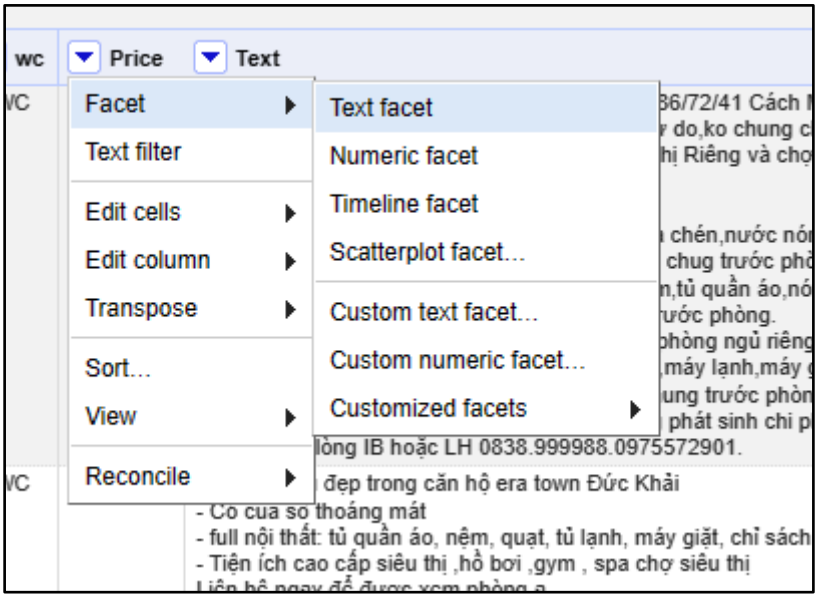


Photo 2. 4: Text facet function in Openrefine

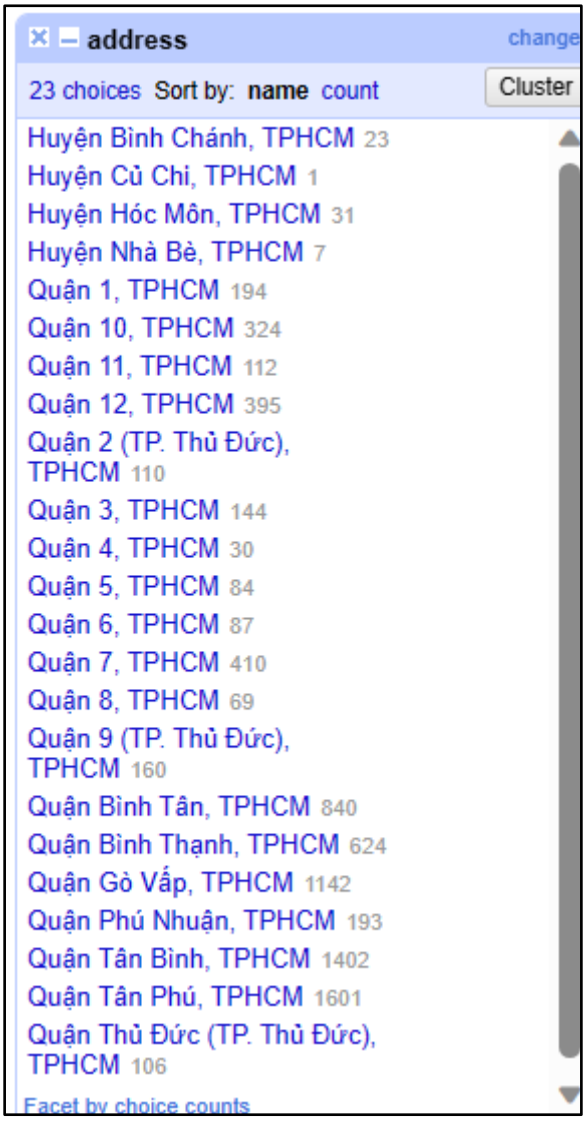


Photo 2. 5: The "address" column after using the text facet function

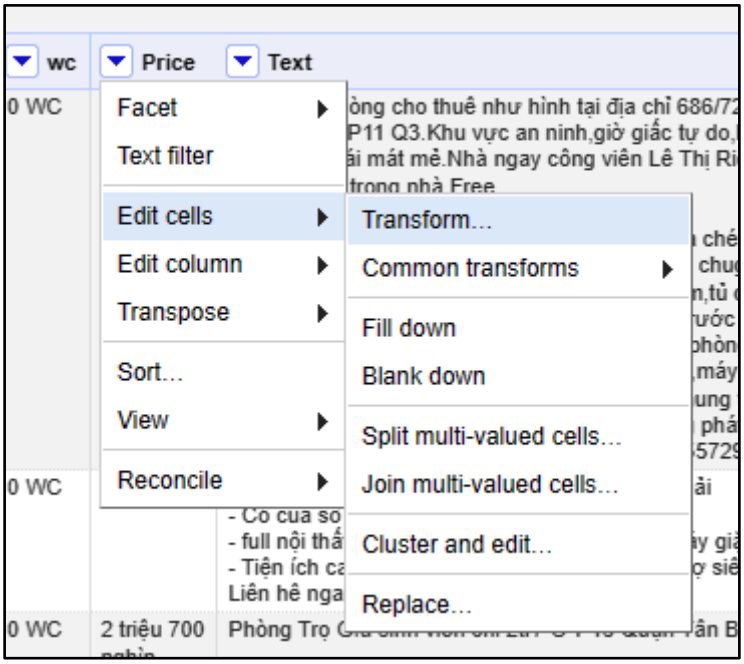


Photo 2. 6: Transform Function in Openrefine

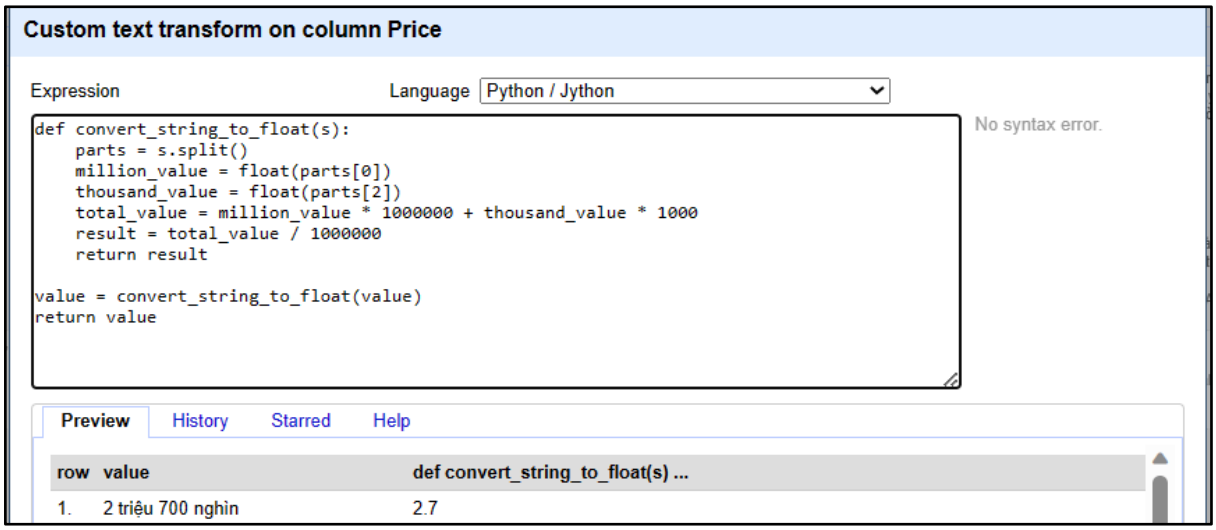


Photo 2. 7: Transforming Data Using Python in Openrefine

- Added new columns: for the purpose of forecasting hostel prices in Ho Chi Minh City, we need to add related factors, affecting house prices, so we will rely on the text column (meaning to describe more hostel-related factors such as: cameras, washing machines...) to add new columns. We use the Add column based on function in Openrefine to add new columns. Then we rely on the Text filter function to filter the values related to this new column (for example, we want the new column to be: air conditioning (meaning whether the room has air conditioning or not?), we will text filter from the air conditioner to filter the lines contained from the air conditioner. It can be seen that there are 3908 columns containing this word, so we can complete the air conditioner column including:

2 values 1 (containing the word air conditioner => 3908 lines just found), 0 the remaining values.

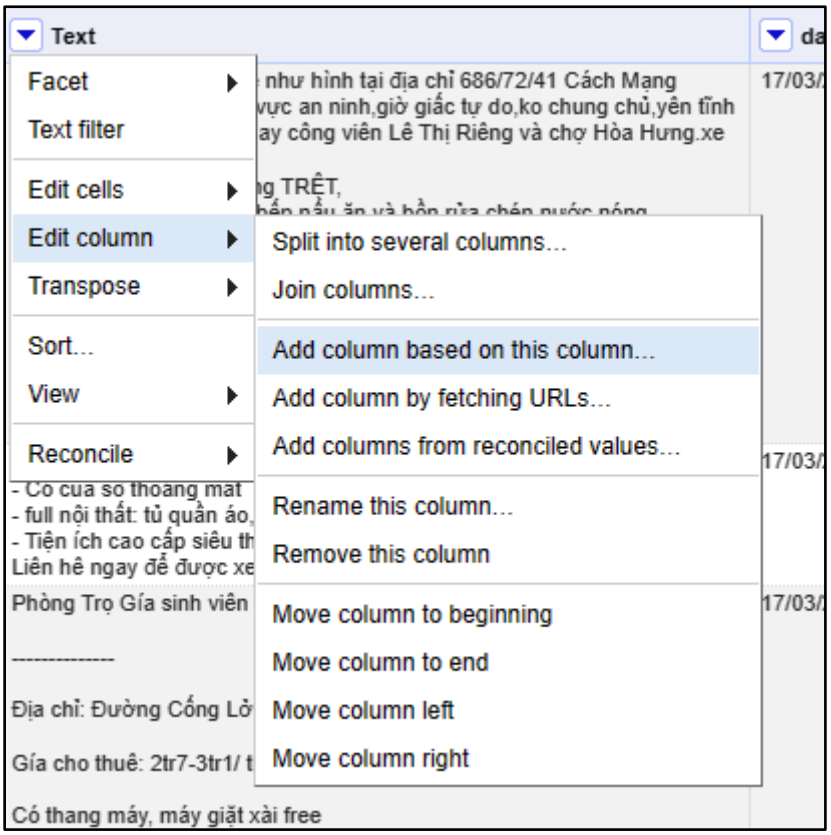


Photo 2. 8: Add column function based on this column in Openrefine

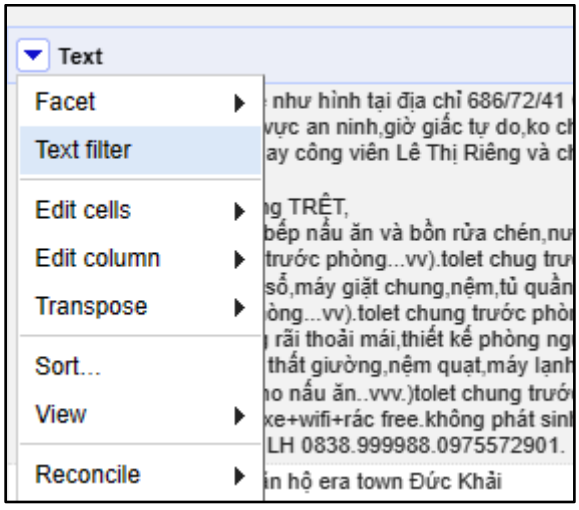


Photo 2. 9: Text filter function

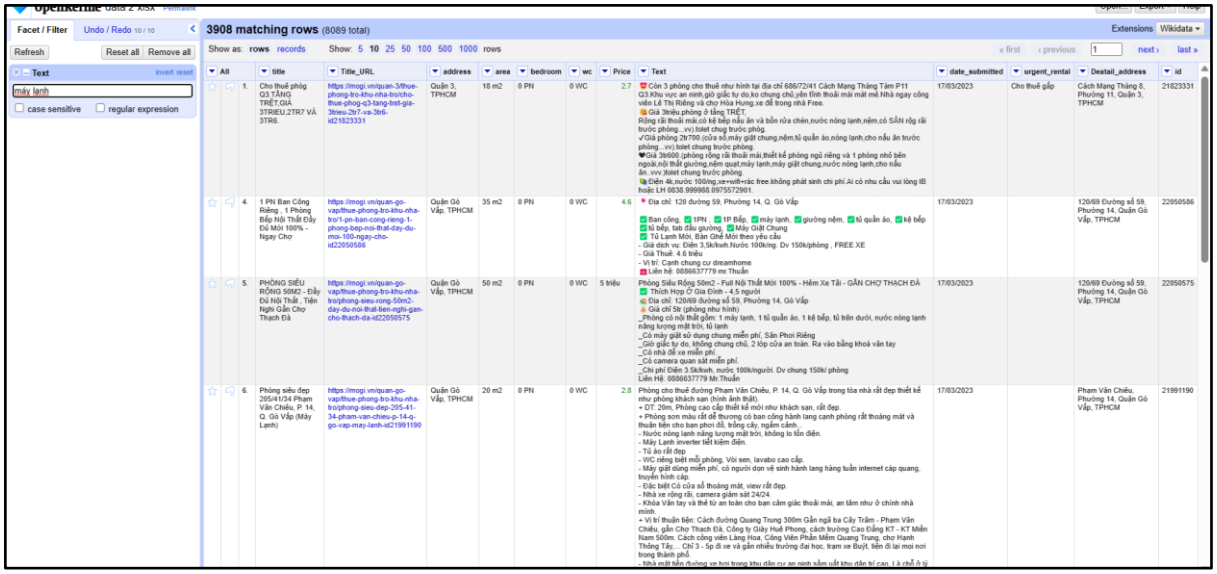


Photo 2. 10: Filter the word "air conditioner" using the text filter function in Openrefine in the Text column

After processing the entire dataset using the above methods, we have the following complete dataset:

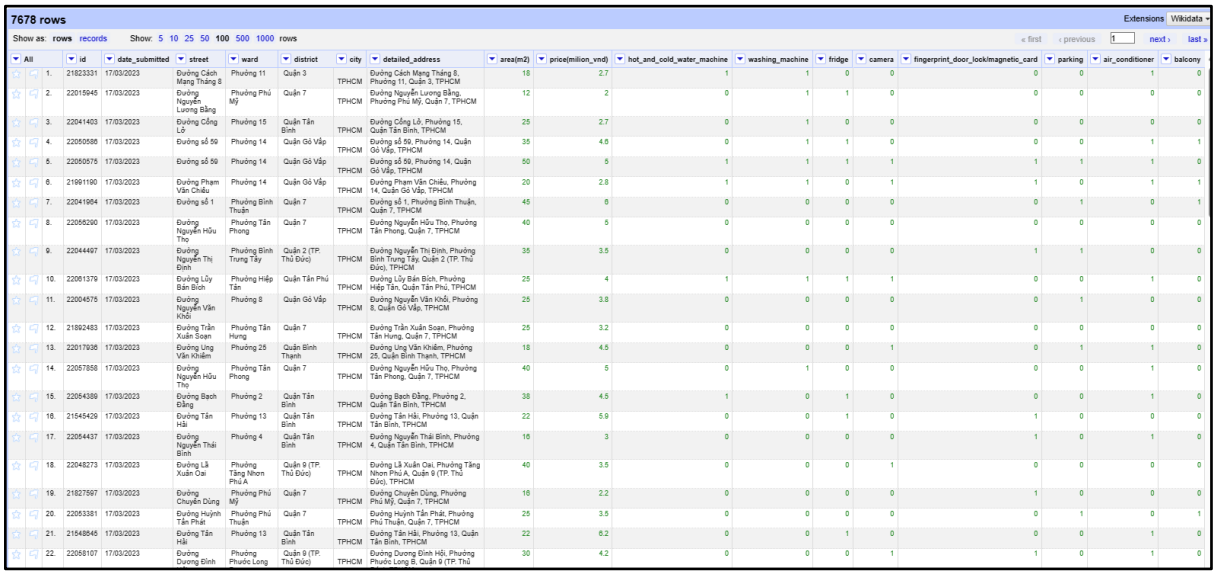


Photo 2. 11: Complete dataset after being processed by Openrefine

2.2 Data Discovery

In the EDA (Exploratory Data Analysis) section of the house price forecasting topic, we will analyze and explore the information in the data about the factors affecting house prices. By applying statistical techniques and data visualization, we will try to find the relationships and trends between variables.

One of the important factors that affect home prices is location and area, so we'll explore the distribution of locations and acreage in the data and learn the relationship between them and other variables to forecast home prices. In addition, we will also look

at other factors such as location, number of bedrooms, number of bathrooms, year of construction, and condition of the home to better understand how these factors affect home prices.

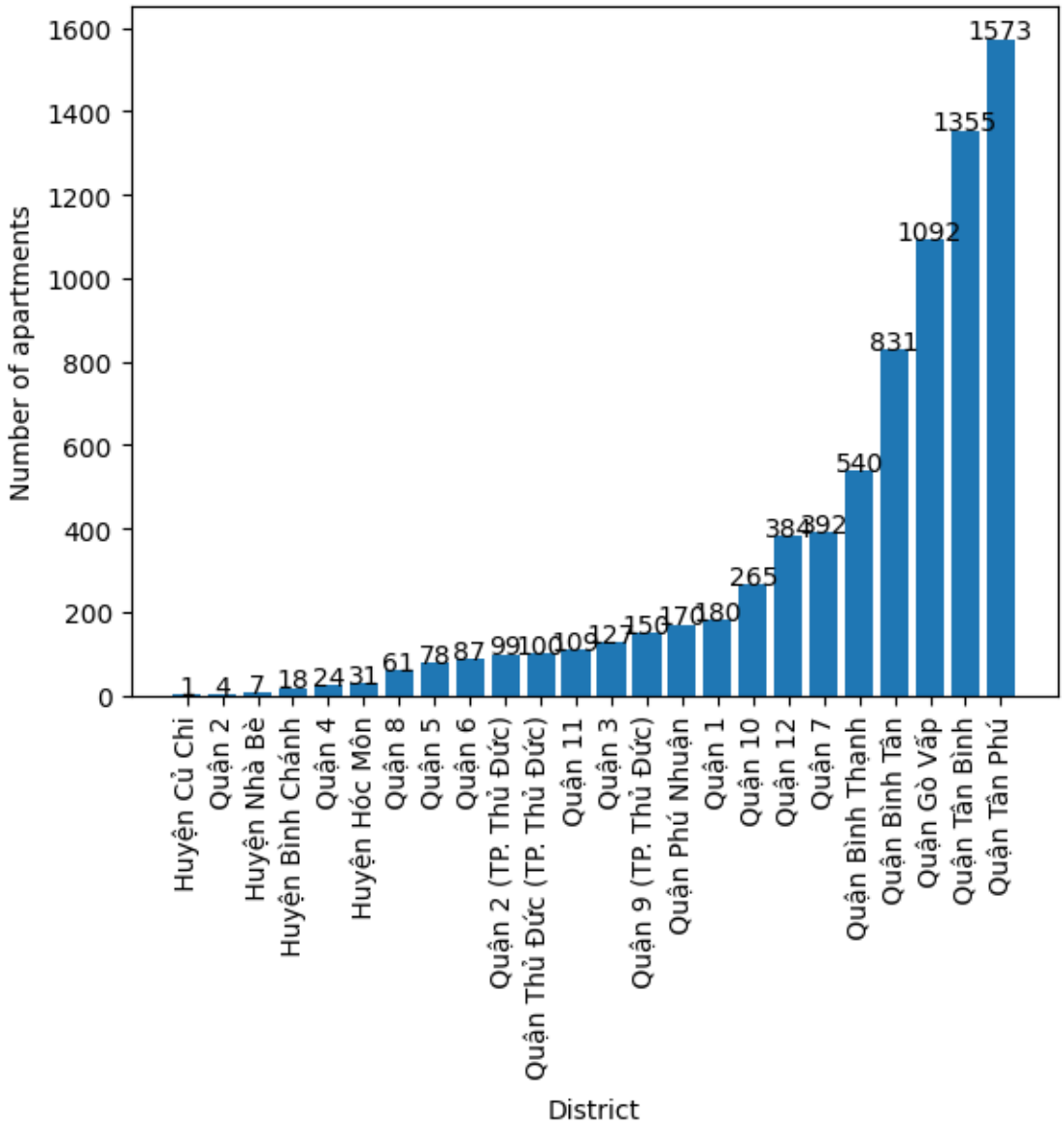


Photo 2. 12: Statistics on the number of rooms by district

From the chart above, we can see that in our data set there are 1573 houses in Tan Phu (the most), the second is 1335 houses in Tan Binh, the third is 1092 houses in Go Vap then other districts are spread out in other districts. To understand the details of the house information, come to the following chart:

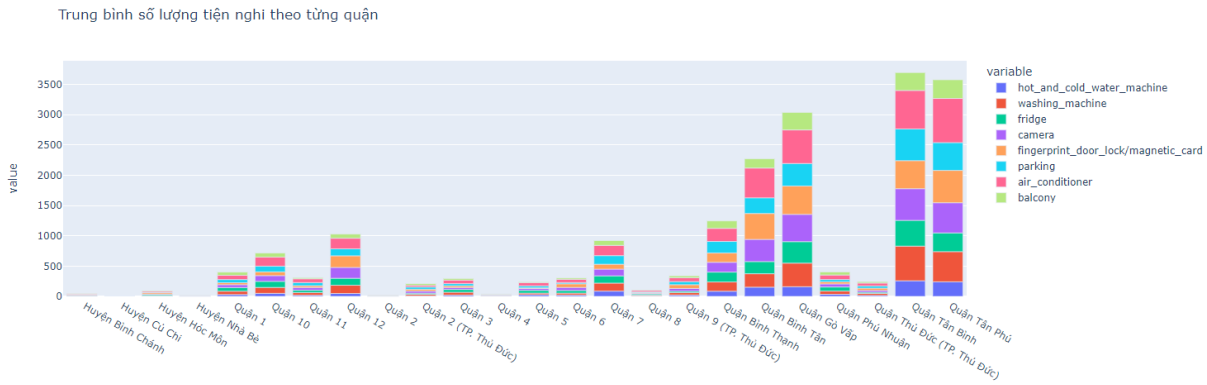


Photo 2. 13: Average number of amenities by district

In the 3 districts with the most houses as mentioned, Tan Phu, Tan Binh, Go Vap, the number of amenities has little difference. Except for hot and cold water generators in the Go Vap area, it is much less than the other 2 areas. In general, if you want to choose an area with the most amenities, you should choose Tan Binh.

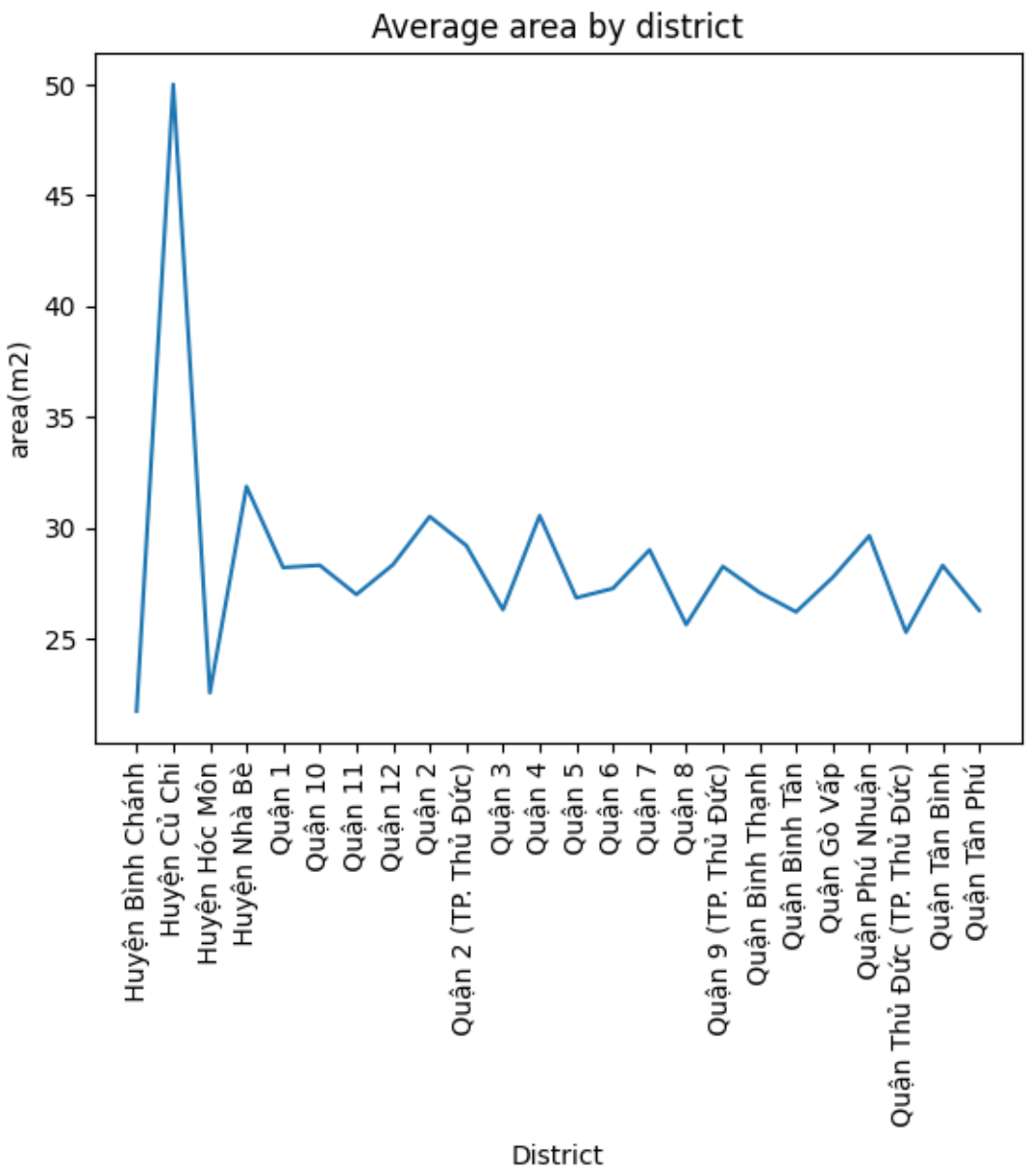


Photo 2. 14: Average apartments by district

From the chart, we can see that the area of apartments in Cu Chi occupies the highest. It may be because the houses in Cu Chi district are quite far from each other, so the land here is quite large, leading to a significantly higher apartment area here than other areas

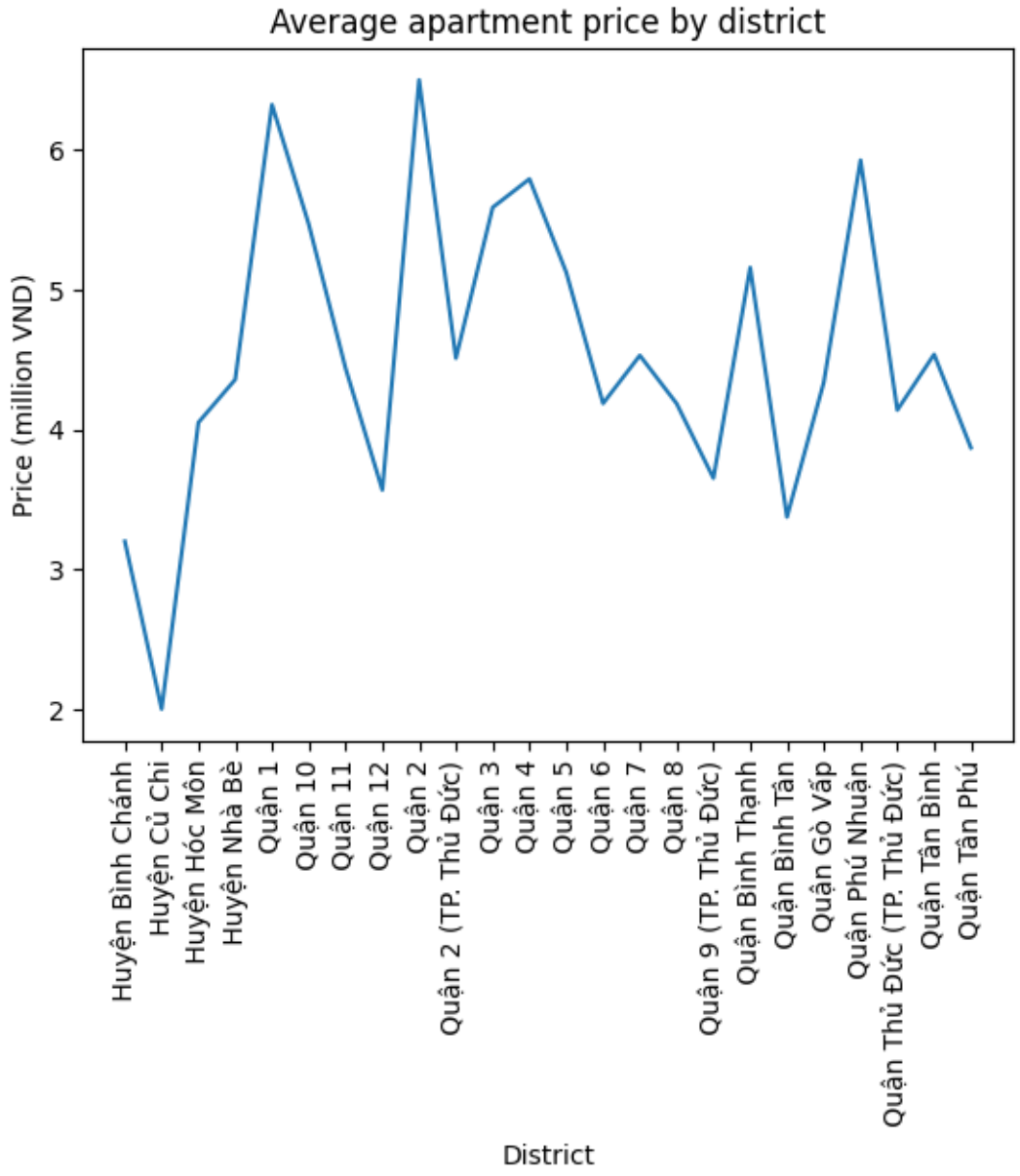


Photo 2. 15: Average house prices by district

As we can see, house prices in District 1 and District 2 are the highest of all. The reason is also easy to understand is that these 2 areas are very close to the city center, the population is concentrated, so it is obvious that high house prices are high. As I said in chart 2, if you choose, you should choose an apartment in Tan Binh office area for ordinary people.

2.3 Data Conversion

Normalization is a data preprocessing technique used to adjust the value of features in a dataset to a general scale. This is done to facilitate data analysis and modeling, and to reduce the impact of various scales on the accuracy of machine learning models.

Normalization is a scaling technique in which values are shifted and scaled so that they are between 0 and 1. It is also known as Min-Max scaling.

Here's the formula for Normalization:

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Photo 2. 16: Normalization Formula

Here, max(x) and min(x) are the largest and smallest values of the data, respectively.

- When the value of is the smallest value in the column, the numerator will be 0 and therefore $0 x_{scaled} x_{scaled}$
- On the other hand, when the value of is the largest value in the column, the numerator is equal to the denominator and therefore the value of is $1 x_{scaled} x_{scaled}$
- If the value of is between the smallest and largest values, then the value of X' is between 0 and $1 x_{scaled}$

Experiments in the article

```
# Min-Max Scaling
def min_max_scaling(data):
    """
    Apply the Min-Max Scaling method to the entire dataset

    Parameter:
        - data: pandas DataFrame or numpy array, dataset to be normalized

    Result:
        - scaled_data: pandas DataFrame or numpy array, standardized dataset
    """
    scaled_data = data.copy()

```

```
    # Browse through each column in the dataset
    for col in scaled_data.columns:
        # Calculate the min, max values of that column
        min_val = np.min(scaled_data[col])
        max_val = np.max(scaled_data[col])

        # Apply the Min-Max Scaling formula to that column
        scaled_data[col] = (scaled_data[col] - min_val) / (max_val - min_val)

    return scaled_data

# Use min_max_scaling function to normalize the entire dataset
# df is the pandas DataFrame containing the dataset's data
df_scaled = min_max_scaling(df)
df_scaled.describe()
```

Table 2. 1

Table 2.3.1: Code that performs data transformation using Min-Max Scaling

	area(m2)	price(milion_vnd)	hot_and_cold_water_machine	washing_machine	fridge	camera	fingerprint_door_lock/magnetic_card	parking	air_conditioner	balcony
count	7678.000000	7678.000000	7678.000000	7678.000000	7678.000000	7678.000000	7678.000000	7678.000000	7678.000000	7678.000000
mean	0.474445	0.256128	0.169575	0.336937	0.286403	0.362985	0.359990	0.330685	0.481115	0.216984
std	0.135198	0.093021	0.375284	0.472694	0.452109	0.480892	0.480028	0.470490	0.499676	0.412218
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.418605	0.197183	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.418605	0.232394	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.534884	0.302817	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Photo 2. 17: Statistics describing a dataset after being transformed

It can be seen that all data columns are attributed to the value range from (0; 1)

2.4 Selecting Input Variables for the Model

Selecting input variables for a model is the process of reducing the number of input variables when developing a machine learning model. Reduce the number of input variables to both reduce the computational cost of the model and to improve the performance of the model in some cases.

Statistics-based feature selection methods involve evaluating the relationship between each input variable and the target variable using statistics and selecting the

input variables that have the strongest relationship with the target variable. These methods can be quick and effective, although the choice of statistical measures depends on the type of data of both the input and output variables.

SFS (Sequential Forward Selection)

The SFS (Sequential Forward Selection) variable selection method is a variable selection technique used in the construction of machine learning models. It is a greedy method that seeks to find the most important set of variables to use in the model.

Specifically, the SFS method starts with an empty set of variables and from there looks for the next variable to add to that set. When a new variable is added to the set, a model is retrained and evaluates the model's performance with the new variable added. If adding a new variable improves the model's performance, it is retained in the set. This process is repeated until no other variables are added to the set, or adding a new variable does not improve the model's performance.

The SFS method is often used to reduce the number of variables to be used in the model and increase the interpretability of the model. It can also help improve the performance of the model by using only the most important variables. However, this method can also cause problems such as overfitting and not finding the best set of variables.

Experiments in the article

We choose the Forward Feature Selection (FS) method in Sequential Forward Selection to perform:

```
# Create original dataset with 10 features and 1000 templates
X, y = df_scaled.drop(['price(milion_vnd)'], axis=1), df_scaled['price(milion_vnd)']

# Random Forest model initialization
rfc = RandomForestRegressor(n_estimators=100, random_state=0)

# SFS Initialization
sfs = SequentialFeatureSelector(rfc, direction='forward', cv=5)

# Model Training with SFS
sfs.fit(X, y)
```

```

# Create a new dataset from selected features
X_sfs = sfs.transform(X)
selected_features = X.columns[sfs.get_support()]
df = pd. DataFrame(X_sfs, columns=selected_features)
df['target'] = y

# Print out a new dataset with selected features
print(df.head())
```

Table 2. 2: Forward Feature Selection code

```
Index(['area(m2)', 'washing_machine', 'fridge', 'air_conditioner'], dtype='object')
```

id	price(milion_vnd)	area(m2)	washing_machine	fridge	air_conditioner
21823331	0.140845	0.255814	1.0	0.0	1.0
22015945	0.091549	0.116279	1.0	1.0	0.0
22041403	0.140845	0.418605	1.0	0.0	0.0
22050586	0.274648	0.651163	1.0	1.0	1.0
22050575	0.302817	1.000000	1.0	1.0	1.0
...	...	...	...	...	...
21809932	0.211268	0.418605	1.0	0.0	0.0
21844927	0.232394	0.302326	1.0	0.0	0.0
21937596	0.267606	0.534884	0.0	0.0	0.0
21901169	0.302817	0.418605	0.0	1.0	0.0
21825160	0.204225	0.302326	0.0	0.0	0.0

7678 rows × 5 columns

Photo 2. 18: Data after feature selection

3. Build ML Models

3.1 Model training

3.1.1 Decision Tree

3.1.1.1 Decision Tree - Regression

A regression decision tree algorithm is a machine learning algorithm used to build a model that predicts a continuous output value based on input features. This algorithm uses a decision tree to model the relationship between input and output.

Decision trees are built through the process of dividing input data into subsets so that these subsets are as separate and homogeneous as possible in terms of output value. This division process is done by selecting a feature and a threshold so that the separation of output values between the two sub-branches of the tree is the largest.

These decisions are repeated until all the leaves of the plant contain almost uniform output values. Then, the output value of a new data point will be predicted by going from the root node of the tree to a corresponding leaf and taking the average of the output values in that leaf as the predicted value.

The parameters that need to be set for the regression decision tree algorithm include the maximum number of leaf nodes, the maximum depth of the tree, and the minimum number of data points required in each node to continue the division.

3.1.1.2 Decision Tree Algorithm

The core algorithm for building a decision tree called ID3 by J. R. Quinlan, uses greedy, top-down search in the space of possible branches without going back. The ID3 algorithm can be used to construct a decision tree for regression by replacing the Information Gain with a Standard Deviation Reduction.

3.1.1.3 Standard deviation

Decision trees are built from the top down from the root node and involve partitioning the data into subsets containing representations of similar (homogeneous) values. We use the standard deviation to calculate the uniformity of a denominator. If the denominator is completely homogeneous, then its standard deviation is zero.

3.1.1.4 Decision Tree Construction

The problem is that now that we have the data, how to build a decision tree.

Let's say I have a 2-class classification problem and each data has 2 properties, x1 and x2. My data when plotting the scatter chart up will look like this.

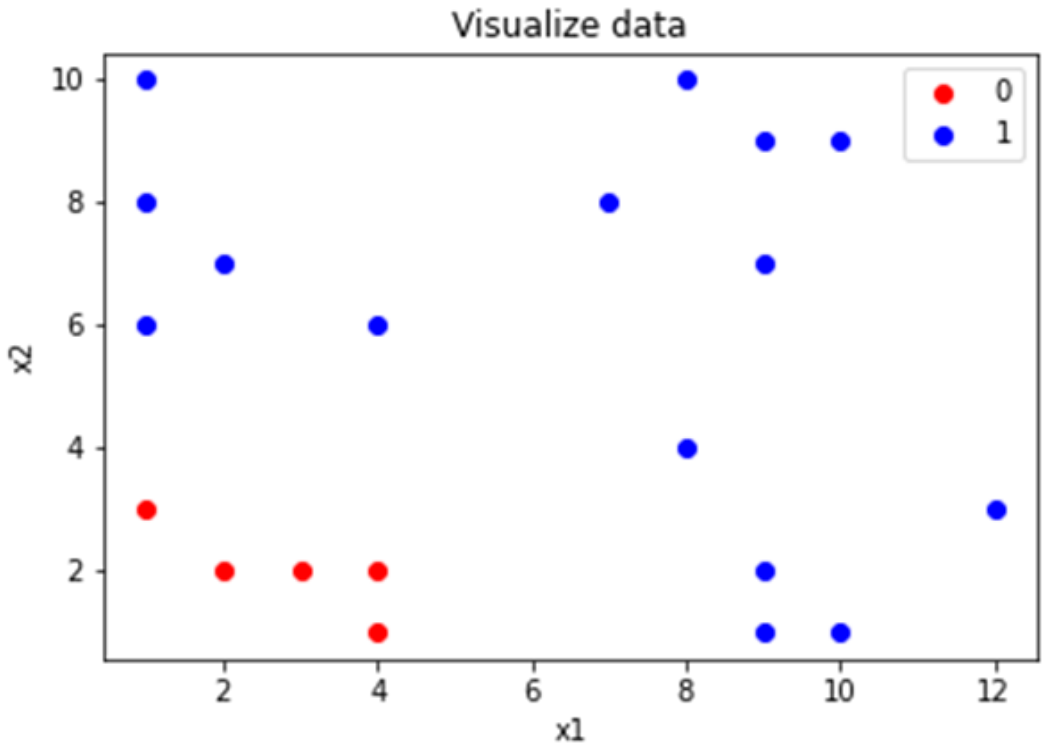


Photo 3. 1: Decision Tree Visualization 1

With this data, if you ask people to use pen and paper to draw trees, what will everyone do?

Consider the condition x1 > 5, like a dividing line, dividing the data into 2 parts, 1 part satisfies the condition and 1 part does not satisfy the condition.

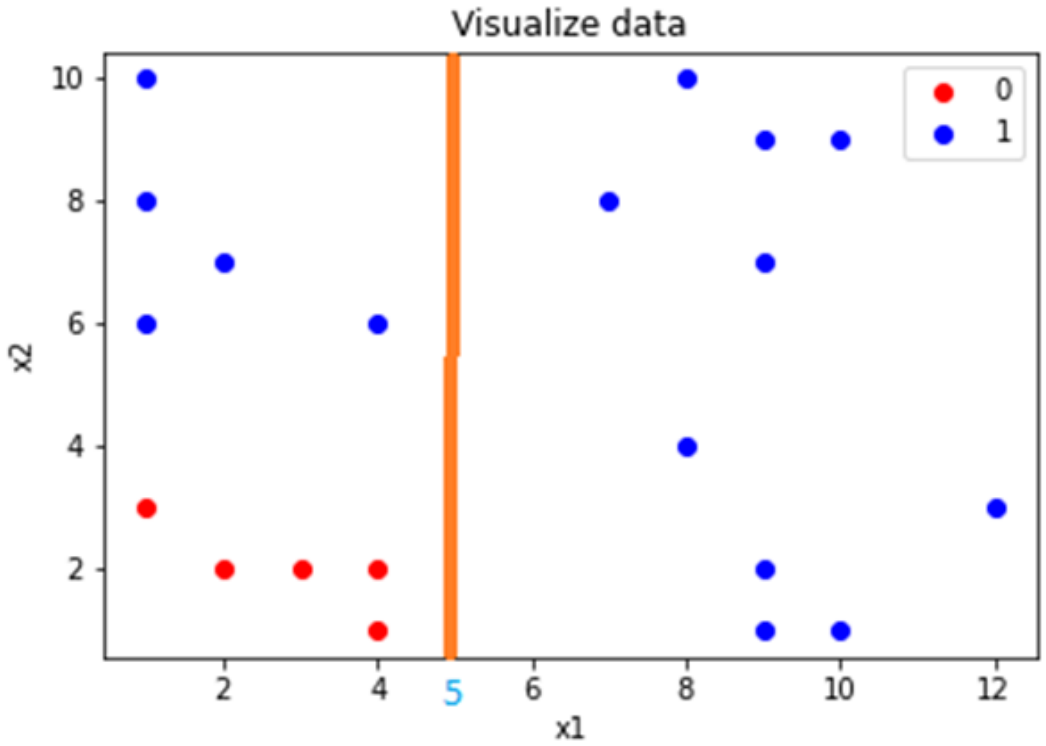
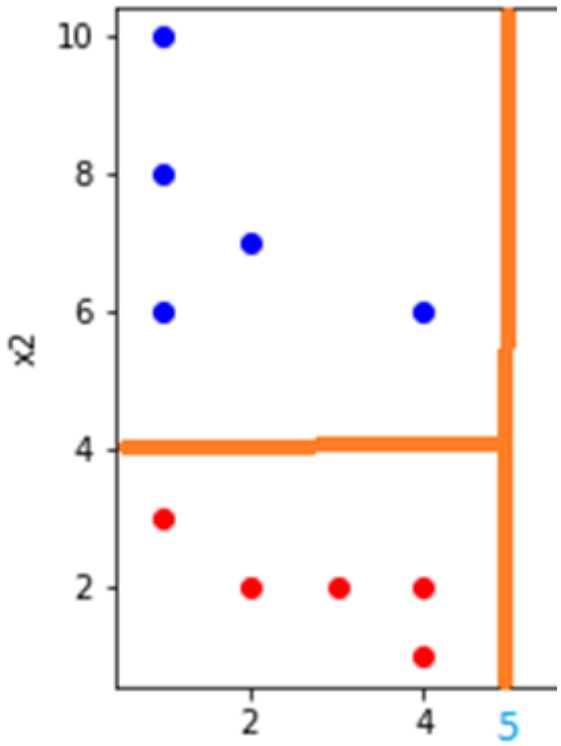


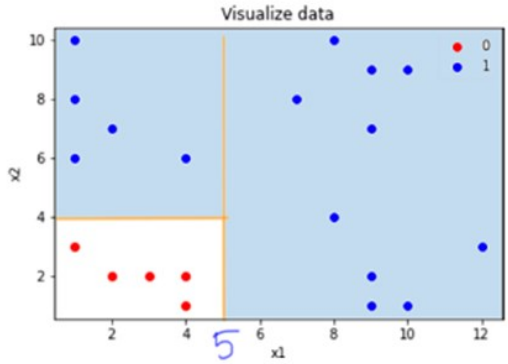
Photo 3. 2: Decision Tree Visualization 3

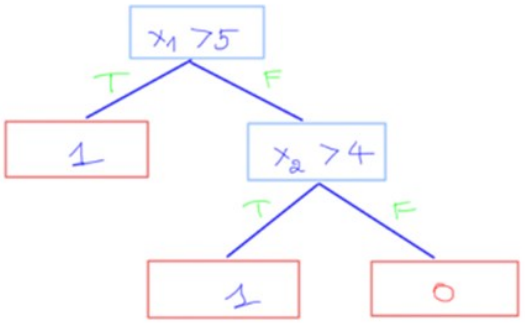
I see that if x1 > 5 are correct, then all the data is in class 1, so I will use the leaf layer to predict that this is class 1. On the contrary, I see that the data has both class 1 and class 0, so I continue to add conditions x2 > 4



If the conditions x2 > 4 are correct, I see that the data belongs to class 1, the opposite is to the data of class 0. Therefore, the 2 sub-nodes of the above condition node are both leaf nodes to produce the prediction result.

In the end, I will be decided by the tree like this.





So what are the criteria for me to find the first condition? Why is it x1 and why is it 5 and not some other number? If you pay attention to the above, we will create conditions to separate the data into 2 parts, where each part of the data is more separate than the original data. For example, condition x1 > 5, at the correct branch, all elements belong to class 1.

- So the condition x1 > 8 is also branched into the whole 1st grade, why not choose? Since the true branch at conditions x1 > 5 contains more class 1 elements and is more general than the correct label of x1 > 8
- As for the x1 > 2 condition, both subbranches contain data from both class 0 and class 1.

I already know the criteria to choose, but the criteria I have just set are based on the eyes and senses, the computer needs data to evaluate to compare separation conditions. Indicators to evaluate were born, including: **entropy, information gain**.

For each condition for separation, there will be a corresponding information gain index, the higher the information gain index, the better the separation. Therefore, I will browse through all the attributes of the data, each attribute tries different values to separate, then choose the condition with the highest information gain index to separate, and so on until the leaf node, which consists of only 1 layer of data.

In addition to ID3, there are other algorithms for Decision Trees such as:

- **C4.5**: Successor of ID3
- **CART**: Classification And Regression Tree
- **CHAID**: Chi-square automatic interaction detection Performs multi-level splits when computing classification trees
- **MARS**: multivariate adaptive regression splines

class sklearn.tree.DecisionTreeRegressor(*, criterion='squared_error', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha=0.0)

3.1.1.5 Parameters

criterion{“squared_error”, “friedman_mse”, “absolute_error”, “poisson”}, default=”squared_error”

Function to evaluate the quality of a division. Supported criteria include "squared_error" for the mean squared error, which is equivalent to reducing the variance as a feature selection criterion and minimizing L2 losses using the average of each end node, "friedman_mse", using the mean square error with Friedman's improvement score for potential divisions, "absolute_error" gives the absolute mean error, minimizes the L1 loss using the median of each end node, and "poisson" uses the Poisson error reduction to find the divisions.

splitter{“best”, “random”}, default=”best”

The function is used to select the division at each node. "Best" to choose the best split and "random" to choose the best random split.

max_depthint, default=None

Maximum depth of the tree. If not, then the buttons are extended until all the leaves are clean or until all the leaves contain less than min_samples_split samples.

min_samples_splitint or float, default=2

Minimum number of templates required to split an internal node:

- If int, then consider min_samples_split as the minimum.
- If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) is the minimum number of samples for each split.

random_stateint, RandomState instance or None, default=None

Control the randomness of the estimator. Features are always randomly permuted at each split, even if the splitter is set to "best". When max_features < n_features, the algorithm randomly selects max_features at each split before figuring out the best of them. But the best split found can vary between different runs, even if max_features=n_features. That is the case, if the improvement of the criteria is identical for several divisions and one division must be chosen at random. To obtain a definite behavior during adjustment, Random_state must be fixed to an integer.

3.1.2 Random Forest

Random Forest is a decision tree-based supervised machine learning algorithm. The algorithm is used for both classification and regression tasks. The goal of Random Forest is to create a forest of decision trees trained on different subsets of training data and features, and then combine the predictions of each tree to make the final prediction.

Theory:

The algorithm generates multiple decision trees by selecting a random subset of features and data. It then selects the best split points to create subnodes until the stop criteria are met. This process is repeated for each tree. When the forest is built, predictions are made by passing new data through all the trees in the forest and averaging the results.

The main advantage of Random Forest is that it can handle height datasets with a large number of features and can handle missing data. Additionally, Random Forest is relatively sensitive to hyperparameter selection and can provide estimates of the importance of the feature.

Applications:

Random Forest has a wide range of applications in both academia and industry. Some of its applications include:

Image Classification: Random Forest has been used to classify images in a number of fields, including medicine and biology. For example, it has been used to classify cancer cells from microscopic images and to identify birds from images.

Fraud Detection: Random Forest has been used to detect fraudulent activities in financial transactions. It can detect suspicious patterns in the data and flag them for further investigation.

Customer segmentation: Random Forest has been used to segment customers based on their purchasing behavior, demographics, and other factors. This information can be used to develop targeted marketing strategies.

Predictive Maintenance: Random Forest has been used to predict equipment failures in industrial environments. By analyzing data from sensors and other sources, it can predict when maintenance is needed before a fault occurs.

Benefits:

Random Forest offers several benefits over other machine learning algorithms:

Powerful: Random Forest is powerful against foreign values, noise, and lack of data. It can handle datasets with missing values by assigning them mean or mean values.

Feature selection: it is possible to identify important features that contribute to the prediction. This information can be used to select a subset of the features that are most relevant to the issue.

Interpretability: Random Forest is relatively easy to interpret compared to other machine learning algorithms. Feature importance scores can provide insights into the relationship between input features and goal variables.

Scalability: Random Forest can handle large data sets and can run in parallel to run on multiple processors. This makes it suitable for big data applications.

In conclusion, Random Forest is a powerful and flexible machine learning algorithm that can be used for a wide range of applications. Its ability to handle height datasets, missing data, and feature selection make it a valuable tool for data analysis. Its robustness, interpretability, and scalability make it a popular choice for both academia and industry.

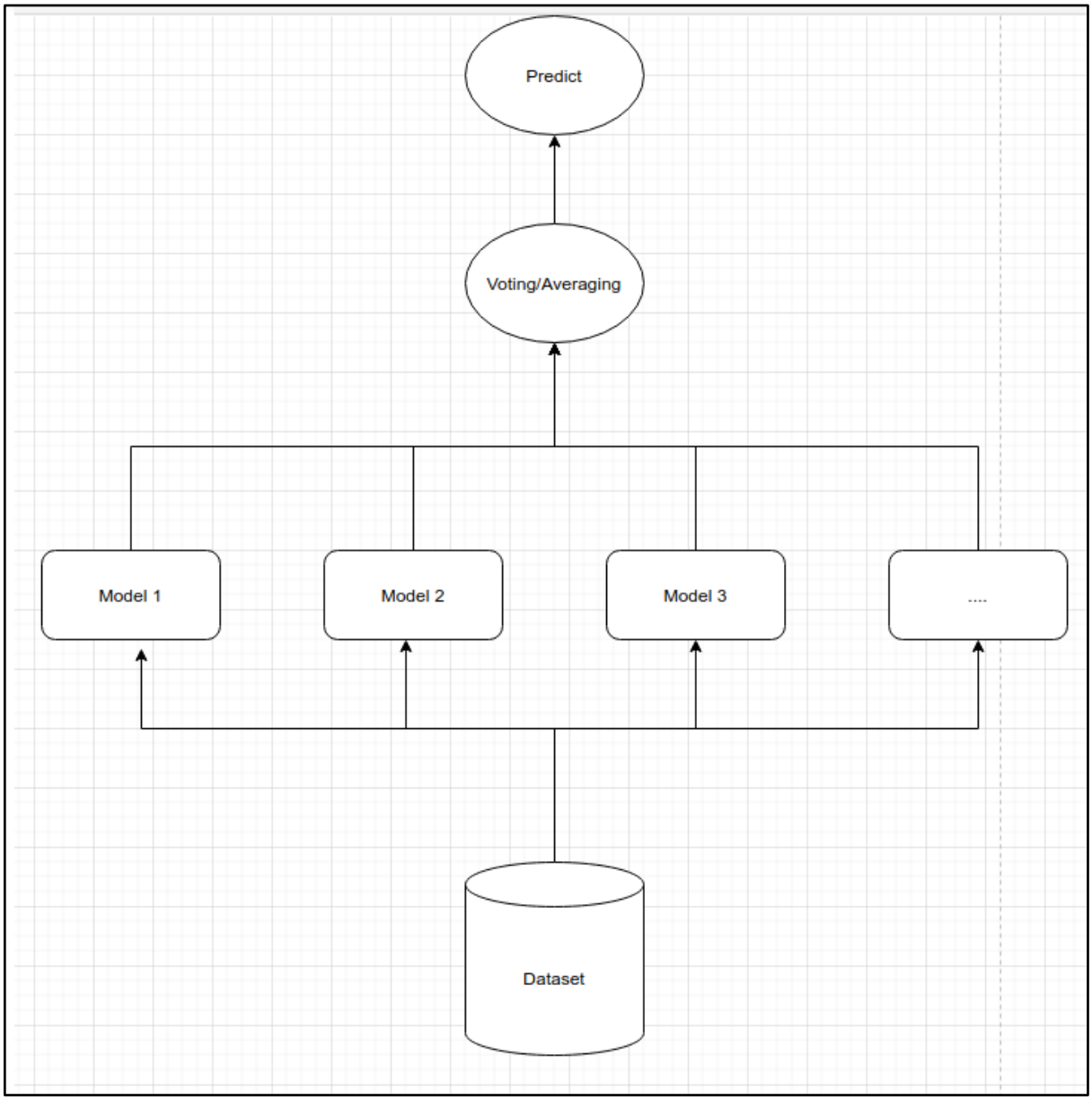
3.1.2.1 Ideas of the forest model

The forest model is trained based on a combination of ensembling and boostrapping. Specifically, this algorithm generates multiple decision trees, each of which is trained based on different subsamples, and the predicted result is voting from all decision trees. Thus, a forecast result is synthesized from many models, so their results will not be deviated. At the same time, combining forecast results from multiple models will have a smaller variance than just one model.

3.1.2.2 Ensemble model

Let's assume that you're building a binary classification model for dog and cat photos that correspond to two labels, 0 and 1, respectively. For a specific image, if only a single model is used, the predicted result has a probability of belonging to the cat label is only 0.6. This is a probability that is not too high so you are not sure that your image is a cat.

Because it's uncertain, you want to consult the results from more models. That's why you decide to build 9 different models and conduct an election of the results returned between them. Because this is a difficult case to detect, for example, the photo is blurred and the animal is hiding under a tree, the models forecast a probability of not too close to 1. But surprisingly, in the results returned from 9 models, there are 8 models that forecast label 1 and 1 model that forecasts label 0. Thus, based on the election results, you can be confident that the forecast label for the photo is a cat is correct.




```
 from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
```

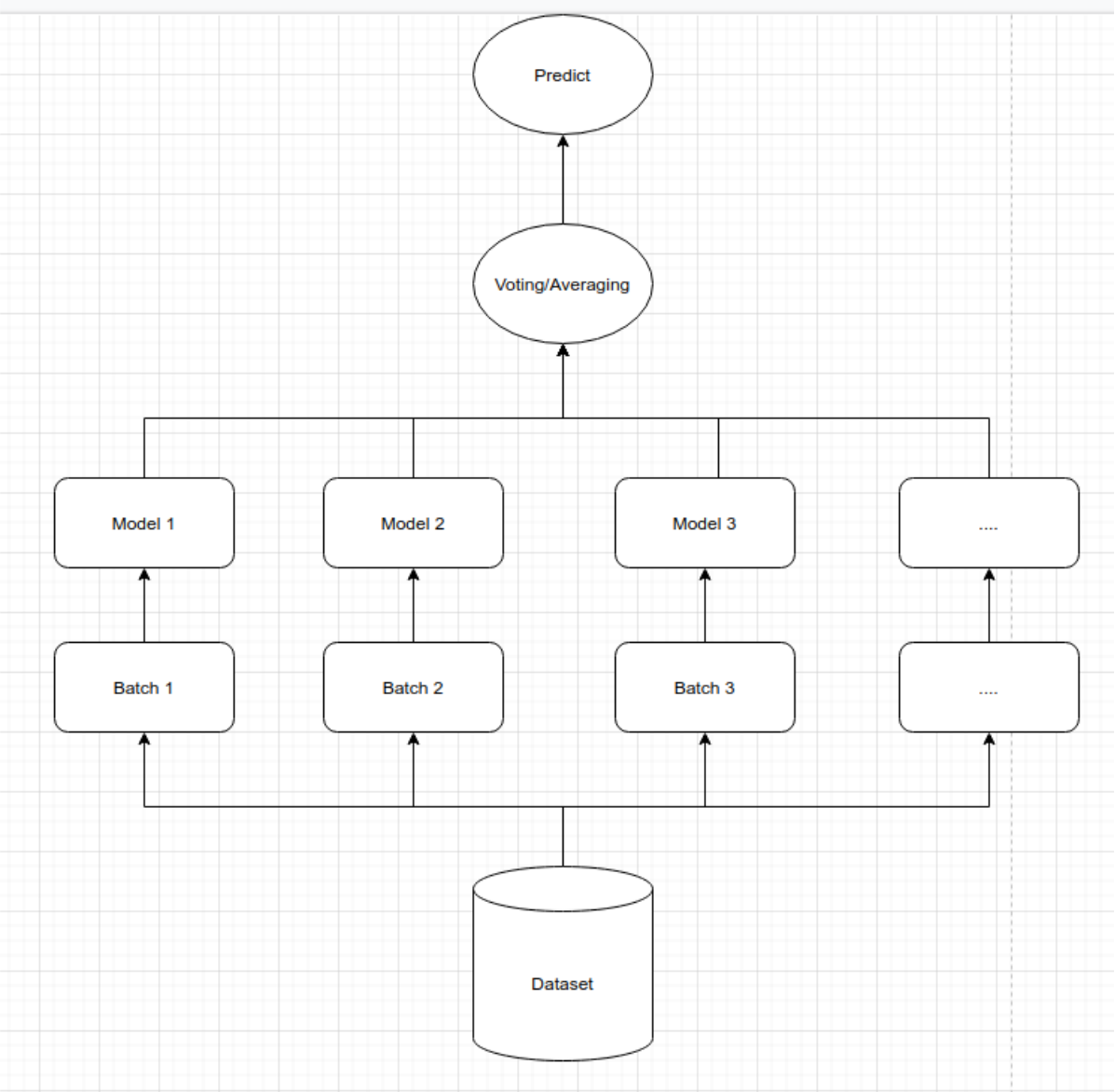
```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from sklearn.datasets import make_blobs
import numpy as np

# Load the dataset
iris = load_iris()
X = iris.data
y = iris.target == 1

# Three model in ensemble learning
log_clf = LogisticRegression()
svm_clf = SVC()
tree_clf = DecisionTreeClassifier(max_depth=3)

voting_clf = VotingClassifier(
 estimators=[('lr', log_clf), ('svc', svm_clf), ('tree_clf', tree_clf)],
 voting='hard'
)

cv = RepeatedStratifiedKFold(n_splits=3, n_repeats=5,
random_state=1)

# Evaluate the model on each single model
scores = cross_val_score(log_clf, X, y, scoring='accuracy', cv=cv,
n_jobs=-1)
print('Logistic Regression Mean Accuracy:
{:.03f}'.format(np.mean(scores)))
scores = cross_val_score(svm_clf, X, y, scoring='accuracy', cv=cv,
n_jobs=-1)
print('SVM Mean Mean Accuracy: {:.03f}'.format(np.mean(scores)))
scores = cross_val_score(tree_clf, X, y, scoring='accuracy', cv=cv,
n_jobs=-1)
print('DecisionTree Classifier Mean Accuracy:
{:.03f}'.format(np.mean(scores)))


```

```
    # Evaluate the model on each single model
    scores = cross_val_score(log_clf, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
    print('Logistic Regression Mean Accuracy: {:.03f}'.format(np.mean(scores)))
    scores = cross_val_score(svm_clf, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
    print('SVM Mean Mean Accuracy: {:.03f}'.format(np.mean(scores)))
    scores = cross_val_score(tree_clf, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('DecisionTree Classifier Mean Accuracy: {:.03f}'.format(np.mean(scores)))
```

```
Logistic Regression Mean Accuracy: 0.713
SVM Mean Mean Accuracy: 0.941
DecisionTree Classifier Mean Accuracy: 0.949
# Evaluate the model on the combined model
q
    scores = cross_val_score(voting_clf, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
    print('Voting Classifier Mean Accuracy: {:.03f}'.format(np.mean(scores)))
Voting Classifier Mean Accuracy: 0.949
```

3.1.2.3 Boostrapping

Giả định dữ liệu huấn luyện mô hình là một tập $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ bao gồm N quan sát. Thuật toán *rừng cây* sẽ sử dụng phương pháp *lấy mẫu tái lập* để tạo thành B tập dữ liệu con. Quá trình *lấy mẫu tái lập* này còn gọi là bỏ túi (*bagging*). Tức là chúng ta sẽ thực hiện M lượt nhặt các mẫu từ tổng thể và bỏ vào túi để tạo thành tập $\mathcal{B}_i = \{(x_1^{(i)}, y_1^{(i)}), (x_2^{(i)}, y_2^{(i)}), \ldots, (x_M^{(i)}, y_M^{(i)})\}$. Tập $\mathcal{B}_i$ cho phép các phần tử được lặp lại. Như vậy sẽ tồn tại những quan sát thuộc $\mathcal{D}$ nhưng không thuộc $\mathcal{B}_i$. Đây là những quan sát chưa được bỏ vào túi và chúng ta gọi chúng là *nằm ngoài túi* (*out of bag*).


Predict
Voting/Averaging
Model 1
Model 2
Model 3
....
Batch 1
Batch 2
Batch 3
....
Dataset

Với mỗi tập dữ liệu $\mathcal{B}_i$ chúng ta xây dựng một mô hình *cây quyết định* và trả về kết quả dự báo là $\hat{y}_j^{(i)} = f_i(\mathbf{x}_j)$. Trong đó $\hat{y}_j^{(i)}$ là dự báo của quan sát thứ j từ mô hình thứ (i), $\mathbf{x}_j$ là giá trị véc tơ đầu vào, $f_i(.)$ là hàm dự báo của mô hình thứ i. Mô hình dự báo từ cây quyết định là giá trị trung bình hoặc bầu cử của B cây quyết định.

- Đối với mô hình dự báo: Chúng ta tính giá trị trung bình của các dự báo từ mô hình con.

$$\hat{y}_j = \frac{1}{B}\sum_{i=1}^{B}\hat{y}_j^{(i)}$$

- Đối với mô hình phân loại: Chúng ta thực hiện *bầu cử* từ các mô hình con để chọn ra nhãn dự báo có tần suất lớn nhất.

$$\hat{y}_j = \arg\max_c \sum_{i=1}^{B} p(\hat{y_j}^{(i)} = c)$$

Như vậy phương sai của mô hình trong trường hợp đối với bài toán dự báo:

$$\begin{aligned}\sigma_{\hat{y}}^2 &= \mathrm{Var}(\frac{1}{B}\sum_{i=1}^{B}\hat{y}^{(i)}) \\ &= \frac{1}{B^2}[\sum_{i=1}^{B}\mathrm{Var}(\hat{y}^{(i)}) + 2\sum_{1\le m<n\le B}\mathrm{cov}(y^{(m)}, y^{(n)})]\end{aligned}$$

Do kết quả của mô hình con A không chịu ảnh hưởng hoặc phụ thuộc vào mô hình con B nên ta có thể giả định kết quả dự báo từ các mô hình là hoàn toàn độc lập nhau. Tức là ta có $\mathbf{cov}(y^{(m)}, y^{(n)}) = 0, \ \forall 1 \le m < n \le B$. Đồng thời giả định chất lượng các mô hình là đồng đều, được thể hiện qua phương sai dự báo là đồng nhất $\mathrm{Var}(\hat{y}^{(i)}) = \sigma^2, \ \forall i = \overline{1, B}$. Từ đó suy ra:

$$\begin{aligned}\sigma_{\hat{y}}^2 &= \frac{1}{B^2}[\sum_{i=1}^{B}\mathrm{Var}(\hat{y}^{(i)})] \\ &= \frac{1}{B^2}B\sigma^2 = \frac{1}{B}\sigma^2\end{aligned}$$

Thus, if the forecast is used as a combined average from many decision tree models, the variance can be reduced B times compared to using only one model. In a forest model, the number of trees is huge. Therefore, the forecast variance from the model can be reduced many times and produce a more stable forecast.

On sklearn we use the sklearn.ensemble.BaggingClassifier module to apply the pocket algorithm. Below is an example of the pocketing process using 200 different decision tree models. Each model is built on 100 randomly selected input data samples. When we select boostrap=True, the model will use a repeating sampling method, whereas the sampling process does not allow observations to be repeated (also known as pasting sampling).

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
bag_clf = BaggingClassifier(
```

```
    DecisionTreeClassifier(),
    n_estimators=200,
    max_samples=100,
    bootstrap=True,
    n_jobs=-1
)

bag_clf.fit(X, y)

scores = cross_val_score(bag_clf, X, y, scoring='accuracy', cv=cv,
n_jobs=-1)
print('Logistic Regression Mean Accuracy:
{:.03f}'.format(np.mean(scores)))
```

```
Logistic Regression Mean Accuracy: 0.949
```

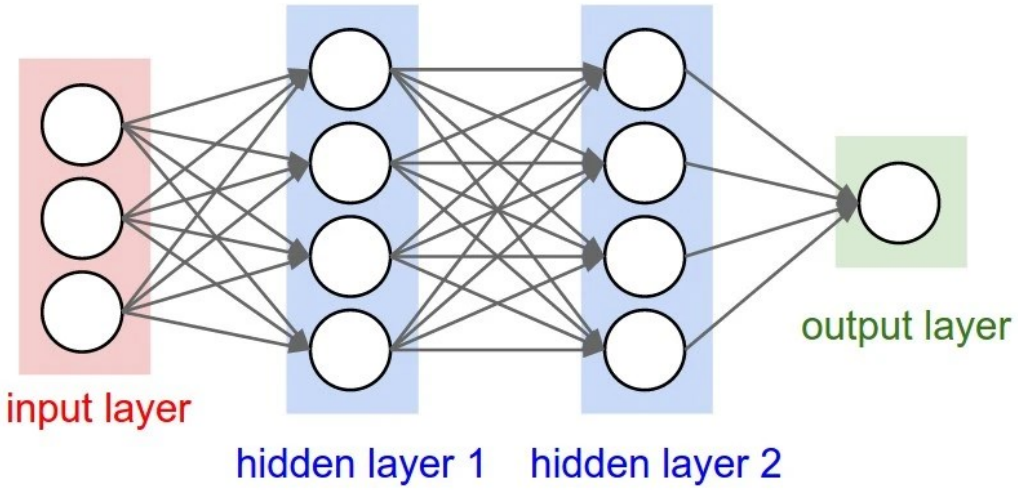
We can see that when using pockets, the accuracy of the model has improved by 0.002 points compared to using only the combined model on a single dataset.

3.1.3 Neural Networks

3.1.3.1 Artificial Neural Networks

Artificial neural networks contain artificial neurons called units. These units are arranged in a series of layers together to form the entire Artificial Neural Network in a system. A class can only have dozens or millions of units because this depends on how complex neural networks will be required to learn hidden patterns in the dataset. Typically, Artificial Neural Networks have an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world that the neural network needs to analyze or learn. This data then goes through one or more hidden layers to convert the input into valuable data for the output layer. Finally, the output layer provides the output in the form of the Artificial Neural Network's response to the input data provided.

In the majority of neural networks, units are connected to each other from layer to layer. Each of these connections has weights that determine the effect of one unit on another. As data travels from one unit to another, the neural network learns more and more about the data, which ultimately leads to output from the output layer.



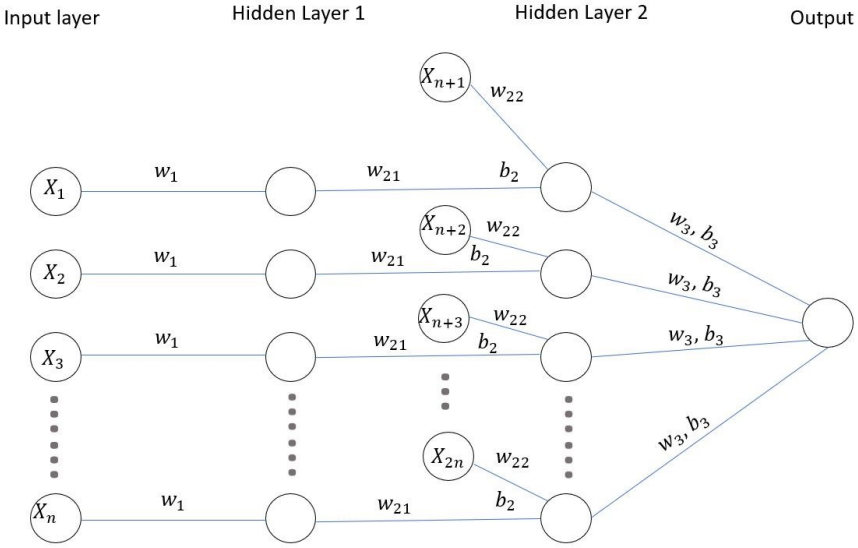
The structure and activity of human neurons are the basis for artificial neural networks. It is also known as a neural network or neural network. The input layer of the artificial neural network is the first layer, and it receives input from external sources and releases it into the hidden layer, which is the second layer. In the hidden layer, each neuron receives input from the previous layer of neurons, calculates the total weight, and sends it to the neurons in the next layer. These connections are weighted, meaning that the effect of the inputs from the previous layer is more or less optimized by assigning different weights to each input, and it is adjusted during training by optimizing these weights to improve model performance.

3.1.3.2 How Neural Networks Work

Artificial neural networks are trained using a training set. For example, let's say you want to teach ANN to recognize a cat. It is then shown thousands of different images of the cats so that the network can learn how to identify a cat. Once the neural network has been sufficiently trained using the cat's image, then you need to check if it can accurately identify the cat's image. This is done by making the ANN categorize the images it provides by deciding whether or not they are cat images. The output obtained by ANN is corroborated by a human-provided description of whether the image is a cat image or not. If the ANN is incorrectly identified, backpropagation is used to correct whatever it has learned during training. Reverse propagation is performed by fine-tuning the weighting of the connections in the ANN unit based on the error rate obtained. This process continues until an artificial neural network can accurately recognize a cat in a photo with the minimum possible error rate.

a) Weights

Weights are parameters in neural networks that convert input data in the hidden layers of the network. A neural network is a series of nodes or neurons. Within each node is a set of inputs, weights, and skew values. When an input enters the node, it is multiplied by a weighted value and the output is observed or transferred to the next layer in the neural network. Normally, the weights of the neural network are contained in the hidden layers of the network.



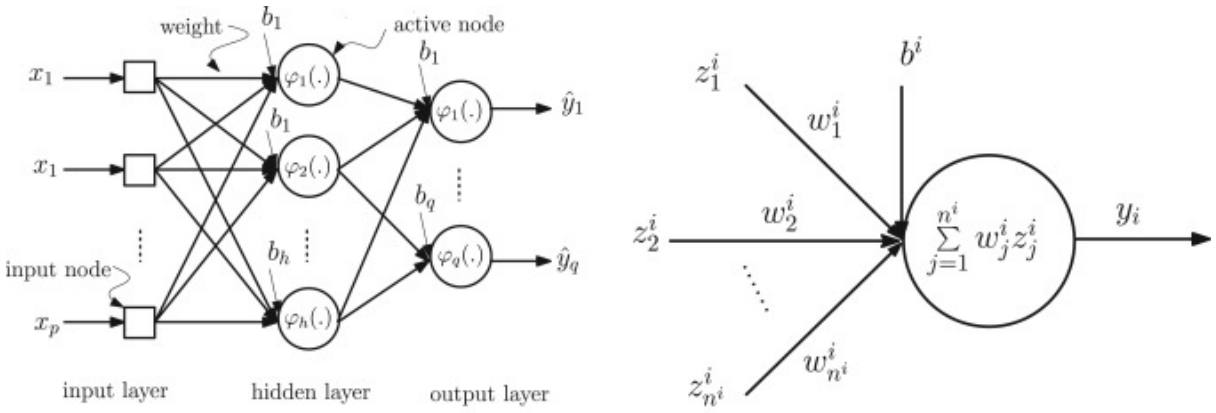
Total of 4 weights and 2 biases (in the whole network) : w_1, w_{21}, w_{22}, w_3, b_1, b_2 are all shared.

In the neural network there is an input layer, which receives the input signals and transfers them to the next layer.

Next, the neural network contains a series of hidden layers that apply transformations to the input data. Weights are applied in the buttons of the hidden layers. For example, a single node can take the input data and multiply it by a specified weight value, then add a deviation before moving the data to the next layer. The last layer of the neural network is also known as the output layer. The output layer usually adjusts the inputs from the hidden layers to produce the desired numbers within a defined range.

b) Feed Forward Neural Network

A feed-forward neural network is an artificial neural network in which the connections between nodes do not form a cycle. In contrast to the feed-forward neuron network is a regression neural network, in which certain paths are recycled. The feed forwarding model is the simplest form of neural network because the information is processed in only one direction. Although the data can pass through many hidden nodes, it always moves in one direction and never back.



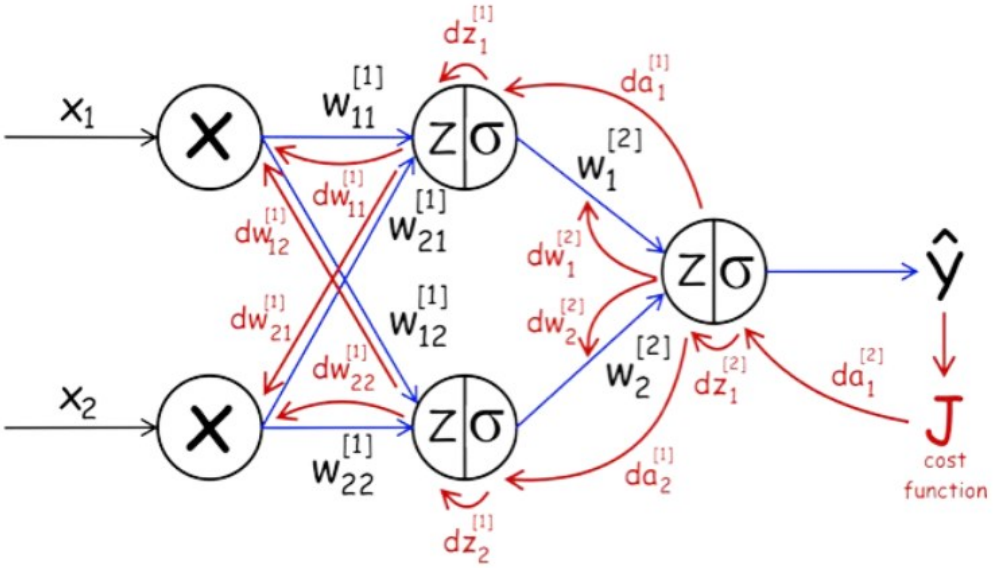
(a) Three-layer feedforward neural network (b) Node of the network

Feed-forward neural networks are often seen in their simplest form as a single-layer perceptual. In this model, a series of inputs are inserted into the class and multiplied by weights. Each value is then added together to have the sum of the weighted input values. If the sum of the values is higher than a specific threshold, which is usually set to 0, the value generated is usually 1, while if the sum is below the threshold, the output value is -1. Single-layer perceptrons are an important model of feed-forward neuron networks and are commonly used in classification tasks. Furthermore, single-layer perceptions can incorporate aspects of machine learning. Using an attribute known as the delta rule, the neural network can compare the output of its nodes to the intended values, thus allowing the network to adjust its weights through training to produce more accurate output values. This training and learning process creates a kind of downward slope. In multilayer perceptrons, the process of updating weights is almost similar, however, this process is more specifically defined as reverse propagation. In such cases, each hidden layer in the network is adjusted according to the output values generated by the last layer.

c) Backpropagation

Backpropagation, which stands for error backpropagation, is a widely used method for calculating derivatives within deep forward neural networks. Backpropagation forms an important part of some supervised learning algorithms for training straight-line neural networks, such as decreasing the random slope.

When training a neural network by decreasing the slope, a loss function is calculated, denoting the predicted distance of the network from the real label. Backpropagation allows us to calculate the slope of the loss function for each weight of the network. This allows each weight to be updated individually to gradually reduce lost functionality over multiple training iterations.



Backpropagation

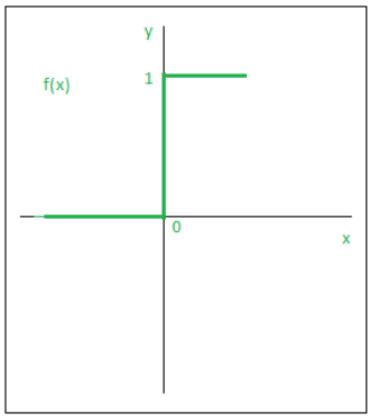
Backpropagation involves the calculation of the gradient proceeding backwards through the feedforward network from the last layer to the first layer. To calculate the slope at a particular layer, the slope of all subsequent layers is combined through the sequence rule of the calculation.

d) Active function

Step function: is one of the simplest types of trigger functions. In this case, we consider a threshold value, and if the value of the net input indicates y is greater than the threshold then the neuron is activated.

$$f(x) = 1, \text{if } x >= 0$$
$$f(x) = 0, \text{if } x < 0$$

- Illustration Graphs



Step function illustration

Sigmoid function: is a commonly used trigger function.

$$\frac{1}{(1+e^{-x})}$$

This is a smooth and continuous differential function. Its biggest advantage over stepper and linear functions is that it is not linear. This is an extremely interesting feature of the sigmoid function. Essentially, this means that when I have multiple neurons with sigmoid functions as their activation function – the output isn't linear either. Jaws between 0-1 are S-shaped.

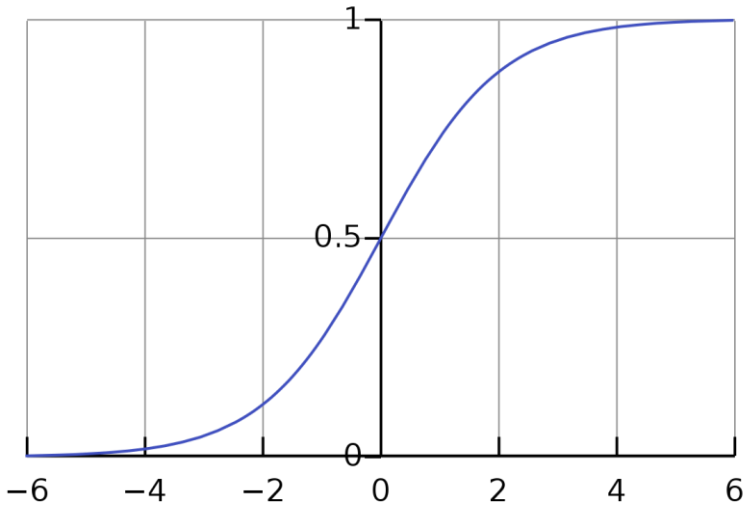
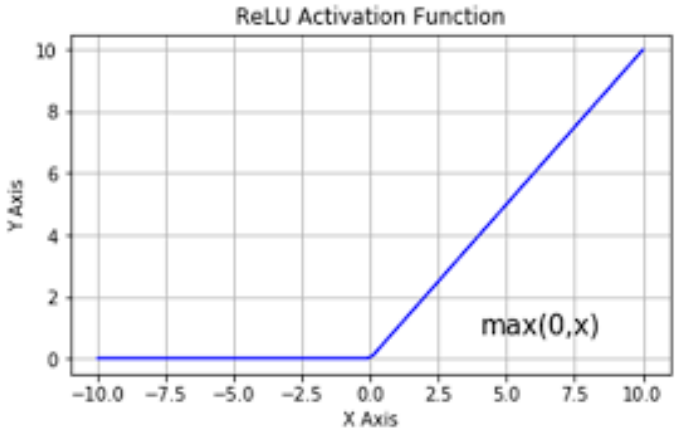


Figure: Illustration of sigmoid function

The ReLU function: is the rectified linear unit. This is the most widely used trigger function. It is defined as:

$$f(x) = \max(0, x)$$

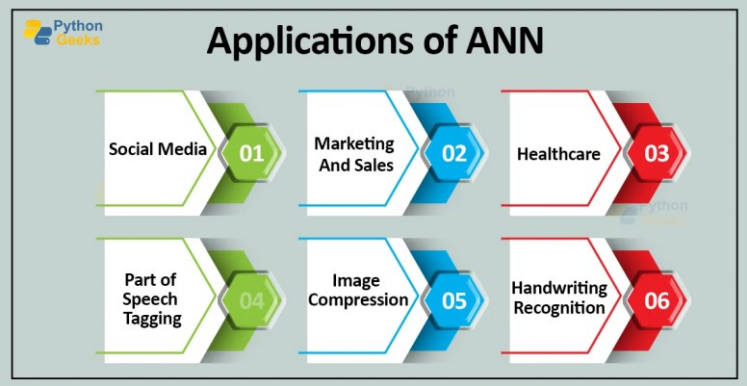
Simulation graph:



3.1.3.2 Applications of Neural Networks

Social Media: Artificial neural networks are heavily used in Social Media. For example, take Facebook's 'People You May Know' feature that suggests people you might know in real life so you can send them friend requests. Well, this magical effect is achieved by using an Artificial Neural Network that analyzes your profile, interests, your current friends as well as their friends, and many other factors to calculate the people that you might know. Another popular application of machine learning on social media is facial recognition. This is done by finding about 100 reference points on the person's face and then matching them to those that are already available in the database using a convolutional neural network.

Marketing and Sales: When you log in to eCommerce sites like Amazon and Flipkart, they will recommend products you should buy based on your past browsing history. Similarly, suppose you love Pasta, then Zomato, Swiggy, etc. will show you restaurant recommendations based on your preferences and previous order history. This is true across all new-age marketing segments such as Book pages, Movie Services, Hospitality pages, etc., and it is done by implementing personalized marketing. This uses Artificial Neural Networks to identify customer preferences, dislikes, previous shopping history, etc., and then tailor marketing campaigns accordingly.



Healthcare: Artificial neural networks are used in the Oncology Department to train algorithms that can identify cancerous tissue at the microscopic level with the same accuracy as trained doctors. Various rare diseases can manifest in physical characteristics and can be identified at an early stage using Facial Analysis on patient photos. Therefore, the comprehensive deployment of Artificial Neural Networks in healthcare settings can only enhance the diagnostic capabilities of medical professionals and ultimately lead to an overall improvement in the quality of medical care worldwide.

Personal Assistant: I'm sure you've all heard of Siri, Alexa, Cortana, etc., and have also heard of them based on the phone you have!! These are personal assistants and examples of speech recognition that use Natural Language Processing to interact with users and give corresponding responses. Natural language processing uses artificial neural networks that are created to handle many of the tasks of these personal assistants such as managing language syntax, semantics, correct speech, ongoing conversations, and more.

3.2 Model Evaluation

MAPE

It shows the average of the absolute percentage errors of each entry in the dataset to calculate how accurate the forecast quantity is compared to the actual number. MAPE is often effective for analyzing large datasets and requires the use of other dataset values of 0.

The formula for calculating MAPE is as follows:

$$MAPE = (1/n) * \Sigma| (Yi - Pi) / Yi | * 100\%$$

In which:

- MAPE: is the Mean Absolute Percentage Error, which represents the average percentage of the predicted error compared to the actual value.
- n: is the number of data points in the dataset being evaluated.

- Yi: is the actual value of the data at the ith data point.
- Pi: is the prediction calculated by the model at the ith data point.

MPE

Mean Percentage Error (MPE) is an index that evaluates the error of a predictive model or forecasting method as a percentage of the actual value. It is used to measure the accuracy of the prediction model in measuring the deviation between the predicted value and the actual value of the data.

In which:

- MAPE: is the Mean Percentage Error, which is the average percentage of error.
- n: is the number of data points in the dataset being evaluated.
- Yi: is the actual value of the data at the ith data point.
- Pi: is the prediction calculated by the model at the ith data point.

RMSE

RMSE (Root Mean Squared Error) is a method of evaluating the accuracy of a predictive model, commonly used in predictive problems, to measure the magnitude of the error between the predicted values and the actual values in a data set. RMSE is the square root of the sum mean of the squares of the errors.

RMSE Calculation Formula:

RMSE = sqrt(1/n * Σ(y_pred - y_true)^2)

In which:

- n is the number of samples in the evaluation dataset
- y_pred is the value predicted by the model
- y_true is the actual value in the evaluation dataset
- Σ is the sum symbol of the values in the sequence of numbers

RMSE is calculated as the unit of measurement of the actual data, for example, the unit of measurement of the predicted variable is meters, the unit of measurement of RMSE will also be meters.

To convert RMSE to RMSE percentage (%RMSE), we can divide RMSE by the average of the actual variable (y_true) and multiply by 100 to get the percentage value:

%RMSE = (RMSE / mean(y_true)) * 100

In which:

- mean(y_true): is the average of the actual variable y_true.
- %RMSE is a relative measure, which shows what percentage the average error of the prediction model is compared to the average value of the actual variable. The smaller the %RMSE, the more accurate the prediction model is, and conversely, the larger the %RMSE, the more inaccurate the prediction model is.

Indicator/ Model name	**Random Forest**	**Decision Tree**	**ANN**
MAPE	54,19%	24,28%	24,52%
MPE	25,01%	24,85%	32,07%
RMSE	0,085	0,085	0,1
%RMSE	33,51%	33,66%	41,48%

Table 3. 1

Table: Comparison of evaluation indicators

4. Conclusion

We have completely built a Machine Learning model using Decision Tree, although the accuracy is relative, but with the project scope of predicting the price of accommodation in Ho Chi Minh City, the algorithm can be used to predict the prices reliably.

There is a limitation that we realize that we have not included the location as a feature due to the limitation of experience as well as the project implementation time, we believe that this is an important factor affecting the price of the hostel.