

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES
DE MONTERREY

DEPARTAMENTO DE CIENCIAS E INGENIERÍA



**Tecnológico
de Monterrey**

Hotel Price Prediction

by

Alexis Vázquez Martínez

Intelligent Systems Final Project

ABSTRACT

In the world of traveling the goal is always to have the best experience, and that includes getting the best possible deal, but how can someone know if a hotel is charging the right amount? how can someone be certain that the price is right? how can someone be certain that the relation price-quality will give the best outcome? those questions are for sure some of the most popular ones between travelers and this model could help to identify those possible prices that hotels can have based on their characteristics

Contents

1	Context of the problem	iv
1.1	Fluctuating Prices	iv
2	Context	1
2.1	Traveling	1
2.2	Hotel hunting	1
2.3	AI in daily life	2
2.4	Linear Regression	3
3	Solution	4
3.1	Model Implementation	4
3.1.1	Imports	5
3.1.2	Data modeling	5
3.2	Manual regression	7
3.3	Results	10
3.3.1	Using sklearn for calculus	10
3.3.2	Manual training	12
4	Conclusions	14
5	Usage	15
5.1	Repo	15
5.2	Instructions	15
	Bibliografía	17

Chapter 1

Context of the problem

1.1 Fluctuating Prices



Figure 1.1: Hotel price fluctuations

When traveling, a main factor in hotel selection is the price, since it fluctuates and changes depending on different factors, but mainly people do their comparisons between other hotels in the area to obtain the price, and that is not the reality. Price depends on the room, the service, the attention, the location, the amenities and other factors that people usually doesn't consider and that represent most of it.

To successfully know how the price of a hotel will behave based on its characteristics, a model is missing to really find correlations that let to a proper result.

Many websites nowadays allow hotel comparison, but not many ones let the user know what to expect based on requested characteristics. With this approach the final user, aka. the traveler, will have an idea of what price to expect for a hotel based in data from other hotels in the region, other cities and with different characteristics.

Chapter 2

Context

2.1 Traveling

Traveling, according to a study conducted by Tourism Review International[2], is an activity that brings an incredible amount of benefits to the traveler, including physical and emotional. The study mainly obtained that traveling was more important to people that travel more, and people than doesn't travel is mainly because of resources like time and money. This also tells us that the lack of visibility in the origin of prices cause that the people prefer to do other thing instead of traveling.



Figure 2.1: Leisure Traveling

2.2 Hotel hunting

Many websites and applications are now very popular to compare prices between hotels, they use many complex algorithms to analyze prices published in different locations, filter data and give the final user the "best" result according to their requests, but this results really just compare known data, they doesn't predict or give relevant information to the user to make decisions.



Figure 2.2: Hotel compare platforms

When airbnb enter the lodging market, many people faced the problem that they didn't really knew how to set their prices, because they tried to compete with the houses around them without paying attention to other details, and the same occurs with hotels (but in a bigger scale).

2.3 AI in daily life

Like the title of the section implies, AI is present (or could be present) in many of the most common activities that we do in a daily basis, and that is because of everything we do, did or will do represents data, data that could be studied and create information that make Artificial Intelligence possible.

AI in price forecasting is incredibly popular since recent years in many subjects, from food price prediction to stock market price prediction. Many frameworks exist to help in the development of Artificial Intelligence models and they had made an amazing improvement in data analysis to make decisions.

Data analysis for predicting properties prices has also been popular since recent years. A model using an AI framework was created by Sayed Huzair[1] to analyze and predict prices in real estate and also will be used as base to this model, with the difference of the approach in the calculus.

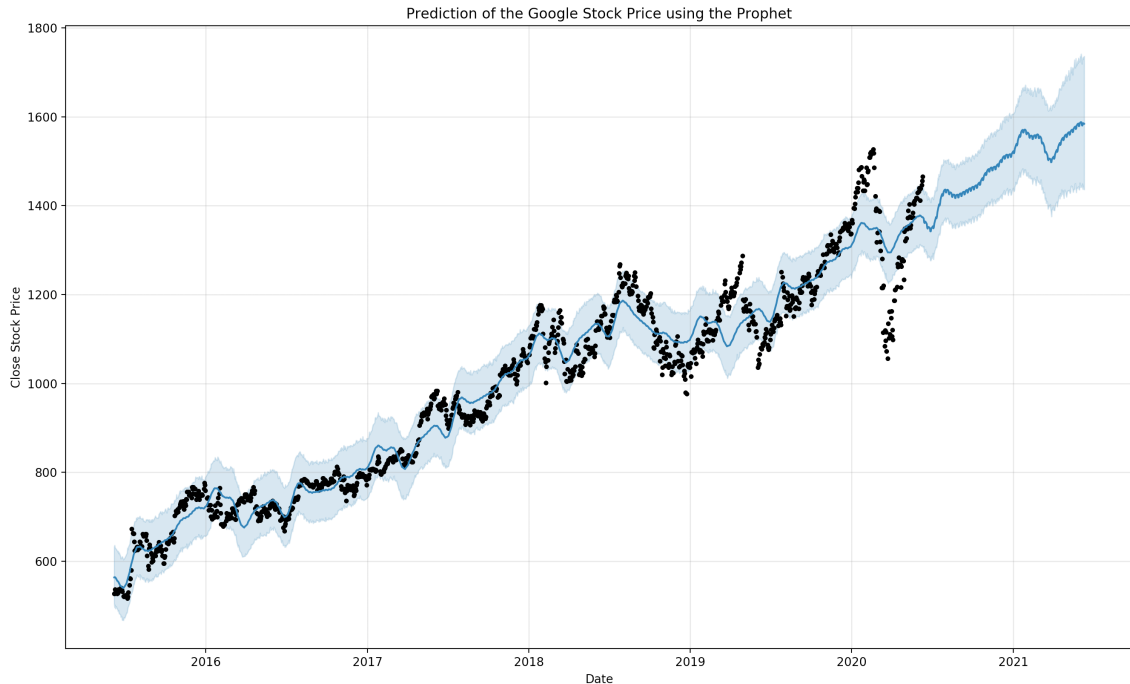


Figure 2.3: AI in stock price prediction

2.4 Linear Regression

LR¹ is an AI technique that is commonly used to predict an outcome based in previous results. Yale define LR as an attempt to model the relationship between two variables by fitting a linear equation to observed data, with an explanatory variable and a dependent variable[3].

To do a linear Regression model, variables should be analyzed to really see if there could be any possible relationship and data can be obtained. Most linear regression models help in prediction and in this case it will help in forecasting.

¹Linear Regression

Chapter 3

Solution

In this project, linear regression will be used to analyze relations between attributes within the data and predict a proper price based in the given input. A dataset will be analyzed, studied, trained and tested to see the results and decide if the model can be used to predict prices.

3.1 Model Implementation

To help in Hotel price prediction, I will use a dataset[4] obtaining information from hotels in Belgium to then be trained and tested. The dataset consists of 120 rows containing data from hotels in bosnia, the information in the dataset is:

- ID
- Name of the hotel
- Price (in Bosnian Currency)
- Hotel Star Rating (from 1 to 5)
- Distance from hotel to midtown
- Customer rating (from 1 to 10)
- Amount of rooms per booking
- Size of room is square meters
- City where the hotel is located

The dataset that will be used is "clean", meaning that data doesn't has to be transformed to fill spaces or to keep in average.

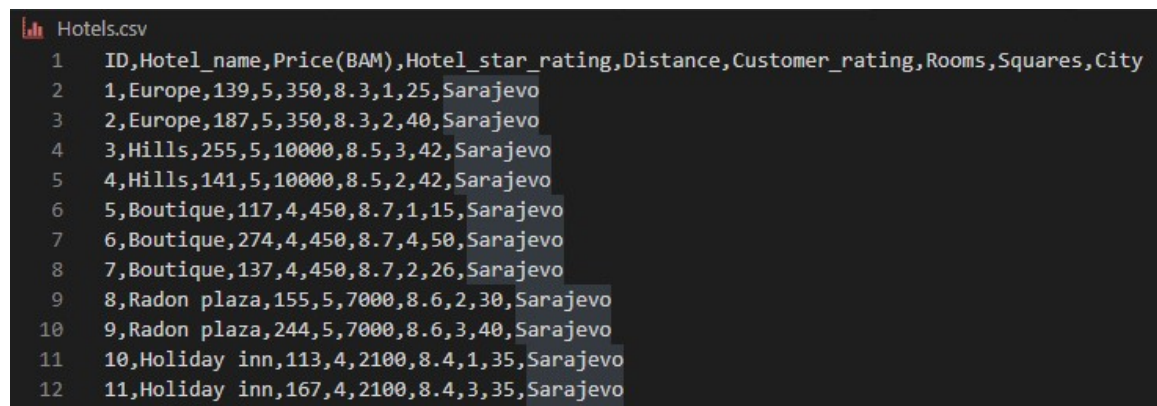
3.1.1 Imports

The model need some libraries to work correctly, pandas for data manipulation, sklearn as an AI framework and seaborn and matplotlib to represent the data. Also when running, some instalations have to ve done as well using pip, like pandas and sklearn.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

3.1.2 Data modeling

Pandas is used to get the information, first the dataset is opened.



	ID	Hotel_name	Price(BAM)	Hotel_star_rating	Distance	Customer_rating	Rooms	Squares	City
1	1	Europe	139,5	350	8.3	1	25		Sarajevo
2	2	Europe	187,5	350	8.3	2	40		Sarajevo
3	3	Hills	255,5	10000	8.5	3	42		Sarajevo
4	4	Hills	141,5	10000	8.5	2	42		Sarajevo
5	5	Boutique	117,4	450	8.7	1	15		Sarajevo
6	6	Boutique	274,4	450	8.7	4	50		Sarajevo
7	7	Boutique	137,4	450	8.7	2	26		Sarajevo
8	8	Radon plaza	155,5	7000	8.6	2	30		Sarajevo
9	9	Radon plaza	244,5	7000	8.6	3	40		Sarajevo
10	10	Holiday inn	113,4	2100	8.4	1	35		Sarajevo
11	11	Holiday inn	167,4	2100	8.4	3	35		Sarajevo

Figure 3.1: Dataset example

When the information is loaded, the explanatory and dependant variables are defined.

```
hotel_dataset = pd.read_csv('Hotels.csv')

X = hotel_dataset[['Hotel_star_rating', 'Distance', 'Customer_rating',
'Squares']]

y = hotel_dataset['Price(BAM)']
```

To see how the data in the dataset behaves, we can use some visuals like sns to try to see some relations between the data and the results, for example here is an

image that helps to see a relation between rating, squares and price. Also a heatmap is a good choice and it lets to understand how strong the relations are.

```
sns.pairplot(hotel_dataset)
sns.heatmap(hotel_dataset.corr(), annot=True)
```

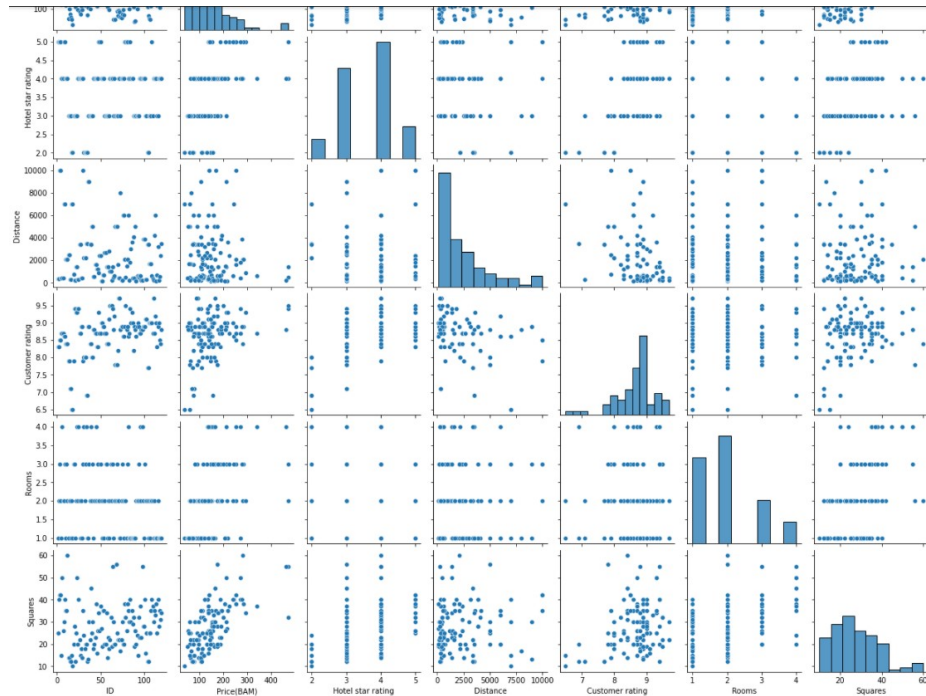


Figure 3.2: Relations between data

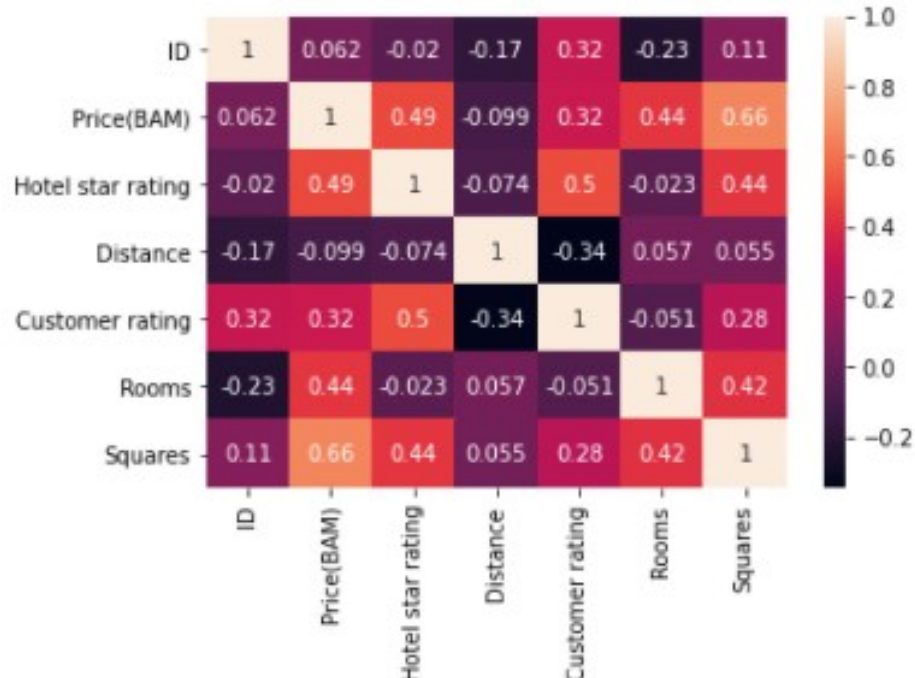


Figure 3.3: Heat map to see how strong the relations are

Sklearn also helps in selecting training sets, which is useful to define the size to learn.

```
X_train, X_test, y_train, y_test = train_test_split (X, y,
test_size=.40, random_state=101)
```

The next steps to follow are training the model and presenting results.

First, to train the model sklearn provides functions like linear regression, that establishes a model from the training.

```
linear = LinearRegression()
linear.fit(X_train, y_train)
predictions = linear.predict(X_test)
```

When this prediction is made and we use some visualization plots, we can see how the information behaves and see that in fact the model has been trained correctly

```
plt.scatter(y_test, predictions)
sns.displot((y_test-predictions),bins=50)
```

3.2 Manual regression

For the manual regression part of the code, the training will be without using a framework, the last steps were as easy as:

```

linear = LinearRegression()
linear.fit(X_train, y_train)
predictions = linear.predict(X_test)

```

but by hand it becomes a little bit trickier.

For a manual training, every step had to be calculated in other way. I used code from our teacher[6] Benjamin Valdés to use some functions:

First, the function `h` evaluates a generic linear function given some parameters, `h` stands for hypothesis.

```

def h(params, sample):
    acum = 0
    for i in range(len(params)):
        acum = acum + params[i]*sample[i]
    return acum

```

The function `show_errors` shows errors, appends the errors or the difference between the estimated hypothesis and the values

```

def show_errors(params, samples,y):
    global __errors__
    error_acum =0
    for i in range(len(samples)):
        hyp = h(params,samples[i])
        error=hyp-y[i]
        error_acum+=error**2
    mean_error_param=error_acum/len(samples)
    __errors__.append(mean_error_param)

```

The function `gradient descent` runs the sample set and generates a new hypothesis, with new parameters including the learning rate

```

def GD(params, samples, y, alfa):
    temp = list(params)
    general_error=0
    for j in range(len(params)):
        acum =0; error_acum=0
        for i in range(len(samples)):
            error = h(params,samples[i]) - y[i]
            acum = acum + error*samples[i][j]
        temp[j] = params[j] - alfa*(1/len(samples))*acum
    return temp

```

The scaling functions normalize the values for converging, normalizing the original values

```
def scaling(samples):
    acum =0
    samples = np.asarray(samples).T.tolist()
    for i in range(1,len(samples)):
        for j in range(len(samples[i])):
            acum+= samples[i][j]
        avg = acum/(len(samples[i]))
        max_val = max(samples[i])
        for j in range(len(samples[i])):
            samples[i][j] = (samples[i][j] - avg)/max_val
    return np.asarray(samples).T.tolist()
```

After this, the model is trained using the functions, and the learning can begin.

```
params = [0,0,0,0]
samples = X_train.values[1:].tolist()
y = y_train[1:].tolist()

alfa =.01 # learning rate
for i in range(len(samples)):
    if isinstance(samples[i], list):
        samples[i]= [1]+samples[i]
    else:
        samples[i]= [1,samples[i]]
##print ("original samples:")
##print (samples)
samples = scaling(samples)
```

Figure 3.4: Defining params and establishing samples with the training x and y values

```
epochs = 0
while True: # run gradient descent until local minima is reached
    oldparams = list(params)
    #print (params)
    params=GD(params, samples,y,alfa)
    show_errors(params, samples, y) #only used to show errors, it is not used in calculation
    #print (params)
    epochs = epochs + 1
    if(oldparams == params or epochs == 2): # local minima is found when there is no further improvement
        #print ("samples:")
        #print(samples)
        print ("final params:")
        print (params)
        break
```

Figure 3.5: Getting the gradient descent to transform the parameters

```

#Delimit test size
tests = X_test.values.tolist()

#Same procedure is done like in training to scale but with the test set
for i in range(len(tests)):
    if isinstance(tests[i], list):
        tests[i] = [1]+tests[i]
    else:
        tests[i] = [1,tests[i]]

tests = scaling(tests)

```

Figure 3.6: Executing the same procedure but using the test set

3.3 Results

3.3.1 Using sklearn for calculus

The training of the model involved a part of the dataset, the testing involved a different set that presented the next results:

The scatter representation of the predictions showed that the results tend to fluctuate in a single area and a pattern is shown.[Fig 3.4]

When the predictions are presented in a distribution plot, we see that the graph tend to normalize. It is because of the amount of data that there are few spaces, but still the normal distribution representation is clear. [Fig 3.5]

The mean squared error, the mean absolute error and the root mean squared deviation presented the next results:

```

MAE: 43.03788800890755
MSE: 2952.3997433743393
RMSE: 54.33598939353492

```

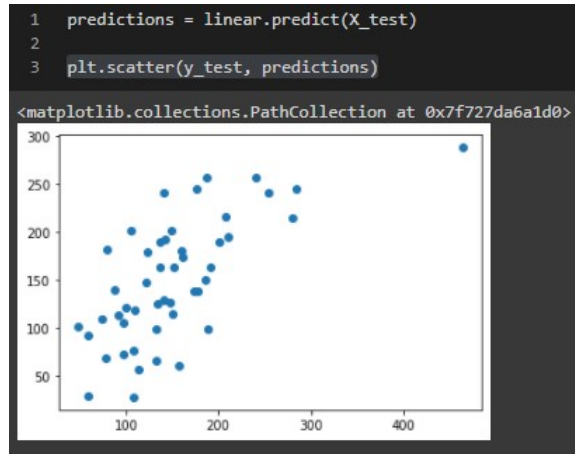


Figure 3.7: Testing using the test set - scatter

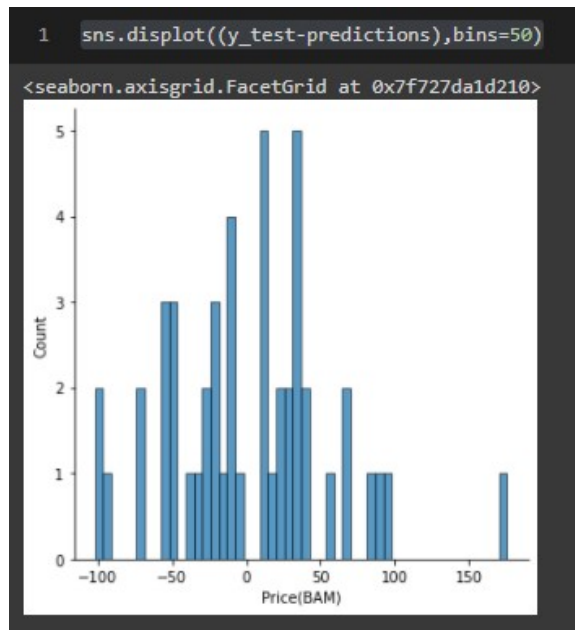


Figure 3.8: Testing using the test set - distribution

After the model was successfully trained, some tests were conducted to get a price from given attributes and the final results showed a proper approximation, and the model can be now really useful.

```
MAE: 43.037888008907565
MSE: 2952.3997433743402
RMSE: 54.33598939353493
Rating of the hotel (From 1 to 5): 5
Distance from the hotel to midtown (in mts): 5
Public rating by users (From 1 to 10): 45
How big is the room? (in square meters) 34

The price of the hotel will be of 294 BAM
or 3646 MXN
```

Figure 3.9: Model implementation test 1

```
MAE: 43.037888008907565
MSE: 2952.3997433743402
RMSE: 54.33598939353493
Rating of the hotel (From 1 to 5): 3
Distance from the hotel to midtown (in mts): 66
Public rating by users (From 1 to 10): 6
How big is the room? (in square meters) 20

The price of the average night in the hotel will be of 101 BAM
or 1252 MXN
```

Figure 3.10: Model implementation test 2

3.3.2 Manual training

More tests were executed but now using the manual approach, the mean squared error, the mean absolute error and the root mean squared deviation presented the next results:

```
MAE: 144.349681563323
MSE: 25793.012854263583
RMSE: 160.60203253465875
Coeff: -4.16941526444752
```

The manual tests also returned results very similar and very meaningful, providing a very close result but with the advantage of defining epochs and an alpha value to set a learning rate.


```
final params:
[3.0152850328374043, 2.2600315625078, 0.6566127878597741, 2.7018802079427564]
MAE: 144.349681563323
MSE: 25793.012854263583
RMSE: 160.60203253465875
Coeff: -4.16941526444752
Rating of the hotel (From 1 to 5): 3
Distance from the hotel to midtown (in mts): 34
Public rating by users (From 1 to 10): 8
How big is the room? (in square meters) 34

The price of the average night in the hotel will be of 183 BAM
or 2269 MXN
```

Figure 3.11: Model manual implementation test 3

Chapter 4

Conclusions

AI has transformed the way we obtain information by providing tools and techniques that help us model our reality. With the use of AI models we can understand aspects that are hard to see in other ways.

In this model, I used supervised learning to train a linear regression model, that helps to predict a possible price, but in reality the uses are many more. I think that hotels or travel agencies could use the information obtained here to provide the final user with tools that can help them to plan their vacations. Other attributes could be used later on, like dates, to obtain a bigger picture of the fluctuation of the prices.

I think that i was a bit limited by te dataset that I obtained, because with more data I am sure that I could have obtained a much better result.

The results showed a precise model, but with a huge area of opportunity, same that could be used later on another investigation. There is still many testing and training to do to obtain the final and better result.

Also, one factor that I didn't considered is the "relevance" of the hotel, there was a case where even when the hotel had the same characteristics than others, the price was incredibly different and all just because of the popularity of the hotel in other parts of the world.

Chapter 5

Usage

5.1 Repo

The repo of this project can be located in <https://github.com/AlexisVM/AI-Booking> and the dataset in <https://www.kaggle.com/c/expedia-personalized-sort/overview/prizes>.

The repo includes:

- Media, particularly a representation of the dataset, the heatmap and the scatter plot.
- 2 datasets, a public studied one, and a personal approach on local hotels.
- 3 python files, one using the bosnian dataset and the sklearn regression framework, one using the bosnian dataset and a manual regression approach and a personal approach using a dataset obtained by me and the manual regression approach.

5.2 Instructions

To run the model clone the repo and follow some steps:

Regression using a framework

- Run the file

```
HotelPriceRegression.py
```

- See the model results
- Wait for an input field and enter requested data
- See the final result using the model

Manual regression

- Run the file

`HotelPriceManualRegression.py`

- See the model results
- Wait for an input field and enter requested data
- See the final result using the model

We use two models to compare a manual approach and one established by a framework.

The characteristics for the data are the following:

- Hotel rating must be from 1 to 5.
- Distance from the hotel to midtown should be in meters.
- The rating from the customers can be obtained from rating agencies and must be from 1 to 10, if it uses another scale, a conversion should be done.
- The size of the room must be in square meters.

The final result will be a hotel's price prediction using the given attributes.

Bibliography

- [1] Huzaif, Sayed, Linear Regression Model For House Prediction, *Github*, Obtained from: <https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price-Prediction>
- [2] Chun-Chu Cheng, Yao-Chin Wang, Perceptions of Travel Importance, Benefits, and Constraints in Predicting Travel Behavior: A Cross-Cultural Comparison of Leisure Travel, *Tourism Review International*, Obtained from: <https://www.ingentaconnect.com/content/cog/tri/2019/00000023/f0020001/art00001;jsessionid=ic-live-01>
- [3] Yale, Linear Regression, *Yale*, Obtained from: <http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm>
- [4] Amar Aladzuz, Hotels accomodation prices dataset, *Kaggle*, Obtained from: <https://www.kaggle.com/aladzuzamar/hotels-accommodation-prices-dataset?select=Hotels.xlsx>
- [5] Huzaif, Sayed, Linear Regression Model For House Prediction, *Studygyaan*, Obtained from: <https://studygyaan.com/data-science-ml/linear-regression-machine-learning-project-for-house-price-prediction>
- [6] Valdes, Benjamin, Linear Reg GD, *Tec*, Obtained from: <https://bit.ly/3cpipCD>