Melbourne Housing Shortage

ITERATION 4

Yan | 722 | 16/10/2019

https://github.com/AKKKKKKKKI/iteration_4_inforsys722

Contents

1. Situation Understanding....3
1.1 Identify the objectives of the situation....3
1.2 Assess the situation....3
1.3 Determine data-mining objectives....5
1.4 Produce a project plan....5
2. Data Understanding....6
2.1 Collect initial data....6
2.2 Describe the data....6
2.3 Explore the data....8
2.4 Verify the data quality....16
3. Data Preparation....20
3.1 Selection the data....20
3.2 Clean the data....20
3.3 Construct the data....23
3.4 Integrate various data sources....25
3.5 Format the data as required....26
4. Data Transformation....27
4.1 Reduce the data....27
4.2 Project the data....28
5. Data-mining methods selection....30
5.1 Match and discuss the objectives of data mining (1.1) to data mining methods...30
5.2 Select the appropriate data-mining method(s) based on discussion....30
6. Data-mining algorithms selection....31
6.1 Conduct exploratory analysis and discuss....31
6.2 Select data-mining algorithms based on discussion....33
6.3 Build/Select appropriate model(s) and choose relevant parameter(s)....34
7. Data Mining....36
7.1 Create and justify test designs....36
7.2 Conduct data-mining – classify, regress, cluster, etc. (models must execute)....37
7.3 Search for patterns....39

8. Interpretation 41
8.1 Study and discuss the mined patterns 41
8.2 Visualize the data, results, models, and patterns 42
8.3 Interpret the results, models, and patterns 44
8.4 Assess and evaluate results, models, and patterns 45
8.5 Iterate prior steps (1 – 7) as required 46
Acknowledge 50
References 52

1. Situation Understanding

Melbourne is the capital, the largest city of Victoria and the second-largest city in Australia. However, due to the rapid growth of the population, housing began to show a shortage. In order to make cities inclusive, secure, resilient and sustainable, housing shortages have become a problem that the Victorian government must address. The housing problem in Melbourne is now the responsibility of the Housing Commission of Victoria. Funding comes from federal and state governments. After the release of Melbourne in 2030, the policy encouraged the development of existing medium-density and high-density areas to gain access to more public transport and other services.

1.1 IDENTIFY THE OBJECTIVES OF THE SITUATION

As of the end of June 2019, Melbourne's population will reach 51.91 million, and based on the Melbourne population over the past eight years, and the growth rate has remained between 1.67% and 6.23%. Many people come to Melbourne to work. Although Melbourne is 3858.1 square miles, one of the largest metropolises in the world, Melbourne has a population density of 453 people per square kilometer. In the long run, housing in Melbourne is in short supply. This result has caused a linear rise in house prices. Many people cannot afford high prices. Melbourne may have many homeless people and slums. Based on the Housing Commission of Victoria and Melbourne's 2030 plan, Melbourne needs to increase housing in medium- and high-density areas. In order to achieve this goal and better address the housing shortage in Melbourne, the project plans to use the data mining technique to help the Housing Commission of Victoria find a more reasonable implementation plan, to look for more potential implementation areas outside the plan based on market characteristics and identifying the influencing factors.

In summary, the objectives that the project needs to achieve are as follows:

- Finding medium- or high-density areas suitable for adding houses.
- Identifying the relationship between house price and housing characteristics, such as carpark, bedrooms, building area and so on.
- Detecting the importance of factors affecting Melbourne's housing prices

The natures of project success are:

- Confirmed the suitable place for building a new housing area where the price is not too high for people.
- The accuracy of the pricing model range in 400,00$
- The three most useful factors can be identified in the data

1.2 ASSESS THE SITUATION

- **Personnel**

The Housing Commission of Victoria has sufficient knowledge of housing construction, but traditional skills make it difficult for them to discover new trends due to the information era. At the same time, among them, people with database analysis and management skills are still a minority. In order to follow the pace of the information era and more timely analysis in the future, the Housing Commission of Victoria should consider hiring more data analysts.

- **Data**

Since the Housing Commission of Victoria is a government agency, they can get a lot of logs and data, such as housing location, price, and a variety of characteristics. As time goes on, more and more data will be generated, because no matter whether the housing market is popular, people will always trade some houses. Moreover, their preference for housing does not change because of changes in the housing market.

- **Risks and Contingencies**

Firstly, it seems this project will not be excess the deadline, because it has the enough time and is not so complex. If it takes longer than anticipated, it will be not a big problem as well. It is a long-term plan but not immediate one. The time of the project can just be extended.

Second, since this project is funded by the government, it is impossible to have the financial risk. If it exists, this project can connect to the United Nations, because when the Australia government cannot afford this project which is cheap, they must have a big problem. It will be the time that the United Nations join in.

Next, I cannot ensure the data are qualitative now, but the attributes and the amount of the data seems to be great. If the quality is not meet, I will go to the website: https://www.domain.com.au, to collect the data by hand.

Finally, if the initial results are less dramatic than expected, I will try to explore another approach to find more results.

- **Requirements**

The security and legal restrictions on the data or project results are not a problem. Firstly, the data has been from the public source and the purpose of this project is to improve one of the United Nations' targets, sustainable development goal. It aims to the public welfare. However, I will concern whether the future team members will use the results for their own interests. For example, maybe they will speculate based on the potential valued house. I cannot make sure whether there will be deployments requirements now. In the following content, if it exists, I will discuss it.

- **Assumptions**

Since this project is highly price related, the results must be influenced by the economic factors. I will assume the economy in Australia will be stable in the future, especially in the currency sector.

Data are safe and qualified, but for the question that do the management team could understand the model, I think it is dependent on which model this project use. It will be discussed later in the model part.

- **Constraints**

There is no any constraint on data, legal. Firstly, the data is from public source. Everyone can get them easily. However, I still cannot ensure whether we need further financial supports, even at this time, it is not necessary.

1.3 DETERMINE DATA-MINING OBJECTIVES

Based on data mining engineers' analysis, the Housing Commission of Victoria can more effectively address housing problems in the region. The preliminary plan is as follows:

- To predict a house price based on the historical information of the house transactions. Then, the commission can determine the structure, features, and location of the new home being built to provide a more reasonable price.
- To observe the houses those people prefer, and to decide what kind of housing to build in which area.
- To derive the main factors affecting home prices has never attempted to lower current house prices.

1.4 PRODUCE A PROJECT PLAN

Table 1

Phase	Time	Resources	Risk
Situation understanding	1 .5 weeks	All analysts	Situation change
Data understanding		All analysts	Data problems, Technology problems
Data preparation	3 weeks	Data mining consultant, some database analyst time	Data problems, Technology problems
Modeling		Data mining consultant, some database analyst time	Technology problems, inability to find an adequate model
Evaluation		All analysts	Situation change, inability to find an adequate model

Deployment	1 week	Data mining consultant, some database analyst time	Situation change, inability to find an adequate model

2. Data Understanding

2.1 COLLECT INITIAL DATA

Melbourne housing shortage needs some essential data. Those data mainly came from Kaggle, and if people are interested in it, they can click this link and press the DOWNLOAD button: https://www.kaggle.com/anthonypino/melbourne-housing-market/downloads/melbourne-housing-market.zip/27

- **Melbourne housing market**: this database is on the Kaggle, and it contains the housing market information from January 2016. There are some missing values. They are recorded as blank.

2.2 DESCRIBE THE DATA

- **Amount of data**

There are many housing transactions every year, so the amount of data is enormous. In the Melbourne housing market dataset, there are 34.9 thousand instances and 21 attributes.

- **Value Type**

Table 2

Attribute	Description	datatype
Suburb	Suburb	Categorical
Address	Address	Nominal
Rooms	Number of rooms	Continuous
Type	Type of property	Nominal
Price:	Price in Australian dollars	Continuous
Method	sales channel	Nominal
Seller G	Real Estate Agent	Nominal
Date	Data	Nominal
Distance	Distance from CBD in kilometers	Continuous
Postcode	Postcode	Nominal
Bedroom2	Number of bedrooms	Continuous
Bathroom:	Number of bathrooms	Continuous

Car	Number of car park	Continuous
Land size	Land size in Square Meters	Continuous
Building Area	Building Size in Square Meters	Continuous
Year Built	Year the house was built	Continuous
Council Area	Governing council for the area	Nominal
Latitudes	Latitudes	Continuous
Longitude	Longitude	Continuous
Region name	General Region	Nominal
Property count	Number of properties that exist in the suburb	Continuous

- **Coding schemes**

In the Method attributes:

- S: Property sold
- SP: Property sold prior
- PI: Property passed in
- PN: Sold prior not disclosed
- SN: Sold not disclosed
- NB: No bid
- VB: Vendor bid
- W: Withdrawn prior to auction
- SA: Sold after auction
- SS: Sold after auction price not disclosed
- N/A: price or highest bid not available.

In the Type attribute:

- br: Bedroom(s)
- h: house, cottage, villa, semi, terrace
- u: unit, duplex
- t: townhouse
- dev site: development site
- o res: other residential.

2.3 EXPLORE THE DATA

Based on this situation, the project makes some hypotheses before using the process to explore the data. For example, the region has an impact on house prices and the type of house, and the structure of the house affects the price of the house. Besides, it is necessary to confirm whether there are houses with suitable price in the medium- and high- density areas.

According to the above statement, the project will then explain the choice of attributes by presenting the results of the data visualization and exploration. Since the process of data visualization is difficult to implement in Pyspark, this project uses pandas, seaborn, etc. to conduct this step.

```
In [15]: from sklearn.feature_selection import f_classif
         from sklearn.feature_selection import SelectKBest
         bestfeatures = SelectKBest(score_func=f_classif , k=10)


         fit = bestfeatures.fit(X,Y)
         dfscores = pd.DataFrame(fit.scores_)
         dfcolumns = pd.DataFrame(X.columns)
         #concat two dataframes for better visualization
         featureScores = pd.concat([dfcolumns,dfscores],axis=1)
         featureScores.columns = ['Specs','Score']  #naming the dataframe columns
         print(featureScores.nlargest(20,'Score'))
```

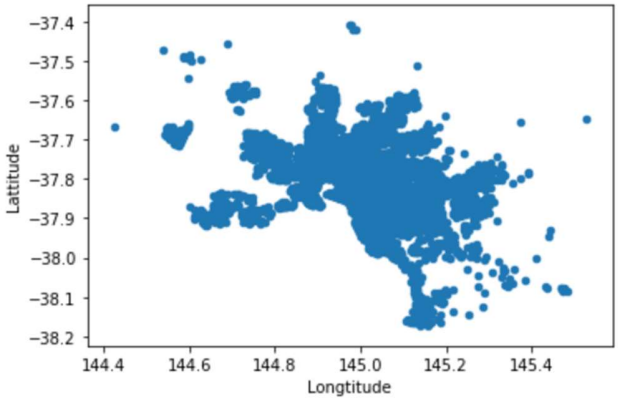
```
            Specs     Score
8    BuildingArea  4.162476
1           Rooms  3.065640
4        Bedroom2  2.872533
5        Bathroom  2.541505
9       YearBuilt  1.857953
10    CouncilArea  1.670604
12     Longtitude  1.437624
0          Suburb  1.411099
3        Distance  1.372392
2            Type  1.365056
11     Lattitude  1.344700
6              Car  1.293605
13     Regionname  1.109224
14  Propertycount  1.044679
7        Landsize  0.587701
```

Figure 1

Before the detailed exploration of the data, the project used feature selection function in sklearn to find the feature importance. As shown in Figure 1, the most important attribute is BuildingArea, Rooms, Bedrooms, Bathroom and YearBuilt.

There are several guesses about this result. First of all, because the larger building area can have a larger house, then it may have the higher price. Secondly, Rooms, Bedrooms and Bathrooms imply the larger living area. People usually are willing to pay for the larger area.

```
In [227]: data.plot(kind="scatter", x="Longtitude", y="Lattitude")
Out[227]: <matplotlib.axes._subplots.AxesSubplot at 0x1d5c944f4a8>
```




```
In [228]: data.plot(kind="scatter", x="Longtitude", y="Lattitude", alpha=0.1)
Out[228]: <matplotlib.axes._subplots.AxesSubplot at 0x1d582224940>
```

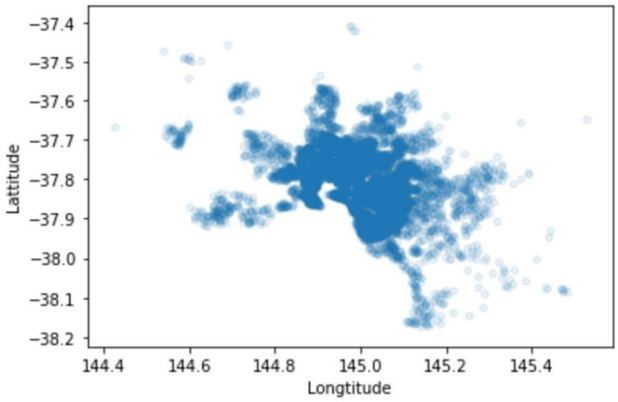


Figure 2

Figure 2 shows the instances can form the map of Melbourne and the most instances are close to city center.

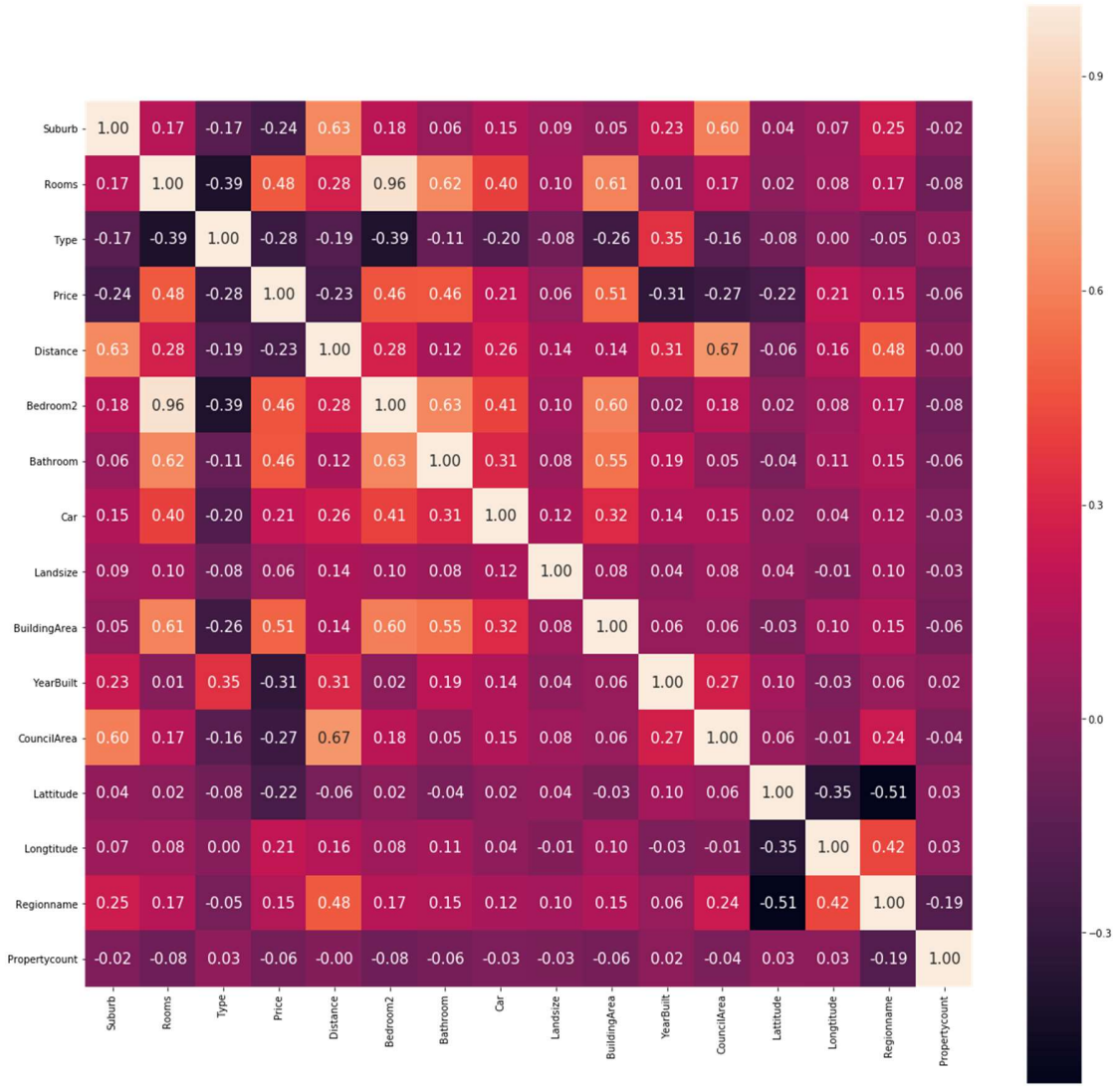


Figure 3

```
In [252]:  corr=data.corr()
           corr = (corr)
           plt.figure(figsize=(20,20))
           sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws=

           {'size': 15},
                      xticklabels=corr.columns.values,
                      yticklabels=corr.columns.values)
```

Figure 4

Next, I use the correlation to find the similar attributes then we can reduce them in the following section. Figure 3 shows the Bedroom2 and the Rooms are quite similar. Then due to the feature importance, the rooms and bedroom2 are the fourth and fifth, I only choose rooms to replace the bedrooms. I think rooms can represent the whole houses better than the bedrooms. The figure 4 shows the code.

Fortunately, the relative impact of the number of houses on prices is not as significant as expected. Therefore, it is basically feasible to build houses in medium and high- density areas. Figure 5 shows, there are still some area available to build house. Next, based on this result, not only analyzing related to density, but the project will also analyze the price and the highest four attributes.

```
In [238]: data.plot(kind="scatter", x="Longtitude", y="Lattitude", alpha=0.4,
              c="Propertycount", cmap=plt.get_cmap("jet"), colorbar=True,
              sharex=False)

Out[238]: <matplotlib.axes._subplots.AxesSubplot at 0x1d581428208>
```

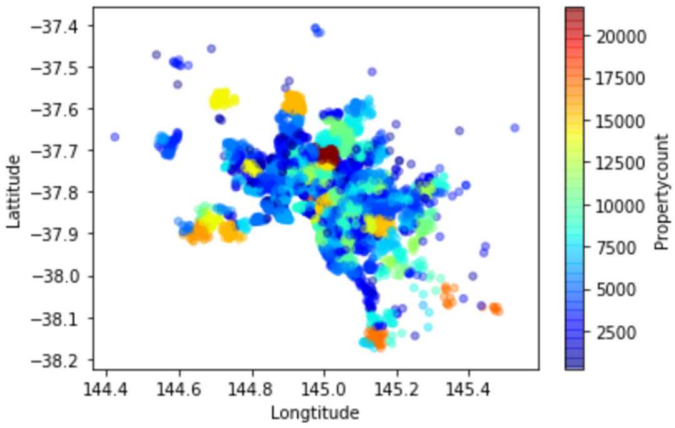


Figure 5

```
In [251]: data.plot(kind="scatter", x="Propertycount", y="Price", alpha=0.1)

Out[251]: <matplotlib.axes._subplots.AxesSubplot at 0x1d5c93d7240>
```

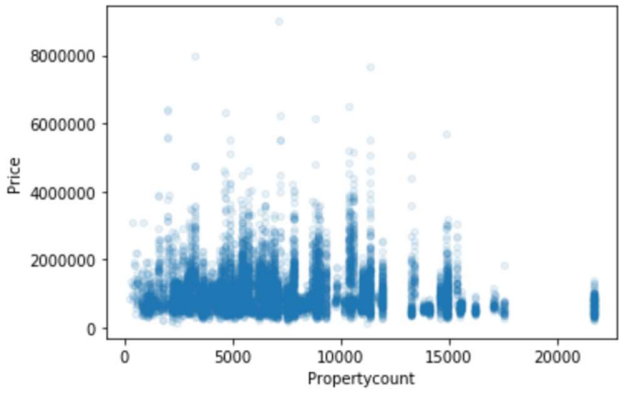


Figure 6

At first, figure 6 illustrates the relationship between price and the number of houses in more detail. When the housing density is moderate, the price is relatively high, while the

low- or too high-density level produces low housing prices. This result still has had a particular impact on the objectives of the project and the establishment of new houses in medium- and high-density areas.

```
In [30]: sns.lmplot(x='BuildingArea',
                    y='Price',
                   data = data)

Out[30]: <seaborn.axisgrid.FacetGrid at 0x235169452b0>
```

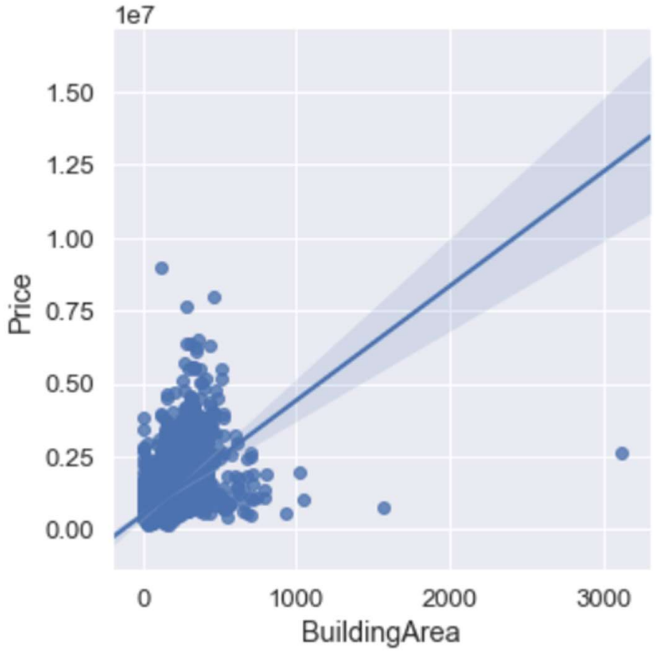


Figure 7

Figure 7 illustrates the relationship between house BuildingArea and price. This project found that these two variables has the positive relationship. As the BuildingArea increases, the Price will go up. Most of them are under 1000 square meters, but it still shows some outliers.

```
In [250]: data.plot(kind="scatter", x="Rooms", y="Price", alpha=0.1)
Out[250]: <matplotlib.axes._subplots.AxesSubplot at 0x1d581331e10>
```

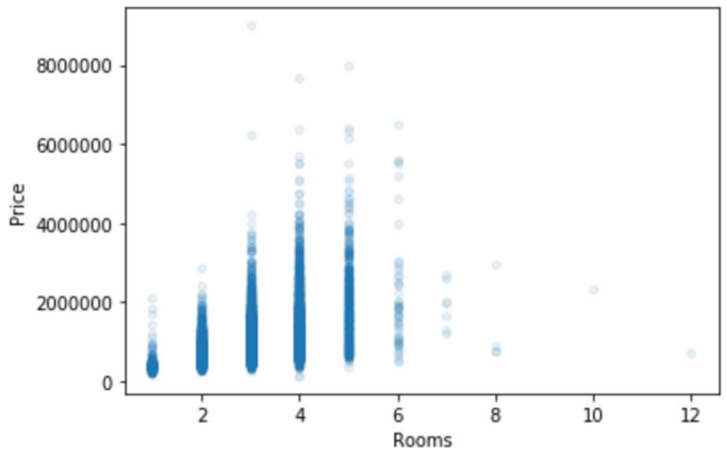


Figure 8

In Figure 8, the relationship between the number of rooms and the price is shown. The house with 4 and 5 rooms has a higher price, so the Housing Commission of Victoria can avoid them when building a house. The 3 rooms housing has the largest price. It might be those houses' characteristics. It cannot remove them directly.

```
In [21]: data.plot(kind="scatter", x="YearBuilt", y="Price",alpha=0.1)
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x2351636a7b8>
```

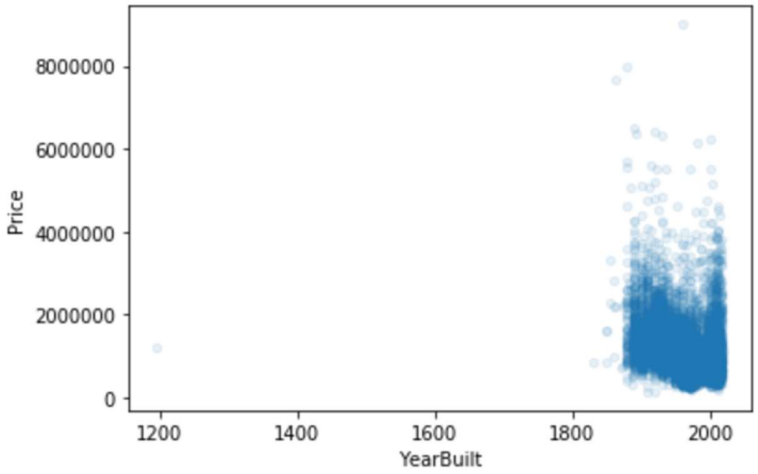


Figure 9

Figure 9 shows the relationship between YearBuilt and Price. In this figure, I cannot conclude some patterns here. There is no clear relationship here, but what I can say is there is almost no housing before 1800, except the obvious outlier in 1200.

```
In [22]: data.plot(kind="scatter", x="Bathroom", y="Price",alpha=0.1)
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x235159a2208>
```

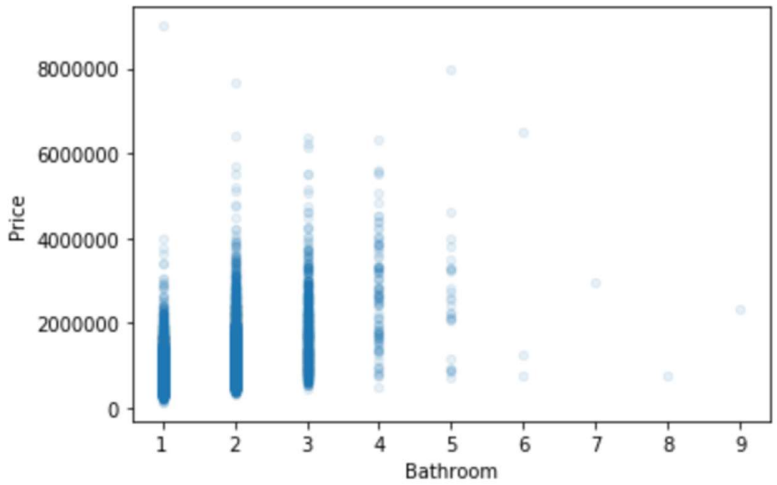


Figure 10

Figure 10 shows the relationship between Bathrooms. It seems a normal distribution. If a house has many bathrooms, it will probably be not high price, but still have some outliers in 6,7, and 9 Bathrooms. The 1 Bathrooms housing has the largest price. It might be those houses' characteristics. It cannot remove them directly. And the house with 3 and 4 Bathrooms has a higher price, so the Housing Commission of Victoria can avoid them when building a house.

```
data = pd.read_csv("Melbourne_housing_FULL.csv")
data.info()
data.dropna(inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
Suburb           34857 non-null object
Address          34857 non-null object
Rooms            34857 non-null int64
Type             34857 non-null object
Price            27247 non-null float64
Method           34857 non-null object
SellerG          34857 non-null object
Date             34857 non-null object
Distance         34856 non-null float64
Postcode         34856 non-null float64
Bedroom2         26640 non-null float64
Bathroom         26631 non-null float64
Car              26129 non-null float64
Landsize         23047 non-null float64
BuildingArea     13742 non-null float64
YearBuilt        15551 non-null float64
CouncilArea      34854 non-null object
Lattitude        26881 non-null float64
Longtitude       26881 non-null float64
Regionname       34854 non-null object
Propertycount    34854 non-null float64
dtypes: float64(12), int64(1), object(8)
memory usage: 5.6+ MB
```

Figure 11

In figure 11, I show the general information of the data. There are totally 348567 instances. 12 attributes are float, 1 attribute is integer and 8 objectives.

```
In [2]: data.describe(include='all').transpose()
```

Out[2]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Suburb	34857	351	Reservoir	844	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Address	34857	34009	5 Charles St	6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rooms	34857	NaN	NaN	NaN	3.03101	0.969933	1	2	3	4	16
Type	34857	3	h	23980	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	27247	NaN	NaN	NaN	1.05017e+06	641467	85000	635000	870000	1.295e+06	1.12e+07
Method	34857	9	S	19744	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SellerG	34857	388	Jellis	3359	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Date	34857	78	28/10/2017	1119	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Distance	34856	NaN	NaN	NaN	11.1849	6.78889	0	6.4	10.3	14	48.1
Postcode	34856	NaN	NaN	NaN	3116.06	109.024	3000	3051	3103	3156	3978
Bedroom2	26640	NaN	NaN	NaN	3.08465	0.98069	0	2	3	4	30
Bathroom	26631	NaN	NaN	NaN	1.6248	0.724212	0	1	2	2	12
Car	26129	NaN	NaN	NaN	1.72885	1.01077	0	1	2	2	26
Landsize	23047	NaN	NaN	NaN	593.599	3398.84	0	224	521	670	433014
BuildingArea	13742	NaN	NaN	NaN	160.256	401.267	0	102	136	188	44515
YearBuilt	15551	NaN	NaN	NaN	1965.29	37.3282	1196	1940	1970	2000	2106
CouncilArea	34854	33	Boroondara City Council	3675	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Lattitude	26881	NaN	NaN	NaN	-37.8106	0.0902789	-38.1904	-37.8629	-37.8076	-37.7541	-37.3902
Longtitude	26881	NaN	NaN	NaN	145.002	0.120169	144.424	144.934	145.008	145.072	145.526
Regionname	34854	8	Southern Metropolitan	11836	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Propertycount	34854	NaN	NaN	NaN	7572.89	4428.09	83	4385	6763	10412	21650

Figure 12

Figure 12 shows some statistical information about the dataset. For those categorical attributes, for example, there are 351 unique suburbs, 33 CouncilArea in Melbourne. For those numeric attributes, on average, there are 3 rooms each housing. The cheapest housing is 85000 dollars.

In summary, the initial hypotheses could be conducted. In addition, it is necessary to consider the number of rooms and the region in the plan to build the new house.

2.4 VERIFY THE DATA QUALITY

The verification of data quality is an important step because in most cases, there are certain problems with the data. For the same reason as the 2.3 section, I will continue implement pandas as the tool. I will use some python code and boxplot to explore, the project found the following problem:

- Missing value:

This data has different degrees of data loss in BuildingArea, YearBuilt, Bedroom, Bathroom, Car, Landsize,Prices, Lattitude and Longtitude by the isnull function in panda package. However, it can be found that the missing bedroom data usually also lack the data of the bathroom, so it can be inferred that the data of this term may not publish specific details, and it is recommended to delete the entire instance. The process is shown in figure 13

```
In [17]: data.isnull().sum()

Out[17]: Suburb              0
         Address             0
         Rooms               0
         Type                0
         Price            7610
         Method              0
         SellerG             0
         Date                0
         Distance            1
         Postcode            1
         Bedroom2         8217
         Bathroom         8226
         Car              8728
         Landsize        11810
         BuildingArea    21115
         YearBuilt       19306
         CouncilArea         0
         Lattitude        7976
         Longtitude       7976
         Regionname          0
         Propertycount       3
         dtype: int64
```

Figure 13

```
In [32]: sns.set_style('darkgrid')

         sns.boxplot(data = data, x = 'Rooms', y = 'Price')

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1be918b9c88>
```

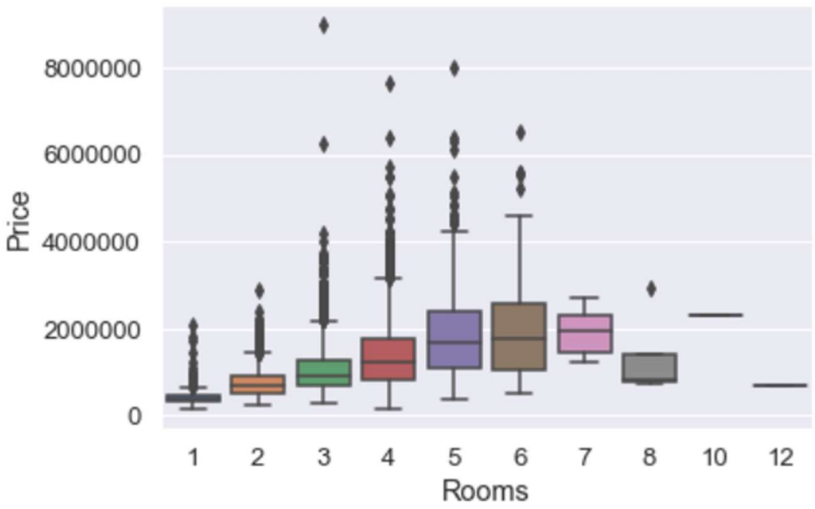


Figure 14

```
In [33]: sns.set_style('darkgrid')

         sns.boxplot(data = data, x = 'Bathroom', y = 'Price')

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1be919e2dd8>
```

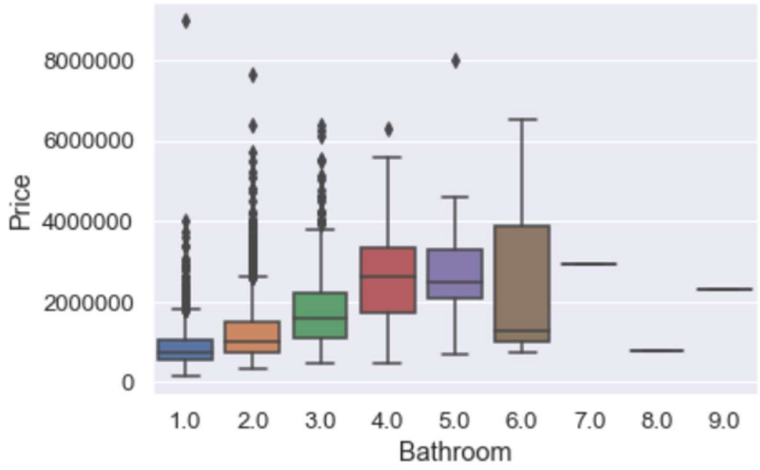


Figure 15

- Outliers and extremes

Outliers and extremes are not so bad. In this subsection, I use the boxplot to find the outliers and extremes, and Rooms and Bathroom are shown as in figure 14 and figure 15. I focus on the four key attributes based on 2.3 section. From figure 14, 15, 16, and 17, the outliers and extremes are not so many, but still some of them. For example, in the Rooms boxplot figure 14, the housing with only one rooms has an extremely high price in room 3, so does room with 5. Also, in Bathroom plot figure 15, there is an obvious outlier in one and five bathrooms.

In [34]:
```
out_hori = data['BuildingArea']
out_horilist = []
out_horilist2 = []
for values1 in out_hori:
    out_horilist += [values1]
    out_horilist2 += [values1]
out_horilist.sort()
new_q1, new_q3= np.percentile(out_horilist,[25,75])

new_iqr = new_q3 - new_q1
lower_bound1 = new_q1 - (1.5 * new_iqr)
upper_bound1 = new_q3 + (1.5 * new_iqr)
print('lower_bound:',lower_bound1)
print('upper_bound:',upper_bound1)
outliers2 = []

for index1 in data.BuildingArea:
    if index1 > upper_bound1 or index1 < lower_bound1:
        outliers2 += [index1]
print(len(outliers2))
```

```
lower_bound: -20.0
upper_bound: 300.0
417
```

Figure 16

In the figure 16, I use the lower bound and upper bound to find the outliers. If the value excesses the bound, then it should be the outlier. The lower bound is negative. That means there is no low value outliers. There are totally 417 outliers in Building Area.

In [35]:
```
out_hori = data['YearBuilt']
out_horilist = []
out_horilist2 = []
for values1 in out_hori:
    out_horilist += [values1]
    out_horilist2 += [values1]
out_horilist.sort()
new_q1, new_q3= np.percentile(out_horilist,[25,75])

new_iqr = new_q3 - new_q1
lower_bound1 = new_q1 - (1.5 * new_iqr)
upper_bound1 = new_q3 + (1.5 * new_iqr)
print('lower_bound:',lower_bound1)
print('upper_bound:',upper_bound1)
outliers2 = []

for index1 in data.YearBuilt:
    if index1 > upper_bound1 or index1 < lower_bound1:
        outliers2 += [index1]
print(len(outliers2))
```

```
lower_bound: 1862.5
upper_bound: 2082.5
10
```

Figure 17

In the figure 17, I use the same method to find the outliers. The lower bound is 1862 and the upper bound is irregular, which is 2082. That means there is no higher value outliers. There are totally 10 outliers in YearBuilt.

3. Data Preparation

3.1 SELECTION THE DATA

- Selecting items

The project focuses on Melbourne housing issues and therefore requires a large amount of Melbourne property data, such as housing price. The data used in this project was derived from Kaggle. Because Kaggle is a database with a lot of public data, it can also give morality while guaranteeing the amount of data. There are 34.9 thousand sets of information on real estate transactions in Melbourne. At the same time, since this data set has been cleaned by the uploaders to a certain extent, they have deleted the extra data, so there is no need for too much data filtering.

- Selecting attributes

Since Kaggle's dataset already provides the features required for this project, such as rooms' amount, area and property density, no additional selection is required.

3.2 CLEAN THE DATA

Since the data preprocessing step is too complicated and time consuming to be inefficient in Pyspark for the project, the project decided to continue using Pandas as a tool for data preprocessing.

According to 2.4 section, the main problem with this data is the missing data, mainly consisting of BuildingArea, YearBuilt, Bedroom, Bathroom, Car, Landsize, Prices, Lattitude and Longtitude. The way this project uses for these missing data is removing the unimportant attributes by using the dropna function. However, before I use this function, I remove several attributes first, Address, Method, SellerG, Postcode and Date (Dropping Price here is to make the X, and move Price as the target value, which is Y). Address, Postcode and SellG is similar as the Record ID. Method is too many which makes the model to complex. I do not care date here because the year period is relatively small. A more detailed explanation will be shown in the 4.1 section. This process is shown in figure 18.

```
In [14]: Y = data["Price"]
         data = data.drop("Address",1)
         data = data.drop("Method",1)
         data = data.drop("SellerG",1)
         data = data.drop("Postcode",1)
         data = data.drop("Date",1)
         X = data.drop("Price",1)

In [15]: X.shape

Out[15]: (8887, 15)
```

Figure 18

Then I apply the dropna to drop the missing values, as shown in figure 19.

```
data.dropna(inplace=True)
```

```
In [28]: data.isnull().sum()

Out[28]: Suburb          0
         Address         0
         Rooms           0
         Type            0
         Price           0
         Method          0
         SellerG         0
         Date            0
         Distance        0
         Postcode        0
         Bedroom2        0
         Bathroom        0
         Car             0
         Landsize        0
         BuildingArea    0
         YearBuilt       0
         CouncilArea     0
         Lattitude       0
         Longtitude      0
         Regionname      0
         Propertycount   0
         dtype: int64
```

Figure 19

Since the information that can help fill in the missing value is not found, filling the missing value arbitrarily will result in inaccurate results, the project decides to delete those instances with missing values by the selection function. Finally, this dataset still contains 8887 instances as shown in figure 20.

```
In [30]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8887 entries, 2 to 34856
Data columns (total 21 columns):
Suburb           8887 non-null int64
Address          8887 non-null object
Rooms            8887 non-null int64
Type             8887 non-null int64
Price            8887 non-null float64
Method           8887 non-null object
SellerG          8887 non-null object
Date             8887 non-null object
Distance         8887 non-null float64
Postcode         8887 non-null category
Bedroom2         8887 non-null float64
Bathroom         8887 non-null float64
Car              8887 non-null float64
Landsize         8887 non-null float64
BuildingArea     8887 non-null float64
YearBuilt        8887 non-null float64
CouncilArea      8887 non-null int64
Lattitude        8887 non-null float64
Longtitude       8887 non-null float64
Regionname       8887 non-null int64
Propertycount    8887 non-null float64
dtypes: category(1), float64(11), int64(5), object(4)
memory usage: 1.8+ MB
```

Figure 20

Secondly, for the outliers and extreme values, I think that should not be dealt with because these abnormal values are many enough to be a group. If they are deleted, it may destroy the complete of the data. Even the final model will have low accuracy.

3.3 CONSTRUCT THE DATA

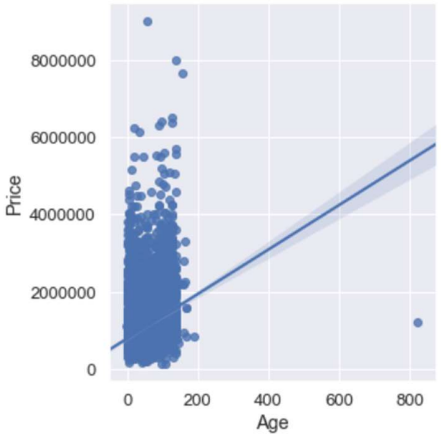
```
In [37]: data['Age']= 2019 - data.YearBuilt
         X['Age']= data.Age

In [38]: print(data.Age)
```

```
2      119.0
4      119.0
6        5.0
11     109.0
14     129.0
18     119.0
24      14.0
25      10.0
30     129.0
32     139.0
35     129.0
37      34.0
38     119.0
42      49.0
43       6.0
44     119.0
49       7.0
51      99.0
56      69.0
57       6.0
58     119.0
59      21.0
61     119.0
63      89.0
66       3.0
67      54.0
68      54.0
70      69.0
71      10.0
72      39.0
```

Figure 21

As of the 2.3 part, I cannot find the obvious relationship between YearBuilt and Price. However, the importance of YearBuilt is relatively high. Then I think it is better to get a more explainable way of showing YearBuilt. And the year may be a relatively complex variable. Therefore, I decided to create a new variable: Age. Since this year is 2019 and the latest housing is built in 2019, I decided to use 2019 as the subtraction to subtract YearBuilt to get the age. The process is as shown in Figure 21.

```
In [51]: sns.lmplot(x='Age', y='Price', data = data)
Out[51]: <seaborn.axisgrid.FacetGrid at 0x232df5d5a20>
```




```
In [49]: sns.lmplot(x='Age', y='Price',data=data)
Out[49]: <seaborn.axisgrid.FacetGrid at 0x1f726c46518>
```

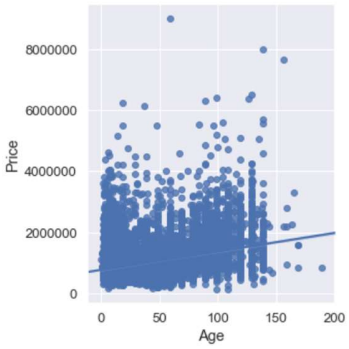


Figure 22

However, as shown in figure 22 left-hand side, there is an obvious outlier. The outlier which the age is 823 years old is quite irregular as shown in figure 23. I think a house is aged 823 in Australia is impossible because Australia only has been 231 years since colony (Australia, n.d.). Then I decide to remove it. The process is shown in figure 23. The outlier in age is removed.

```
In [48]: dfb = data[data['Age']==823].index.values.astype(int)[0]
         data = data.drop(dfb)

In [53]: data.describe(include='all').transpose()
```

Out[53]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Suburb	8886	NaN	NaN	NaN	92.7359	68.2317	0	40	80	127	314
Address	8886	8763	36 Aberfeldie St	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rooms	8886	NaN	NaN	NaN	2.06291	0.752249	1	1	2	3	3
Type	8886	NaN	NaN	NaN	0.335809	0.620957	0	0	0	1	2
Price	8886	NaN	NaN	NaN	1.09289e+06	679419	131000	641000	900000	1.345e+06	9e+06
Method	8886	5	S	5602	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SellerG	8886	250	Nelson	986	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Date	8886	77	24/02/2018	227	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Distance	8886	NaN	NaN	NaN	11.1995	6.81371	0	6.4	10.2	13.9	47.4
Postcode	8886	194	3073	194	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Bedroom2	8886	NaN	NaN	NaN	3.07821	0.966323	0	2	3	4	12
Bathroom	8886	NaN	NaN	NaN	1.64652	0.721619	1	1	2	2	9
Car	8886	NaN	NaN	NaN	1.69199	0.975212	0	1	2	2	10
Landsize	8886	NaN	NaN	NaN	523.448	1061.38	0	212	478	652	42800
BuildingArea	8886	NaN	NaN	NaN	149.313	87.9299	0	100	132	180	3112
YearBuilt	8886	NaN	NaN	NaN	1965.84	36.1315	1830	1945	1970	2000	2019
CouncilArea	8886	NaN	NaN	NaN	10.2741	7.00472	0	4	10	15	32
Lattitude	8886	NaN	NaN	NaN	-37.8045	0.0905511	-38.1744	-37.8585	-37.7987	-37.7489	-37.4072
Longtitude	8886	NaN	NaN	NaN	144.991	0.118918	144.424	144.92	144.998	145.065	145.526
Regionname	8886	NaN	NaN	NaN	1.44238	1.28442	0	0	1	2	7
Propertycount	8886	NaN	NaN	NaN	7475.28	4374.82	249	4381.25	6567	10331	21650
Age	8886	NaN	NaN	NaN	53.16	36.1315	0	19	49	74	189

Figure 23

As shown in figure 22 right-hand side, after the outlier removing, the result shows better than the previous one. And it shows a flatter positive relationship between age and price.

3.4 INTEGRATE VARIOUS DATA SOURCES

In [54]:
```
data1 = pd.read_csv("Housing Pricing 1.csv")
data2 = pd.read_csv("Housing Pricing 2.csv")
dataset = data1.append(data2)

export_csv = dataset.to_csv (r'Newdata.csv', index = None, header=True)
final_set = pd.read_csv("Newdata.csv")
```

In [55]:
```
final_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
Suburb           34857 non-null object
Address          34857 non-null object
Rooms            34857 non-null int64
Type             34857 non-null object
Price            27247 non-null float64
Method           34857 non-null object
SellerG          34857 non-null object
Date             34857 non-null object
Distance         34856 non-null float64
Postcode         34856 non-null float64
Bedroom2         26640 non-null float64
Bathroom         26631 non-null float64
Car              26129 non-null float64
Landsize         23047 non-null float64
BuildingArea     13742 non-null float64
YearBuilt        15551 non-null float64
CouncilArea      34854 non-null object
Lattitude        26881 non-null float64
Longtitude       26881 non-null float64
Regionname       34854 non-null object
Propertycount    34854 non-null float64
dtypes: float64(12), int64(1), object(8)
memory usage: 5.6+ MB
```

Figure 24

Since the dataset downloading from Kaggle are separated into two csv files. To have as many as data possibly. It is necessary to merge these two datasets into one csv files. Therefore, I upload these two files on Jupyter. As the process shown in figure 24, I use append function to merge them. I named the merged dataset as final_set. The information on final_set is shown by info () function. After this process, I can get the complete data.

3.5 FORMAT THE DATA AS REQUIRED

In [4]:
```
Type_collection = []
for Type_element in data.Type:
    if Type_element not in Type_collection:
        Type_collection += [Type_element]
print(Type_collection)
transfer_list = []
for transfer in data.Type:
    num = Type_collection.index(transfer)
    transfer_list += [num]
data.Type = transfer_list
print(data.Type)
```

```
['h', 'u', 't']
2       0
4       0
6       0
11      0
14      0
18      0
```

In [5]:
```
Suburb_collection = []
for Suburb_element in data.Suburb:
    if Suburb_element not in Suburb_collection:
        Suburb_collection += [Suburb_element]
print(Suburb_collection)
transfer_list = []
for transfer in data.Suburb:
    num = Suburb_collection.index(transfer)
    transfer_list += [num]
data.Suburb = transfer_list
print(data.Suburb)
```

```
['Abbotsford', 'Airport West', 'Albert Park', 'Alphington', 'Altona', 'Altona North', 'Armadale', 'Ascot Vale', 'Ashburton',
'Ashwood', 'Avondale Heights', 'Balaclava', 'Balwyn', 'Balwyn North', 'Bentleigh', 'Bentleigh East', 'Box Hill', 'Braybroo
k', 'Brighton', 'Brighton East', 'Brunswick', 'Brunswick West', 'Bulleen', 'Burwood', 'Camberwell', 'Canterbury', 'Carlton N
orth', 'Carnegie', 'Caulfield', 'Caulfield North', 'Caulfield South', 'Chadstone', 'Clifton Hill', 'Coburg', 'Coburg North',
'Collingwood', 'Doncaster', 'Eaglemont', 'Elsternwick', 'Elwood', 'Essendon', 'Essendon North', 'Fairfield', 'Fitzroy', 'Fit
zroy North', 'Flemington', 'Footscray', 'Glen Iris', 'Glenroy', 'Gowanbrae', 'Hadfield', 'Hampton', 'Hampton East', 'Hawthor
n', 'Heidelberg Heights', 'Heidelberg West', 'Hughesdale', 'Ivanhoe', 'Kealba', 'Keilor East', 'Kensington', 'Kew', 'Kew Eas
```

In [6]:
```
CouncilArea_collection = []
for CouncilArea_element in data.CouncilArea:
    if CouncilArea_element not in CouncilArea_collection:
        CouncilArea_collection += [CouncilArea_element]
print(CouncilArea_collection)
transfer_list = []
for transfer in data.CouncilArea:
    num = CouncilArea_collection.index(transfer)
    transfer_list += [num]
data.CouncilArea = transfer_list
print(data.CouncilArea)
```

```
['Yarra City Council', 'Moonee Valley City Council', 'Port Phillip City Council', 'Darebin City Council', 'Hobsons Bay City
Council', 'Stonnington City Council', 'Boroondara City Council', 'Monash City Council', 'Glen Eira City Council', 'Whitehors
e City Council', 'Maribyrnong City Council', 'Bayside City Council', 'Moreland City Council', 'Manningham City Council', 'Me
lbourne City Council', 'Banyule City Council', 'Brimbank City Council', 'Kingston City Council', 'Hume City Council', 'Knox
City Council', 'Melton City Council', 'Maroondah City Council', 'Greater Dandenong City Council', 'Nillumbik Shire Council',
'Whittlesea City Council', 'Frankston City Council', 'Macedon Ranges Shire Council', 'Yarra Ranges Shire Council', 'Wyndham
City Council', 'Casey City Council', 'Cardinia Shire Council', 'Mitchell Shire Council', 'Moorabool Shire Council']
2         0
4         0
6         0
11        0
```

In [7]:
```
Regionname_collection = []
for Regionname_element in data.Regionname:
    if Regionname_element not in Regionname_collection:
        Regionname_collection += [Regionname_element]
print(Regionname_collection)
transfer_list = []
for transfer in data.Regionname:
    num = Regionname_collection.index(transfer)
    transfer_list += [num]
data.Regionname = transfer_list
print(data.Regionname)
```

```
['Northern Metropolitan', 'Western Metropolitan', 'Southern Metropolitan', 'Eastern Metropolitan', 'South-Eastern Metropolit
an', 'Northern Victoria', 'Eastern Victoria', 'Western Victoria']
2       0
4       0
6       0
11      0
14      0
18      0
24      0
25      0
```

Figure 25

Because to make a pricing prediction model, the categorical cannot be applied, thus I reformat them into numeric representatives. As shown in figure 25, Type is reformatted as h for 0, t for 1 and u for 2. The Suburb is applied by the same process, from Abbotsford to Research, I give them value from 0 to 314. The CounilArea gives "Yarra City Council" 0 value, and "Moorabool Shire Council" is 32. For the Regionname is from 0 to 7.

After this process, the regression models can be applied.

4. Data Transformation

4.1 REDUCE THE DATA

Since the data preprocessing step is too complicated and time consuming to be inefficient in Pyspark, the project decided to continue using Pandas as a data preprocessing tool.

Firstly, based on logical thinking, I decided to drop 5 features as shown in figure 15 and mentioned in 3.2 section. They are Address, Method, SellerG, Date, and Postcode (Dropping Price here is to make the X and move Price as the target value, which is Y). Consider the possible reasons as follows. First, the price of a house in one block is roughly the same and may vary due to the characteristics of the house, but this information is already present in other attributes. The address and Postcode here are more like a Record ID. Secondly, the way of selling house seems not to affects the price of housing. It might be that buyers do not care the methods but focus more on the house itself. After that, there are many housing transaction agencies. Even though some agencies may be better at selling houses, it is impossible to consider them one by one. The date may be since the data is only from 2016 to 2019, during which time the house price did not fluctuate considerably.

Then this project considers removing the bedrooms2 attributes because as mentioned in 2.3, the bedrooms2 shows similar results, then to dimensionality reduction, it could be removed. The process and results will be shown in figure 26.

As shown in Figure 1 in the 2.3 section, the Landsize is relatively unimportant compared to other factors. I think the potential reason is that there are many units in the dataset which do not have lands, but sometimes they give a higher price than others. Then Landsize is another feature what I need to drop. The process and results will be shown in figure 26 as well.

```
In [40]: data = data.drop("Bedroom2",1)
         X = data.drop("Landsize" and "Price",1)

In [41]: X.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 8887 entries, 2 to 34856
Data columns (total 15 columns):
Suburb          8887 non-null int64
Rooms           8887 non-null int64
Type            8887 non-null int64
Distance        8887 non-null float64
Bathroom        8887 non-null float64
Car             8887 non-null float64
Landsize        8887 non-null float64
BuildingArea    8887 non-null float64
YearBuilt       8887 non-null float64
CouncilArea     8887 non-null int64
Lattitude       8887 non-null float64
Longtitude      8887 non-null float64
Regionname      8887 non-null int64
Propertycount   8887 non-null float64
Age             8887 non-null float64
dtypes: float64(10), int64(5)
memory usage: 1.4 MB
```

Figure 26

After that, since the target of this project is Price and is marked as continuous data, this project cannot process the step of balancing the target.

4.2 PROJECT THE DATA

```
In [43]: data["Rooms"].hist()

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x279503e3ac8>
```

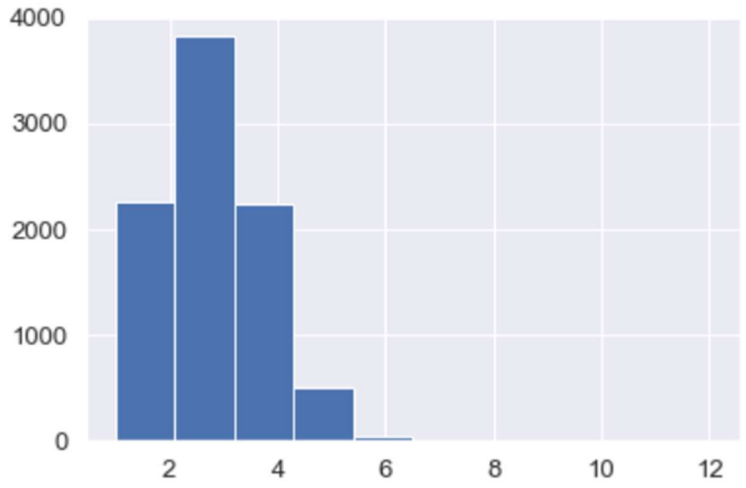


Figure 27

First, due to figure 27, I can find the values in the Rooms are not balanced, there are many properties has 3 rooms, but not so much has over 4 rooms. Therefore, I reclassify the numbers of rooms.

```
In [45]:
Newrooms = []
for group1 in data['Rooms']:
    if group1 < 3:
        Newrooms += [1]
    elif group1 == 3 :

        Newrooms += [2]

    else:
        Newrooms += [3]
data['Rooms'] =  Newrooms

data.head()
```

Out[45]:

	Suburb	Address	Rooms	Type
2	0	25 Bloomburg St	1	0
4	0	5 Charles St	1	0
6	0	55a Park St	2	0
11	0	124 Yarra St	1	0
14	0	98 Charles St	1	0

5 rows × 22 columns

Figure 28

As shown in Figure 28, I put the properties which has one or two rooms into the property with less rooms which I code it as 1. Those three rooms properties are classified into the same categories, which coding as 2. Then the rest properties have many rooms, and I put them together. Their code is 3.

```
In [46]:
data["Rooms"].hist()
```

```
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x232dd9e7048>
```

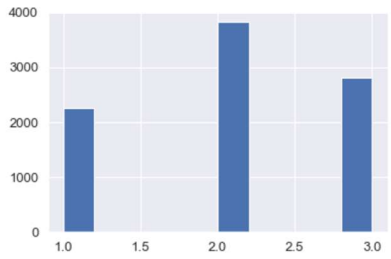


Figure 29

As shown in figure 29, the Rooms now is more balanced than pervious now. Although the housing with three rooms is still the most one, other two categories are relatively comparable. They are not lower the higher one so much.

5. Data-mining methods selection

5.1 MATCH AND DISCUSS THE OBJECTIVES OF DATA MINING (1.1) TO DATA MINING METHODS

The objectives of the project are mainly three.

The first one is to build a housing pricing model by providing attributes, such as rooms and areas. The model will be able to predict the price of the home with a rational accuracy.

Then the project should explore the important attributes that affect housing prices to build new homes with reasonable housing prices. The government could apply the importance to deciding build what kinds of price.

Finally, the project should reflect how those attributes affect the housing price. Hopefully, this project can get the detailed estimated relationship: when increases one unit of certain attribute, the price will increase or decrease some volume.

Thus, the ultimate goal should be to focus on price, which is a continuous number. In general, the data mining process of this project should use various attributes to predict the target, Price.

5.2 SELECT THE APPROPRIATE DATA-MINING METHOD(S) BASED ON DISCUSSION

Basically, there are three methods of data mining. They are Classification, Regression and Clustering.

Firstly, the Classification is a method of predicting numbers or categories, and the data can be classified into different category (Kashid, 2018). This approach can help the user predict the results. As mentioned above, the objectives of this project are by using housing characteristics to predict the house price. Then I can clearly define the input is housing characteristics, and Prices is the output, which is fit in the classification method.

Then Regression are usually used to predict numbers of data. It is a traditional statistical model. In this method, the category will be instead of numeric value. By using this method, the distribution trends can be identified by the dataset (Kashid, 2018). It should be a powerful tool to predict the housing price by houses' age, distance and BuildingArea.

Finally, Clustering partition the data into groups. These groups will be called clusters. It is a useful tool in splitting up the data (Kashid, 2018). This method can distinguish the similar clusters and different clusters. There is no need for label values.

This project tends to use classifications and regressions because it is a way to use the value of an input to predict the value of output or target. This project has the target which is Price, and many attributes, such as Rooms. Ideally, those attributes can be used to predict the housing price. However, I cannot make sure which one is a better one based on my data understanding. It is better to do both models then to decide which one has the higher accuracy. Therefore, Clustering should be excluded, Classification and Regression will be explored first.

6. Data-mining algorithms selection

6.1 CONDUCT EXPLORATORY ANALYSIS AND DISCUSS

To begin this project's models, since the data preprocessing step is too complicated and time consuming to be inefficient in Pyspark, the project decided to continue using Pandas as a data preprocessing tool. Then the second thing is to check the data quality. I use the infor function to do this process. Then I can confirm the data quality is qualified. As shown in figure 30, the results are satisfying. All of attributes are without missing value. Totally, there are 8886 records in 16 attributes It is the time to do the following process.

```
In [49]: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 8886 entries, 2 to 34856
Data columns (total 16 columns):
Suburb          8886 non-null int64
Rooms           8886 non-null int64
Type            8886 non-null int64
Price           8886 non-null float64
Distance        8886 non-null float64
Bathroom        8886 non-null float64
Car             8886 non-null float64
Landsize        8886 non-null float64
BuildingArea    8886 non-null float64
YearBuilt       8886 non-null float64
CouncilArea     8886 non-null int64
Lattitude       8886 non-null float64
Longtitude      8886 non-null float64
Regionname      8886 non-null int64
Propertycount   8886 non-null float64
Age             8886 non-null float64
dtypes: float64(11), int64(5)
memory usage: 1.2 MB
```

Figure 30

Then I will explore the attributes by visualization and summarize their characteristics by different tables and figures in the former processes.

Finally, based on the previous data analysis and data mining objectives, the project needs to compare house features to predict house prices. Because the prediction results of this project are numerical and most of the attributes are continuous, such as the number of rooms, it is more suitable for regression models combined with ordinary classifications. There are many those kinds of models available in Pyspark, such as Decision Tree, Random Forest, and Linear Regression. All of them has their own advantages, disadvantages and suitable situations.

First, Decision tree is tree-based prediction methods. Its targets and input can be continuous or categorized. In Decision tree, it can cover the information classified by Region and so on. The Decision Tree first checks the input to find the best segmentation. It is then measured by the reduction in the index of impurities produced by the segmentation. It has the advantage of being able to effectively handle problems such as data loss and large numbers of fields. Like Decision tree, training time is short and can be easily understood.

Then the Random Forest is an advanced Algorithm with a bagging method based on a tree model. In a random forest, each tree in the whole is constructed from samples drawn from alternatives in the training set. The selected split point is the best split for all random factors in the subset. Due to randomness, forest deviations usually increase slightly, and variances decrease. The reduction in variance can compensate for the increase in bias, so the resulting model will be better.

Finally, in linear regression, relationships are modeled using linear prediction functions whose unknown model parameters can be estimated from the data (Linear Rregression, n.d.). Its advantage is an extremely simple method. It is very easy and intuitive to use and understand. As the same as the tree-based models, a person with only the knowledge of high school mathematics can understand and use it. In addition, it works in most of the cases. Even when it doesn't fit the data exactly, we can use it to find the nature of the relationship between the two variables. One of its disadvantages is that they only model relationships between dependent and independent variables that are linear. It assumes there is a straight-line relationship between them which is incorrect sometimes. Another disadvantage is that if there are more parameters than the number of samples available then the model starts to model the noise rather than the relationship between the variables.

Overall, this project will choose the best models from these three suitable methods.

6.2 SELECT DATA-MINING ALGORITHMS BASED ON DISCUSSION

As mentioned above, this project will use Decision Tree, Random Forest, and Linear regression. The following will explain why I choose those methods.

Firstly, the Decision tree is very suitable for the data on this project, because, in this project, there is a target, Price and many numeric inputs. In Decision tree, it can cover the information classified by Region and so on. What is most important is that the Decision tree models are transparent and interpretable. It can be explained in plain English to the clients.

After that, I will try another tree-based model, Random Forest. It is an advanced Algorithm with a bagging method based on a tree model. The resulting model is usually better than the normal tree-based model. Hopefully, it will be better than Decision tree.

Finally, I will try the Linear Regression. It is the same as the tree-based model as well in the easy explanation features. In addition, I can find the relationship between the features and the target, which can meet the objectives of the project. Also, due to the majority of features are numeric, it should be suitable for this project.

Each of these models have their advantages, but to choose the best model, it is necessary to evaluate them.

6.3 BUILD/SELECT APPROPRIATE MODEL(S) AND CHOOSE RELEVANT PARAMETER(S)

In this section, I will build three models mentioned above, and choose the relevant parameters.

In addition, since the tool we used in the previous steps is pandas, we need an extra step before modeling: convert Pandas data into Spark data. The processes are shown in figure 31.

In [51]:
```
import findspark
findspark.init('/home/ubuntu/spark-2.1.1-bin-hadoop2.7')
import pyspark
from pyspark.sql import *
spark = SparkSession.builder.appName('dataframe_transformation').getOrCreate()
spark = SparkSession.builder.appName('pandasToSparkDF').getOrCreate()
```

In [52]:
```
from pyspark.sql.types import *

mySchema = StructType([ StructField("Suburb", IntegerType(), True)\
                       ,StructField("Rooms", IntegerType(), True)\
                       ,StructField("Type", IntegerType(), True)\
                       ,StructField("Price", FloatType(), True)\
                       ,StructField("Distance", FloatType(), True)\
                       ,StructField("Bathroom", FloatType(), True)\
                       ,StructField("Car", FloatType(), True)\
                       ,StructField("Landsize", FloatType(), True)\
                       ,StructField("BuildingArea", FloatType(), True)\
                       ,StructField("YearBuilt", FloatType(), True)\
                       ,StructField("CouncilArea", IntegerType(), True)\
                       ,StructField("Lattitude", FloatType(), True)\
                       ,StructField("Longtitude", FloatType(), True)\
                       ,StructField("Regionname", IntegerType(), True)\
                       ,StructField("Propertycount", FloatType(), True)\
                       ,StructField("Age", FloatType(), True)])
```

In [53]:
```
df = spark.createDataFrame(data,schema=mySchema)
```

Figure 31

The first model is Decision tree. The process is shown in figure 32. I import the decision tree regressor from Pyspark for modeling. The figure 32 shows the decision tree model building process.

In [67]:
```
from pyspark.ml.regression import DecisionTreeRegressor,RandomForestRegressor
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import RegressionEvaluator
```

In [68]:
```
dtc = DecisionTreeRegressor(labelCol='Price', featuresCol='features')
```

Figure 32

In my opinion, there is no need on more parameter setting. The parameters are by default as:
maxDepth=5, maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, impurity="variance", seed=None, varianceCol=None (Pyspark.ml package).

I decide to change the max depth, which mean the maximum number of trees, which can increase the complexity of the model. The rest parameters will be by default.

```
In [69]: rfc = RandomForestRegressor(labelCol='Price',featuresCol='features')
```

Figure 33

The second model I will build is Random Forest, which is an ensemble model of decision trees. As shown in figure 33, I import the Random forest regressor from pyspark for modeling and make regression for regression. This model has the similar parameters, and they are:
maxDepth=5, maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, impurity="variance", subsamplingRate=1.0, seed=None, numTrees=20, featureSubsetStrategy="auto" (Pyspark.ml package). The numTrees is the different point between decision tree and random forest.

I will try to change the max depth and numTrees. The numTrees is the number of trees. The rest I will keep the default.

```
In [60]: import findspark
         findspark.init('/home/ubuntu/spark-2.1.1-bin-hadoop2.7')
         spark = SparkSession.builder.appName('linear_regression_adv').getOrCreate()

         from pyspark.ml.regression import LinearRegression

         lr = LinearRegression(labelCol='Price')
```

Figure 34

The final model I want to build is linear regression. As shown in figure 34, I import the Linear regression from pyspark for modeling. The parameters are maxIter=100, regParam=0.0, elasticNetParam=0.0, tol=1e-6, fitIntercept=True, standardization=True, solver="auto", weightCol=Non3, aggregationDepth=2 (Pyspark.ml package). Fit intercept

is to decide whether to calculate the intercept for this model. standardization is ignored when fit intercept is set to false. If true, the X will be standardized before regression.

I will keep all parameters by default because there is no such many differences between the different setting on linear regression.

Overall, I will change the parameters I chose above then compare these results to the default results.

7. Data Mining

7.1 CREATE AND JUSTIFY TEST DESIGNS

After selecting the final algorithm and making certain adjustments, in order to test the results of the algorithm, the project defines a function to how to split the data into training set and test set. It will be divided into the two subsets as shown in figure 35. I will use the spark package to split randomly. The training set is 80% and the test set is 20%, which is based on Pareto Principle (Pareto, 1964). This process aims to check whether the model is overfitting. For example, if I use the same data to test the model, it must answer correctly. After using this function, and then the analysis function is used to test the model, I can get the RMSE of the training set and the test set respectively. By comparing the RMSE of the two subsets, I can know whether the model is up to standard and whether exists the overfitting problem.

```
In [58]: train_data,test_data = final_data.randomSplit([0.8,0.2])

In [59]: # Let's see our training data.
         train_data.describe().show()

         # And our testing data.
         test_data.describe().show()
```

```
+-------+------------------+
|summary|             Price|
+-------+------------------+
|  count|              7144|
|   mean|1096099.0648096304|
| stddev| 685729.2846957089|
|    min|          131000.0|
|    max|         9000000.0|
+-------+------------------+

+-------+------------------+
|summary|             Price|
+-------+------------------+
|  count|              1742|
|   mean|1079727.3576349025|
| stddev| 652929.8572922107|
|    min|          220000.0|
|    max|         6400000.0|
+-------+------------------+
```

Figure 35

7.2 CONDUCT DATA-MINING – CLASSIFY, REGRESS, CLUSTER, ETC. (MODELS MUST EXECUTE)

This section will conduct the three models, decision trees, random forest and linear regression. The modeling of this algorithm is based on the splitting of the eight-two rule (Pareto, 1964).

- Decision trees

Figure 36 is the process I build the model. I use the training set to get the predictions of this model and use RMSE to see the accuracy of the results, the applied measurement is RMSE. When default, the error rate is 415,964 dollars.

```
In [85]: dtc = DecisionTreeRegressor(labelCol='Price', featuresCol='features', maxDepth = 20)

In [86]: # Train the models (it's three models, so it might take some time).
         dtc_model = dtc.fit(train_data)

In [87]: dtc_predictions = dtc_model.transform(test_data)

In [74]: # Let's do something a bit more complex in terms of printing, just so it's formatted nicer.
         evaluator = RegressionEvaluator(
             labelCol="Price", predictionCol="prediction", metricName="rmse")

In [88]: rmse_dtc = evaluator.evaluate(dtc_predictions)
         print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_dtc)

         Root Mean Squared Error (RMSE) on test data = 413487
```

Figure 36

Then the only parameter adjusted is the number of trees. This project raises it from the default value, which is none.

Table 3

Max_depth	RMSE
None = 5	415,964
10	385,263
20	413,487

The results are shown in Table 3. The first line is the name of parameters and RMSE, and the second line is the results of by default. The rest is the results of setting parameters. I increased the max depth; the optimal results are 385,263$ when the max depth is 10. Because when max depth is 20, the RMSE increases, I think the after 10 points, the error rate will increase.

- Random forest

Figure 37 is the process I build the Random forest model. The process is similar to the decision trees. When default, the RMSE is 373,954 dollars.

```
In [69]: rfc = RandomForestRegressor(labelCol='Price',featuresCol='features')

In [71]: rfc_model = rfc.fit(train_data)

In [73]: rfc_predictions = rfc_model.transform(test_data)

In [76]: rmse_rfc = evaluator.evaluate(rfc_predictions)
         print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_rfc)

         Root Mean Squared Error (RMSE) on test data = 373954
```

Figure 37

Then the parameter adjusted is the number of trees and max depth. This project raises it from the default value, which is none for max depth and 10 trees.

Table 4

Max depth	numTrees	RMSE
None=5	None=20	373,954
5	30	342,656
5	40	376,105
6	30	335,030
10	30	285,961

The results are shown in Table 4. I increased the max depth and numTrees, the optimal results are 285,961$ when the max depth is 10 and 30 trees. When max depth is fixed at 5, the number of trees increases, the optimal RMSE is at 30 trees. Then I keep the numTrees at 30 and the optimal is up to 10 because then I increase the max depth to 15, the system often crashes, which makes the stability of the model not very good

Then I think the optimal parameter is at 10 max depth and 30 trees.

- Linear regression

Figure 40 is the process I build the linear regression model. When default, the RMSE is 416,901 dollars.

```
In [61]: # Fit the model to the data.
         lrModel = lr.fit(train_data)

In [63]: # Let's evaluate the model against the test data.
         test_results = lrModel.evaluate(test_data)

In [64]: # Interesting results! This shows the difference between the predicted value and the test data.
         test_results.residuals.show()

         # Let's get some evaluation metrics (as discussed in the previous linear regression notebook).
         print("RSME: {}".format(test_results.rootMeanSquaredError))
```

```
+-------------------+
|          residuals|
+-------------------+
|  46854.66550607979|
|-148427.11132210493|
| 275747.32269109786|
| -87734.75924909115|
|-474925.13440196216|
|-215919.93029364944|
|  -96905.4233609289|
|-163532.22720625997|
|-316382.54899026453|
| 375136.14774109423|
|-179184.90167817473|
|-13499.508076056838|
| 130112.93588078022|
| -198144.6903720498|
| -507216.8293547034|
| 178163.45367483795|
| 208045.21718633175|
|  55867.65295098722|
| -396004.9636065513|
| -368817.4577573836|
+-------------------+
only showing top 20 rows

RSME: 416901.960949773
```

Figure 38

Therefore, the optimal model is Random Forest model with 10 max depth and 30 numTrees. The RMSE is 285,961$.

7.3 SEARCH FOR PATTERNS

According to the figure 39, the three most important factors are distance in decision tree, BuildingArea, YearBuilt. That means when the government builds the new house, they can pay more attention to distance and building area. This is in line with expectations, as houses usually have more area. There is no worried in Yearbuilt. Because the newly built properties are in almost the same year.

```
In [91]: feature_cols =X.columns
         dtc_model.featureImportances
         list(zip(feature_cols, dtc_model.featureImportances))

Out[91]: [('Suburb', 0.025625847229275134),
          ('Rooms', 0.017870429038580705),
          ('Type', 0.018921691415835676),
          ('Distance', 0.08425353227459668),
          ('Bathroom', 0.017715346034020658),
          ('Car', 0.02422845163964466),
          ('Landsize', 0.05932703310815278),
          ('BuildingArea', 0.331027881612696),
          ('YearBuilt', 0.11286348539286573),
          ('CouncilArea', 0.14225353673232183),
          ('Lattitude', 0.07560723381072885),
          ('Longtitude', 0.03899949192104502),
          ('Regionname', 0.03399364850127456),
          ('Propertycount', 0.01562734004578777),
          ('Age', 0.0016850512431741038)]
```

Figure 39

Secondly, in order to further explore the relationship between the features and the price I will use the lin_reg to see the intercept and coefficients.

As shown in Figure 40, when the distance is increased by one meter, the price is reduced by $31,437. As the BuildingArea increase one square meter, the price is increased by $1,746. The YearBuilt is shown a negative relationship with price. That means when older houses give higher price.

```
In [94]: list(zip(feature_cols, lrModel.coefficients))

Out[94]: [('Suburb', -763.8103150996283),
          ('Rooms', 167334.96807790457),
          ('Type', -122373.35202766026),
          ('Distance', -31437.19856700348),
          ('Bathroom', 230219.40641255202),
          ('Car', 53971.30415570267),
          ('Landsize', 27.22659284118836),
          ('BuildingArea', 1736.2797733412297),
          ('YearBuilt', -2016.948078991905),
          ('CouncilArea', -7149.4711431197975),
          ('Lattitude', -746936.2797842486),
          ('Longtitude', 584601.2192249328),
          ('Regionname', 75313.5926197775),
          ('Propertycount', 2.314763882355989),
          ('Age', 2016.9480774115389)]

In [97]: print("Intercept: {}".format(lrModel.intercept))

         Intercept: -108731584.93240024
```

Figure 40

8. Interpretation

8.1 STUDY AND DISCUSS THE MINED PATTERNS

This section will be based on the above steps to link the pattern found in this project with the actual situation.

As shown in the figure above, the three most important factors given by the decision tree algorithm are distance, BuildingArea, YearBuilt. As shown in Figure 39, the house with the highest house price, this project speculates that because the house usually has more building area. Secondly, the distances are closely related in reality, because if a house is in the metropolitan area, its value is usually higher.

In addition to the examples given in the decision tree model, the project also looks at the predictor importance given by the other two trial models, Random Forest, which is an advanced model of decision tree. First, the Random forest 's results show in figure 41. The three most important factors are the same as decision trees. The distance and BuildingArea have been discussed in the random forest. But what I want to talk about especially is age, the mirror of YearBuilt and its feature importance is quite close to YearBuilt. Age is also the same as expected, the older house, the higher the price of the house. Fortunately, as speculated by 2.3 section, housing density is the least important reason, which is better help for the implementation of the plan.

```
In [92]: rfc_model.featureImportances
         list(zip(feature_cols, rfc_model.featureImportances))

Out[92]: [('Suburb', 0.01880966643956359),
          ('Rooms', 0.08692710890763647),
          ('Type', 0.029060852677554472),
          ('Distance', 0.05578543657668496),
          ('Bathroom', 0.06590042113034525),
          ('Car', 0.00946003199161765),
          ('Landsize', 0.022589797704872395),
          ('BuildingArea', 0.27549420663733026),
          ('YearBuilt', 0.08380551469466524),
          ('CouncilArea', 0.13041280102216327),
          ('Lattitude', 0.07277325751183526),
          ('Longtitude', 0.03221590047422266),
          ('Regionname', 0.05732277517654255),
          ('Propertycount', 0.004731343757573111),
          ('Age', 0.05471088529739278)]
```

Figure 41

As a result, the project can find out the most important attributes are the distance, Building Area and the age of houses.

```
In [65]: # We can also get the R2 value.
         print("R2: {}".format(test_results.r2))

         R2: 0.6329085975400606
```

Figure 42

Also, we can use the R square function in pyspark to the explanatory power of the linear regression model. There are 63% features which can be explained with Price.

8.2 VISUALIZE THE DATA, RESULTS, MODELS, AND PATTERNS

In the previous steps, the project has visualized many of the results. For example, in Section 2.3, the project explores the most important factors displayed in the initial data, and according to Figure 2, Figure 3, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9 and Figure 10, speculated the relationship between these factors and the price and the reason. In Figures 36, Figure 37 and 38, this project shows the results of the model run. These results are basically consistent with the speculation and exploration in Section 2.3.

In addition to the above summary, in this section, the project will visualize the patterns described in 7.3 and 8.1 and try to explain.

```
In [98]: sns.lmplot(x='Distance', y='Price', data = data)
Out[98]: <seaborn.axisgrid.FacetGrid at 0x1e124843978>
```

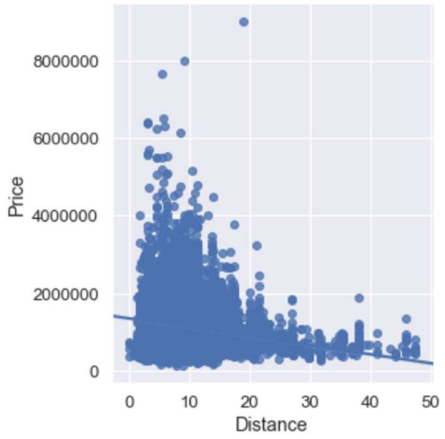


Figure 43

As shown in Figure 43, with the increase in distance, the house is cheaper. That means the farther properties are, the lower prices are. They show the negative relationship, which is consistent with figure 44, the coefficient graph.

Because I have shown the graphs of BuildingArea and Age-related to price in figure 7 and 22 right-hand sides. They show these two parameters are positively related to the price,

which consistent the coefficient in figure 44. Then to more understanding with these two parameters, I will show the relationship between them in figure 44. Distance is negatively related to Age. That means if the property is farther from the city center, the house could be newer. It makes senses because the government often build new properties in the suburb area.

```
In [118]: sns.lmplot(x='Distance', y='Age', data = data)
Out[118]: <seaborn.axisgrid.FacetGrid at 0x1e12dda81d0>
```

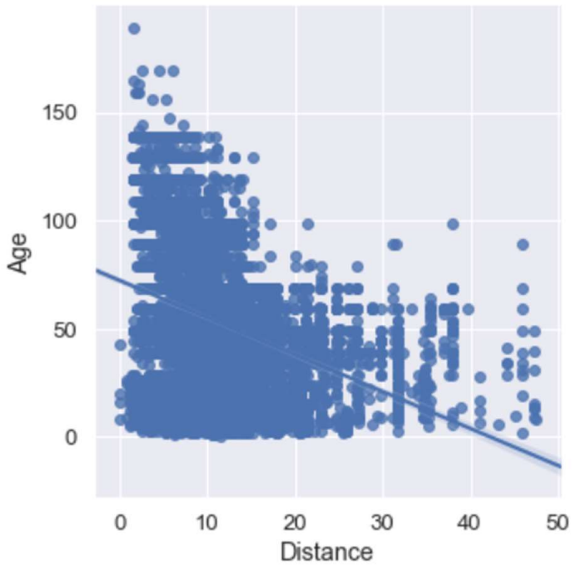


Figure 44

In figure 45, I show the summary of results from the models. They the coefficients from the linear regression, two feature importance from decision tree and random forest. The most three important features are distance, Building Area and Age. As shown in Figure 45, when the distance is increased by one meter, the price is reduced by $31,437. As the BuildingArea increase one square meter, the price is increased by $1,736. As the Age increase one year, the price is increased by $2016. Because the age reflects Yearbuilt, their coefficients are the same except the sign. That means as the YearBuilt increase one year, the price is decreased by $2016. YearBuilt increasing means age decreasing.

```
In [94]: list(zip(feature_cols, lrModel.coefficients))

Out[94]: [('Suburb', -763.8103150996283),
          ('Rooms', 167334.96807790457),
          ('Type', -122373.35202766026),
          ('Distance', -31437.19856700348),
          ('Bathroom', 230219.40641255202),
          ('Car', 53971.30415570267),
          ('Landsize', 27.22659284118836),
          ('BuildingArea', 1736.2797733412297),
          ('YearBuilt', -2016.948078991905),
          ('CouncilArea', -7149.4711431197975),
          ('Lattitude', -746936.2797842486),
          ('Longtitude', 584601.2192249328),
          ('Regionname', 75313.5926197775),
          ('Propertycount', 2.314763882355989),
          ('Age', 2016.9480774115389)]

In [97]: print("Intercept: {}".format(lrModel.intercept))

         Intercept: -108731584.93240024

In [92]: rfc_model.featureImportances
         list(zip(feature_cols, rfc_model.featureImportances))

Out[92]: [('Suburb', 0.01880966643956359),
          ('Rooms', 0.08692710890763647),
          ('Type', 0.029060852677554472),
          ('Distance', 0.05578543657668496),
          ('Bathroom', 0.06590042113034525),
          ('Car', 0.00946003199161765),
          ('Landsize', 0.022589797704872395),
          ('BuildingArea', 0.27549420663733026),
          ('YearBuilt', 0.08380551469466524),
          ('CouncilArea', 0.13041280102216327),
          ('Lattitude', 0.07277325751183526),
          ('Longtitude', 0.03221590047422266),
          ('Regionname', 0.05732277517654255),
          ('Propertycount', 0.004731343757573111),
          ('Age', 0.05471088529739278)]

In [91]: feature_cols =X.columns
         dtc_model.featureImportances
         list(zip(feature_cols, dtc_model.featureImportances))

Out[91]: [('Suburb', 0.025625847229275134),
          ('Rooms', 0.017870429038580705),
          ('Type', 0.018921691415835676),
          ('Distance', 0.08425353227459668),
          ('Bathroom', 0.017715346034020658),
          ('Car', 0.02422845163964466),
          ('Landsize', 0.05932703310815278),
          ('BuildingArea', 0.331027881612696),
          ('YearBuilt', 0.11286348539286573),
          ('CouncilArea', 0.14225353673232183),
          ('Lattitude', 0.07560723381072885),
          ('Longtitude', 0.03899949192104502),
          ('Regionname', 0.03399364850127456),
          ('Propertycount', 0.01562734004578777),
          ('Age', 0.0016850512431741038)]
```

Figure 45

8.3 INTERPRET THE RESULTS, MODELS, AND PATTERNS

According to RMSE as the standard, the project selected random forest as the prediction model, but also used the decision trees and linear regression model to make an analogy. The conclusion is that the distance, Age and BuildingArea can best influence the price. their relationships with Price are shown in figure 46.

At first, the closer the city is, the higher the price, because there will be more facilities and entertainment close to the city center, and the house will bring more convenience.

Whether it is close to the metropolitan area and the distance has the same reason, people are willing to pay more premiums to get the convenience of life.

Then older houses usually have a higher price. I guess because the older the house is, the possibility of more complete facilities is greater because people have more time to build. For these convenient facilities, its price may be higher.

In addition, the house has a higher probability of having more Building area and having more Building area also potentially means more area to live. People usually prefer a more spacious living environment, for which people will be willing to pay a higher price.

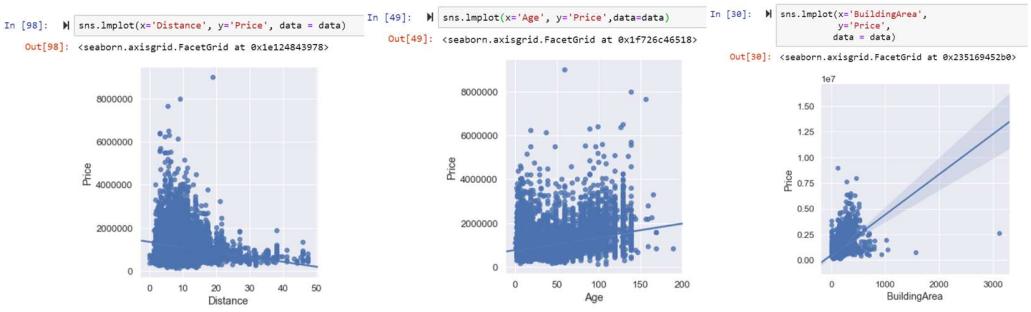


Figure 46

8.4 ASSESS AND EVALUATE RESULTS, MODELS, AND PATTERNS

In the previous section, the content of the assessment has been mentioned. This section will summarize the previous conclusions.

First, the project selected three models at the beginning, Decision Tree, Linear Regression and Random forests. According to the RMSE mentioned above which is Figure 36, 37 and 38, the project considers random forests to be the best choice because the RMSE of the other two models is relatively larger do not meet the objectives of this project. These error rates are slightly beyond the data mining objectives of this project. Figure 46 shows the RMSE of these three models when the parameters are set at optimal. Because every time rerun the model, the results will be different, but they are similar, the figure 46 results of RMSE are different what we mentioned above due to the different splitting training and testing random set, but the differences are not so large.

```
In [72]: rmse_dtc = evaluator.evaluate(dtc_predictions)
         print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_dtc)

         Root Mean Squared Error (RMSE) on test data = 401234

In [76]: rmse_rfc = evaluator.evaluate(rfc_predictions)
         print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_rfc)

         Root Mean Squared Error (RMSE) on test data = 335030

In [102]: # Let's get some evaluation metrics (as discussed in the previous linear regression notebook).
          print("RSME: {}".format(test_results.rootMeanSquaredError))

          RSME: 417441.71165983996
```

Figure 47

Among all three models I build, the random forest model is qualified with the objective about the RMSE. The decision tree is the second. The worst one is linear regression.

Also, this project finds the three key features to meet the requirement of another objectives. They are Age, Distance and BuildingArea, which are confirmed in both two tree-based model.

Through the completion above, the pricing model meets the needs, and the overall project complete the objectives.

8.5 ITERATE PRIOR STEPS (1 – 7) AS REQUIRED

In the first step, the project introduces the basic information of Melbourne's housing shortages, such as the funders and their needs for the project. The project also identified the final goals and data mining objectives of the project. What I did is shown in figure 48.

1. Situation Understanding 3
1.1 Identify the objectives of the situation 3
1.2 Assess the situation 4
1.3 Determine data mining objectives 5
1.4 Produce a project plan 6

Figure 48

In the second step, the project first introduces the source and content of the data set. Since the data exploration step is too complicated and time consuming to be inefficient in Pyspark for the project, the project decided to continue using Pandas as a tool for data visualization. Then the relationship between the raw data was initially observed to make some hypotheses. Finally, I also found out the problem with the data set, missing value and outliers. As shown in figure 49 it is some of my process in this step.

```
In [ ]: data['CouncilArea'].value_counts().head(30).plot.bar()

In [ ]: fig, ax = plt.subplots(figsize=(15,7))
        sns.set(font_scale=1.2)
        sns.heatmap(data.isnull(),yticklabels = False, cbar = False, cmap = 'Greys_r')
        plt.show()

In [ ]: data.isnull().sum()

In [ ]: sns.set_style('darkgrid')

        sns.boxplot(data = data, x = 'Rooms', y = 'Price')

In [ ]: sns.set_style('darkgrid')

        sns.boxplot(data = data, x = 'Bathroom', y = 'Price')

In [ ]: out_hori = data['BuildingArea']
        out_horilist = []
        out_horilist2 = []
        for values1 in out_hori:
            out_horilist += [values1]
            out_horilist2 += [values1]
        out_horilist.sort()
        new_q1, new_q3= np.percentile(out_horilist,[25,75])

        new_iqr = new_q3 - new_q1
        lower_bound1 = new_q1 - (1.5 * new_iqr)
        upper_bound1 = new_q3 + (1.5 * new_iqr)
        print('lower_bound:',lower_bound1)
        print('upper_bound:',upper_bound1)
        outliers2 = []

        for index1 in data.BuildingArea:
            if index1 > upper_bound1 or index1 < lower_bound1:
                outliers2 += [index1]
        print(len(outliers2))
```

Figure 49

In the third step, the project has been cleaned and pre-processed. Since the data preprocessing step is too complicated and time consuming to be inefficient in Pyspark for the project, the project decided to continue using Pandas as a tool for data preprocessing. According to the problem mentioned in the second step, the project handles missing data here. The method used is to directly discard the instance containing the missing value because the data is large enough. If I fill in the data at will, it is likely to cause inaccuracies in the final model. In addition, the project also removes some unimportant attributes based on the feature selection function. Due to concerns about the characteristics of the house, this project does not deal with outliers and extreme values. Finally, the project also

did the work of forming new attributes, making the data less complicated. As shown in figure 50, it is some of my process in this step.

```
In [ ]:  data['Age']= 2019 - data.YearBuilt
         X['Age']= data.Age

In [ ]:  print(data.Age)

In [ ]:  data1 = pd.read_csv("Housing Pricing 1.csv")
         data2 = pd.read_csv("Housing Pricing 2.csv")
         dataset = data1.append(data2)

         export_csv = dataset.to_csv (r'Newdata.csv', index = None, header=True)
         final_set = pd.read_csv("Newdata.csv")

In [ ]:  final_set.info()
```

Figure 50

In the fourth step, the project again uses the feature selection function in python, which mentioned in section 2 to reduce the data. I also reduce the bedrooms due to similar covariances. This project projects the numeric rooms into three categories. As shown in figure 51, it is some of my process in this step.

```
In [ ]:  data = data.drop("Bedroom2",1)
         X = data.drop("Landsize" and "Price",1)

In [ ]:  X.info()

In [ ]:  data["Rooms"].hist()

In [ ]:  Newrooms = []
         for group1 in data['Rooms']:
             if group1 < 3:
                 Newrooms += [1]

             elif group1 == 3 :

                 Newrooms += [2]

             else:
                 Newrooms += [3]
         data['Rooms'] =  Newrooms

         data.head(30)

In [ ]:  data.info()

In [ ]:  data["Rooms"].hist()
```

Figure 51

In the fifth step, the project discussed different methods and found a matching data mining method based on the target. Each of the methods considered has its own excellent place, but it is not suitable for this project.

In the sixth step, after analyzing and discussing the project, the random forest, decision tree and linear regression was finally selected according to their advantages and suitable situations. The project then introduces the parameters that may be of interest to customers in the models and selects the parameters that may be used during the modeling process. As shown in figure 52, it is some of my process in this step.

```
In [ ]: # The input columns are the feature column names, and the output column is what you'd like the new column to be named.
assembler = VectorAssembler(
    inputCols=["Suburb","Rooms",
               "Type",'Distance',
              "Bathroom","Car",
              "Landsize","BuildingArea",
              'YearBuilt',"CouncilArea",
              "Lattitude","Longtitude",
              "Regionname","Propertycount",
              "Age"],
    outputCol="features")

In [ ]: # Now that we've created the assembler variable, let's actually transform the data.
output = assembler.transform(df)

In [ ]: final_data = output.select("features",'Price')
final_data.printSchema()

In [ ]: train_data,test_data = final_data.randomSplit([0.8,0.2])

In [ ]: # Let's see our training data.
train_data.describe().show()

# And our testing data.
test_data.describe().show()

In [ ]: import findspark
findspark.init('/home/ubuntu/spark-2.1.1-bin-hadoop2.7')
spark = SparkSession.builder.appName('linear_regression_adv').getOrCreate()

from pyspark.ml.regression import LinearRegression

In [ ]: lr = LinearRegression(labelCol='Price')
```

Figure 52

In the seventh step, the project is selected according to the fifth step and the sixth step and attempts to create a test method for the model already has it. In this step, I adjusted a parameter and observed the best result based on RMSE. After combining these results, I can get the best model for this study. Finally, in the complex model, the available information is summarized in a pattern. As shown in figure 53, it is some of my process in this step.

```
In [ ]: rmse_rfc = evaluator.evaluate(rfc_predictions)
        print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_rfc)

In [ ]: # Select example rows to display.
        dtc_predictions.select("prediction", "Price", "features").show(5)
        rfc_predictions.select("prediction", "Price", "features").show(5)

In [ ]: feature_cols =X.columns
        dtc_model.featureImportances
        list(zip(feature_cols, dtc_model.featureImportances))

In [ ]: rfc_model.featureImportances
        list(zip(feature_cols, rfc_model.featureImportances))

In [ ]: list(zip(feature_cols, lrModel.coefficients))

In [ ]: print("Intercept: {}".format(lrModel.intercept))
```

Figure 53

Acknowledge

"I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright. (See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.html).

I also acknowledge that I have appropriate permission to use the data that I have utilised in this project. (For example, if the data belongs to an organisation and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data."

In addition to upload this project on Canvas, I also submit it through Github. Here is the link: https://github.com/AKKKKKKKKI/iteration_4_inforsys722

The process is shown in figure 54.

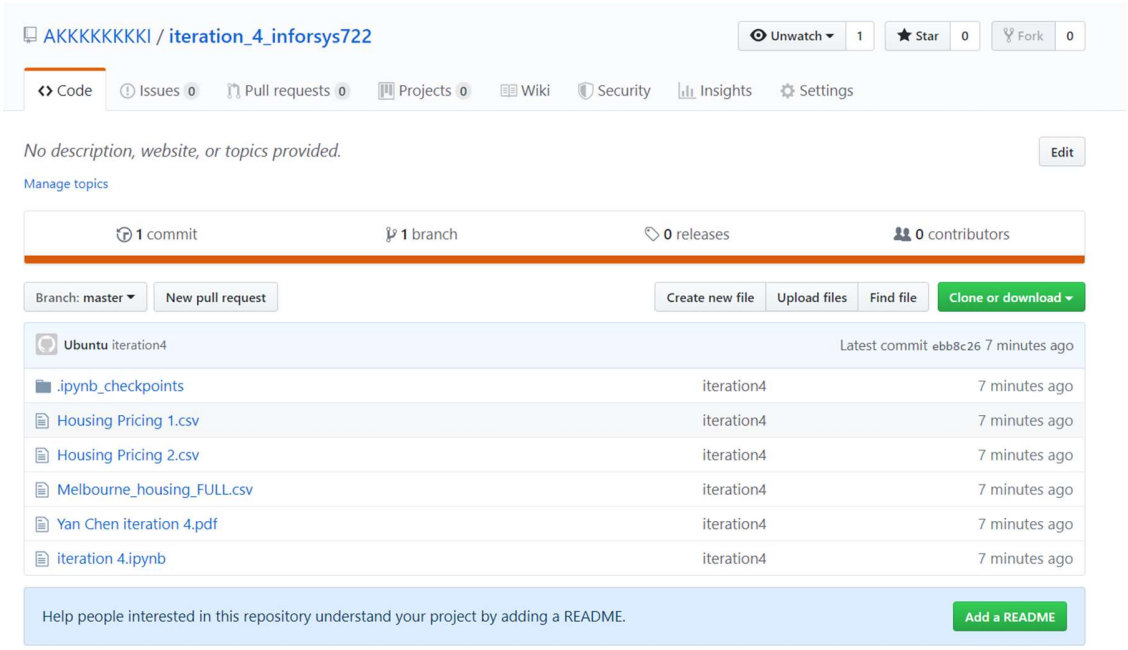

AKKKKKKKKI / iteration_4_inforsys722
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Settings
No description, website, or topics provided.
Edit
Manage topics
1 commit
1 branch
0 releases
0 contributors
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
Ubuntu iteration4
Latest commit ebb8c26 7 minutes ago
.ipynb_checkpoints iteration4 7 minutes ago
Housing Pricing 1.csv iteration4 7 minutes ago
Housing Pricing 2.csv iteration4 7 minutes ago
Melbourne_housing_FULL.csv iteration4 7 minutes ago
Yan Chen iteration 4.pdf iteration4 7 minutes ago
iteration 4.ipynb iteration4 7 minutes ago
Help people interested in this repository understand your project by adding a README.
Add a README

Figure 54

References

Australia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Australia

Galindo, J., & Tamayo, P. (2000). Credit Risk Assessment Using Statisticaland Machine Learning: Basic Methodology and Risk Modeling Applications. *Computational Economics*, 108-142.

Kashid, V. (2018). Retrieved from Quora: https://www.quora.com/What-is-the-difference-between-regression-classification-and-clustering-in-machine-learning

Linear Rregression. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Linear_regression

Pareto, V. (1964). *Cours d'économie politique.* Librairie Droz.

Pyspark.ml package. (n.d.). Retrieved from Spark: https://spark.apache.org/docs/2.1.0/api/python/pyspark.ml.html#pyspark.ml.regression.LinearRegressionModel