

**PROJEK UAS: Clustering, Linear Model & Tree Based Model**

**Anak Agung Ayu Diva Shanty Darmawan – 01112190018**

Pernyataan Kejujuran:

Saya Anak Agung Ayu Diva Shanty Darmawan, menyatakan bahwa saya mengerjakan soal-soal ini secara mandiri tanpa bantuan orang lain ataupun memberikan bantuan kepada orang lain. Jika saya terbukti melakukan kecurangan tersebut (mendapat bantuan dari orang lain ataupun memberi bantuan kepada orang lain) maka saya bersedia untuk tidak lulus dalam mata kuliah ini.

Untuk projek UAS akan disimulasikan clustering, linear based model/additive model serta tree based model. Data yang digunakan adalah melb\_data\_clean.csv yang merupakan data penjualan rumah di Melbourne, Australia. Sebelum memulai untuk simulasi data, berikut adalah beberapa library yang akan digunakan dalam projek ini.

```
rm(list=ls())
library(readr)
library(gam)
library(foreach)
library(splines)
library(ggplot2)
library(caret)
library(randomForest)
library(ggpubr)
library(gbm)
library(car)
```

Fungsi function `rm(list = ls())` adalah untuk menghapus *environment* pada rstudio agar mempermudah kita dalam mengolah data sehingga data tidak akan tercampur dengan data lain dari projek lain.

## 1. Data Import

Data yang akan digunakan adalah data penjualan rumah di Melbourne yang berjumlah 5179 data. Pertama, data akan di import ke dalam rstudio.

```
melb = read.csv("melb_data_clean.csv")
head(melb)
```

ID	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	
1	1 Abbotsford	25 Bloomberg St	2	h	1035000	S	Biggin	4/02/2016	2.5	3067	2	1	0	
2	2 Abbotsford	5 Charles St	3	h	1465000	SP	Biggin	4/03/2017	2.5	3067	3	2	0	
3	3 Abbotsford	55a Park St	4	h	1600000	VB	Nelson	4/06/2016	2.5	3067	3	1	2	
4	4 Abbotsford	124 Yarra St	3	h	1876000	S	Nelson	7/05/2016	2.5	3067	4	2	0	
5	5 Abbotsford	98 Charles St	2	h	1636000	S	Nelson	8/10/2016	2.5	3067	2	1	2	
6	6 Abbotsford	10 Valiant St	2	h	1097000	S	Biggin	8/10/2016	2.5	3067	3	1	2	
Landsize	BuildingArea	YearBuilt	CouncilArea	Latitude	Longitude	Regionname		Propertycount	SalesYear	EffAge				
156	79	1900	Yarra	-37.8079	144.9934	Northern Metropolitan		4019	2016	100				
134	150	1900	Yarra	-37.8093	144.9944	Northern Metropolitan		4019	2017	100				
120	142	2014	Yarra	-37.8072	144.9941	Northern Metropolitan		4019	2016	2				
245	210	1910	Yarra	-37.8024	144.9993	Northern Metropolitan		4019	2016	100				
256	107	1890	Yarra	-37.8060	144.9954	Northern Metropolitan		4019	2016	100				
220	75	1900	Yarra	-37.8010	144.9989	Northern Metropolitan		4019	2016	100				

i-25 of 24 columns

Dengan function **read.csv()** , data di *import* dari komputer dan disimpan di dalam variable melb. Dapat dilihat beberapa data awal yang disimpan di dalam melb dari hasil function **head()** .

Sesuai dengan perintah yang diberikan untuk projek ini, kita akan menghapus beberapa variable yang ada di dalam melb yang tidak akan digunakan.

```
names(melb)
[1] "ID"                 "Suburb"             "Address"            "Rooms"
[5] "Type"               "Price"              "Method"             "SellerG"
[9] "Date"               "Distance"           "Postcode"           "Bedroom2"
[13] "Bathroom"           "Car"                "Landsize"           "BuildingArea"
[17] "YearBuilt"           "CouncilArea"        "Latitude"           "Longitude"
[21] "Regionname"         "Propertycount"     "SalesYear"          "EffAge"

drop = c("Suburb", "Address", "Method", "SellerG",
        "Date", "Postcode", "Bedroom2", "Bathroom",
        "YearBuilt", "Latitude", "Longitude",
        "Propertycount", "SalesYear")

melb2 = melb[, !(names(melb) %in% drop)]
head(melb2)
```

ID <int>	Rooms <int>	Type <chr>	Price <dbl>	Distance <dbl>	Car <chr>	Landsize <int>	BuildingArea <dbl>	CouncilArea <chr>	Regionname <chr>	EffAge <int>
1	1	2 h	1035000	2.5	0	156	79	Yarra	Northern Metropolitan	100
2	2	3 h	1465000	2.5	0	134	150	Yarra	Northern Metropolitan	100
3	3	4 h	1600000	2.5	2	120	142	Yarra	Northern Metropolitan	2
4	4	3 h	1876000	2.5	0	245	210	Yarra	Northern Metropolitan	100
5	5	2 h	1636000	2.5	2	256	107	Yarra	Northern Metropolitan	100
6	6	2 h	1097000	2.5	2	220	75	Yarra	Northern Metropolitan	100

Dengan fungsi **names()** , dengan mudah kita mengetahui nama-nama kolom variable yang tersimpan di dalam **melb**. Nama-nama tersebut akan kita simpan di variable **drop**, dengan fungsi **c()** untuk menyatukan mereka. Variable yang tersisa dari melb akan disimpan di dalam **melb2** dan dari function **head()** dapat kita lihat beberapa data awal dari **melb2**.

```

summary(melb2)

      ID      Rooms       Type        Price
Min.   : 1    Min.   :1.000   Length:5179    Min.   :131000
1st Qu.:1296  1st Qu.:3.000   Class  :character 1st Qu.: 700000
Median :2590  Median :3.000   Mode   :character  Median : 960000
Mean   :2590  Mean   :3.119
3rd Qu.:3884  3rd Qu.:4.000
Max.   :5179  Max.   :8.000

      Distance     Car      Landsize     BuildingArea
Min.   : 0.7  Length:5179    Min.   : 1.0  Min.   : 1.0
1st Qu.: 6.6  Class  :character 1st Qu.: 248.0 1st Qu.: 104.0
Median : 9.7  Mode   :character  Median : 488.0  Median : 133.0
Mean   :10.4
3rd Qu.:13.0
Max.   :47.4

      CouncilArea   Regionname     EffAge
Length:5179      Length:5179    Min.   : 0.00
Class :character  Class  :character  1st Qu.: 19.00
Mode  :character  Mode   :character  Median : 52.00
                                         Mean   : 52.33
                                         3rd Qu.: 86.00
                                         Max.   :100.00

```

Dengan function **summary()**, dapat dilihat ringkasan dari variable-variable yang ada di dalam **melb2**. Ringkasan ini berbentuk range statistik dari quartil 1, min hingga max data.

## 2. Data Exploration

Pada bagian ini, akan dijelaskan mengenai *dependent variable* dan *independent variable* yang akan digunakan dalam projek ini.

### a. Dependent Variable

Dependent Variable adalah variable yang akan menjadi variable efek yang kita lihat untuk mengukur apakah berhasil modelnya atau tidak. Anggaplah *dependent variable* seperti efeknya dari model yang ingin kita simulasi. Variable yang kita gunakan sebagai *dependent variable* adalah variable **Price**.

```

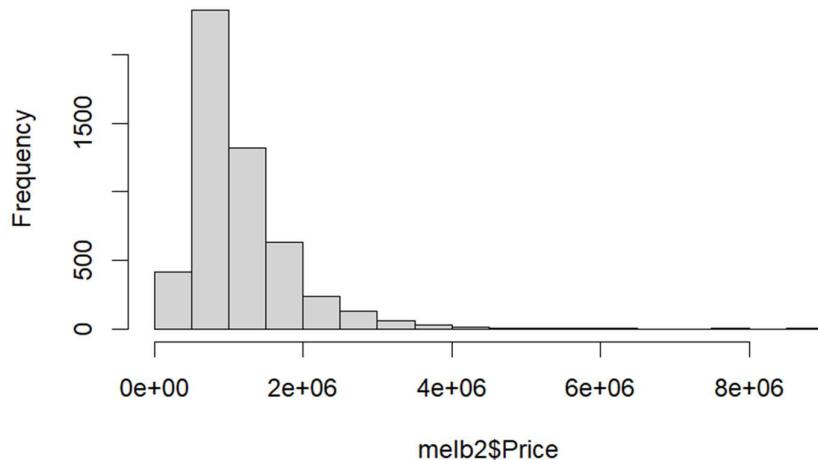
summary(melb2$Price)
      Min. 1st Qu. Median     Mean 3rd Qu.      Max.
      131000 700000 960000 1156211 1416375 9000000

```

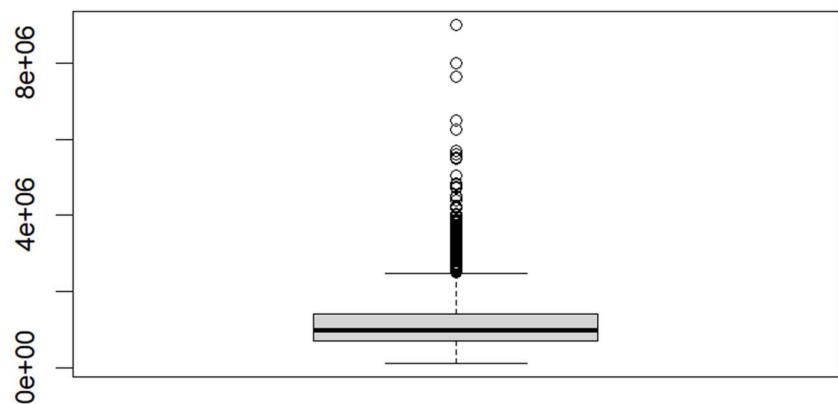
Dengan function **summary()**, kita dapat melihat secara lebih ringkas bagaimana bentuk dari variable **Price**.

```
hist(melb2$Price)
```

Histogram of melb2\$Price

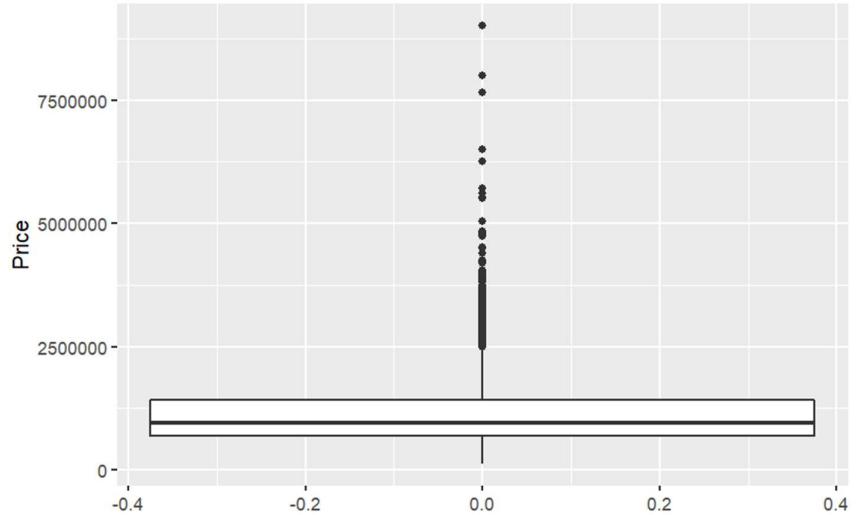


```
boxplot(melb2$Price)
```



Plot-plot di atas membantu kita untuk memvisualisasikan bagaimana bentuk variable **Price**. Dapat dilihat dari kedua plot di atas, bahwa data variable **Price** telihat *skewed*, lebih spesifiknya *strongly left skewed* apabila dilihat dari plot histogram nya.

```
ggplot(melb2) + geom_boxplot(aes(y = Price))
```



Dengan function di atas, kita dapat melihat visualisasi variable **Price** dengan data **melb2**. Dari plot yang dihasilkan, terlihat beberapa *outlier* yang ada dalam variable **Price**.

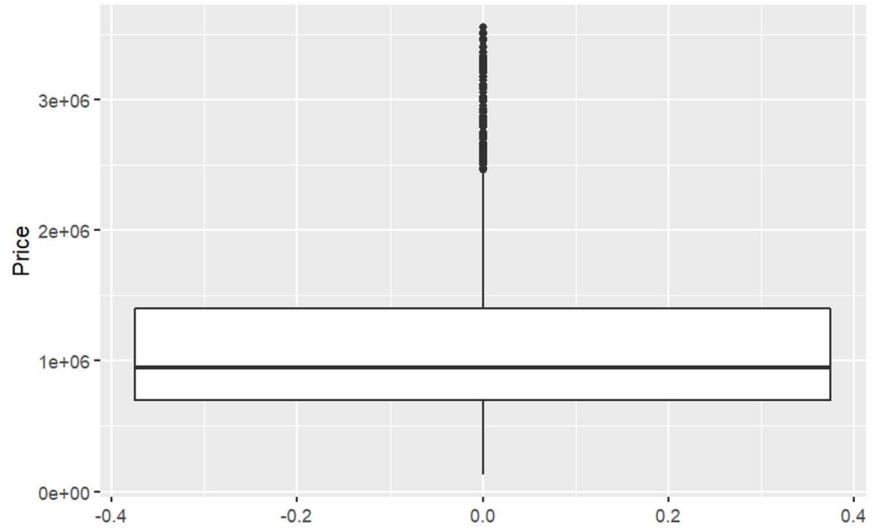
```
length(boxplot.stats(melb2$Price, coef = 3)$out)
[1] 55
```

Dengan function **length()** kita dapat mencari berapa banyak *outliers* yang ada di dalam variable **Price**. Hasilnya adalah 55, sehingga terdapat 55 *outliers* yang ada di dalam variable **Price**.

```
out = boxplot.stats(melb2$Price, coef = 3)$out
out_key = which(melb2$Price %in% c(out))
melb2 = melb2[-out_key, ]
```

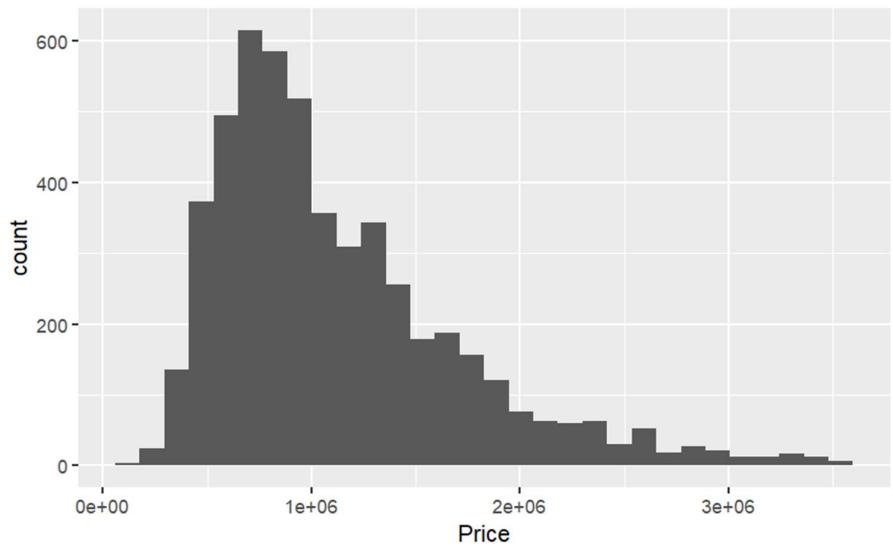
Dengan kumpulan fungsi di atas, dapat kita hapus *outliers* yang ada. Function **which()** bertugas untuk memilih data *outliers*, yang disimpan di dalam variable **out**, di dalam **melb2** khususnya variable **Price**. Kemudian *outliers* ini dihapus dengan function **melb2[-out\_key, ]** dan disimpan kembali data yang benar di dalam **melb2**.

```
ggplot(melb2) + geom_boxplot(aes(y = Price))
```



Berikut adalah plot visualisasi dari variable **Price** dengan `melb2` setelah dihapus *outliers* yang ada. Dapat dilihat bahwa datanya terlihat lebih rapi dan baik. Dapat dilihat juga dari histogram di bawah, datanya menjadi terlihat tidak *strongly skewed*.

```
ggplot(melb2) + geom_histogram(aes(x = Price))
```



## b. Independent Variable

Apabila *dependent variable* adalah efek, maka *independent variable* adalah penyebab dari efek tersebut. *Independent variable* digunakan sebagai variable yang akan mempengaruhi bagaimana hasil model yang akan dihasilkan. *Independent variable* yang digunakan pada kali ini adalah: **Type**, **Car**, **CouncilArea**, **Regionname**, **ID**, **Rooms**, **Distance**, **Landsize**, **BuildingArea**, dan **EffAge**.

```
table(melb2$Type)
```

h	t	u
3957	516	651

```
table(melb2$Car)
```

>4	0	1	2	3
252	376	1938	2273	285

```
table(melb2$CouncilArea)
```

Banyule	Bayside	Boroondara
269	188	475
Brimbank	Cardinia	Casey
190	5	16
Darebin	Frankston	Glen Eira
381	30	348
Greater Dandenong	Hobsons Bay	Hume
21	198	94
Kingston	Knox	Macedon Ranges
105	42	5
Manningham	Maribyrnong	Maroondah
146	350	34
Melbourne	Melton	Monash
117	41	163
Moonee Valley	Moreland	Nillumbik
422	574	18
Port Phillip	Stonnington	Whitehorse
183	190	129
Whittlesea	Wyndham	Yarra
89	47	244
Yarra Ranges		
10		

```
table(melb2$Regionname)
```

Eastern Metropolitan	Eastern Victoria
548	23
Northern Metropolitan	Northern Victoria
1518	19
South-Eastern Metropolitan	Southern Metropolitan
151	1610
Western Metropolitan	Western Victoria
1241	14

```

summary(melb2$ID)

      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
           1       1297    2594     2594     3888     5179

summary(melb2$Rooms)

      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
     1.000   3.000   3.000     3.106   4.000    8.000

summary(melb2$Distance)

      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
      0.70    6.60    9.70     10.44   13.00    47.40

summary(melb2$Landsize)

      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
      1.0    245.8   480.0     559.8   657.0  37000.0

summary(melb2$BuildingArea)

      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
      1.0   103.0   132.0     151.3   180.0  3112.0

summary(melb2$EffAge)

      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
      0.00   19.00   52.00     52.22   86.00  100.00

```

Berikut adalah *summary* dari semua *independent variable* yang akan digunakan di dalam projek ini.

```

melb2 = melb2[melb2$Landsize != 0,]
melb2 = melb2[melb2$BuildingArea != 0,]

```

Karena tidak mungkin terdapat Landsize dan Building Area yang 0, maka dengan function di atas akan dihapuskan apabila ada data yang memiliki nilai 0 dan disimpan kembali di dalam melb2. Sekarang, akan kita lihat korelasi masing-masing independent variable yang numerical dengan dependent variable.

```

cor(melb2$Price, melb2$ID)
[1] -0.1997585

```

Dapat dilihat bahwa, hubungan antara Price dan ID menghasilkan nilai yang negatif yang artinya variable ID tidak terlalu signifikan bagi variable Price.

```
cor(melb2$Price, melb2$Rooms)
[1] 0.472127
```

Dapat dilihat bahwa, hubungan antara Price dan Rooms menghasilkan nilai yang cukup besar yang artinya variable Rooms memiliki korelasi yang moderate bagi variable Price.

```
cor(melb2$Price, melb2$Distance)
[1] -0.2697733
```

Dapat dilihat bahwa, hubungan antara Price dan Distance menghasilkan nilai yang negatif yang artinya variable Distance tidak terlalu signifikan bagi variable Price.

```
cor(melb2$Price, melb2$Landsize)
[1] -0.006769088
```

Dapat dilihat bahwa, hubungan antara Price dan Landsize menghasilkan nilai yang negatif yang artinya variable Landsize tidak terlalu signifikan bagi variable Price.

```
cor(melb2$Price, melb2$BuildingArea)
[1] 0.4760642
```

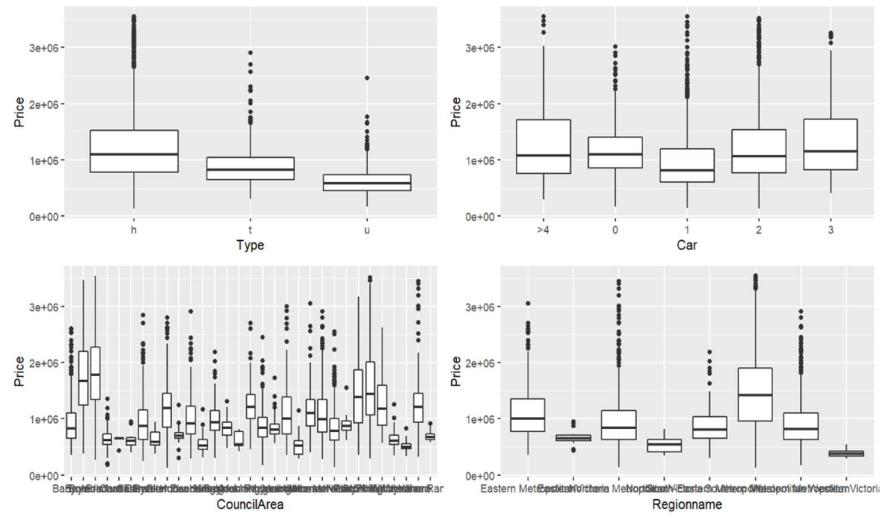
Dapat dilihat bahwa, hubungan antara Price dan BuildingArea menghasilkan nilai yang cukup besar yang artinya variable BuildingArea memiliki korelasi yang moderate bagi variable Price.

```
cor(melb2$Price, melb2$EffAge)
[1] 0.3121283
```

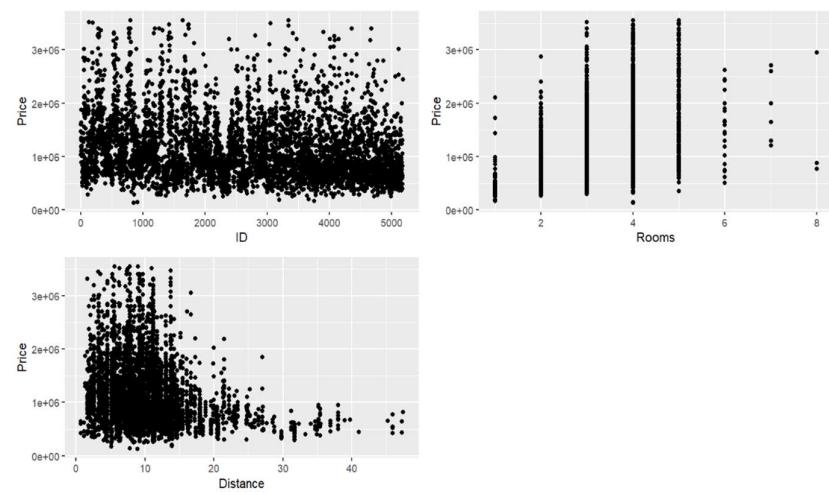
Dapat dilihat bahwa, hubungan antara Price dan EffAge menghasilkan nilai korelasi yang kecil yang artinya variable RMTOT memiliki korelasi yang lemah bagi variable Price.

Berikutnya, dapat kita bandingkan hasil korelasi yang sudah didapatkan dengan plot-plot untuk independent variable.

```
ggarrange(ggplot(melb2) + geom_boxplot(aes(Type, Price)),
          ggplot(melb2) + geom_boxplot(aes(Car, Price)),
          ggplot(melb2) + geom_boxplot(aes(CouncilArea,
Price)),
          ggplot(melb2) + geom_boxplot(aes(Regionname,
Price)))
```



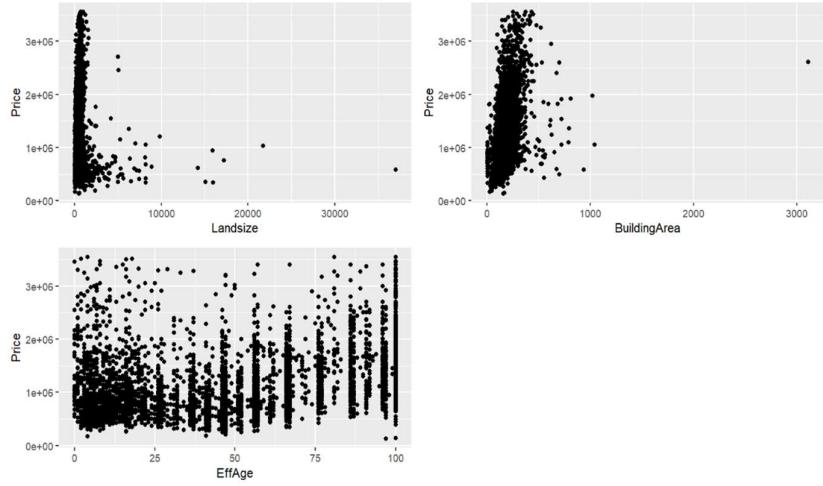
```
ggarrange(ggplot(melb2) + geom_point(aes(ID, Price)),
          ggplot(melb2) + geom_point(aes(Rooms, Price)),
          ggplot(melb2) + geom_point(aes(Distance,
Price)))
```



```

ggarrange(ggplot(melb2) + geom_point(aes(Landsize,
Price)),
          ggplot(melb2) + geom_point(aes(BuildingArea,
Price)),
          ggplot(melb2) + geom_point(aes(EffAge, Price)))

```



Dari plot, dapat dilihat juga bahwa terjadi kesamaan dengan hasil korelasi. Seperti Landsize dengan Price, yang menunjukkan hampir tidak adanya korelasi antara kedua variable. Sedangkan untuk Type dengan Price, dapat kita lihat bahwa Price ikut menurun sesuai dengan Type nya. Yang masih bisa dilihat juga outliers yang masih terdapat di masing-masing independent variable seperti pada plot Type dengan Price, dapat terlihat outliers pada Type 'u'.

### 3. Clustering

Clustering merupakan metode pengelompokan data yang merupakan bagian dari Data Mining. Pengelompokan ini dilakukan dengan membagi grup yang terdiri dari data-data atau objek abstrak ke dalam kelas atau grup yang terdiri dari data-data yang sama.

Pada projek ini, kita akan mencoba untuk mensimulasikan clustering pada data melb2 dengan 3 cluster.

```
melb2$Type = as.factor(melb2$Type)
melb2$CouncilArea = as.factor(melb2$CouncilArea)
melb2$Regionname = as.factor(melb2$Regionname)
melb2$Car = as.factor(melb2$Car)
melb2$Type = as.numeric(melb2$Type)
melb2$CouncilArea = as.numeric(melb2$CouncilArea)
melb2$Regionname = as.numeric(melb2$Regionname)
melb2$Car = as.numeric(melb2$Car)
```

Sebelum dapat memulai clustering, terdapat beberapa data yang harus diubah menjadi numerical menggunakan function as.factor(). dan as.numeric(). Data-data ini akan berubah menjadi angka dan ini dapat dilihat dibawah dari hasil function summary(). Terdapat juga function is.na() di bawah ini untuk menunjukkan apabila terdapat nilai NA di dalam data melb2. Apabila iya, maka function na.omit() akan mengeluarkan nilai tersebut dan disimpan kembali dalam melb2.

```
table(is.na(melb2))
    FALSE
    56364
melb2 = na.omit(melb2)
```

```

summary(melb2)
      ID          Rooms          Type
Min.   : 1    Min.   :1.000  Length:5124
1st Qu.:1297 1st Qu.:3.000  Class  :character
Median :2594  Median :3.000  Mode   :character
Mean   :2594  Mean   :3.106
3rd Qu.:3888 3rd Qu.:4.000
Max.   :5179  Max.   :8.000

      Price          Distance          Car
Min.   :131000  Min.   : 0.70  Length:5124
1st Qu.:700000 1st Qu.: 6.60  Class  :character
Median :950000  Median : 9.70  Mode   :character
Mean   :1119807  Mean   :10.44
3rd Qu.:1400000 3rd Qu.:13.00
Max.   :3550000  Max.   :47.40

      Landsize        BuildingArea        CouncilArea
Min.   : 1.0    Min.   : 1.000  Length:5124
1st Qu.: 245.8 1st Qu.: 103.0  Class  :character
Median : 480.0  Median : 132.0  Mode   :character
Mean   : 559.8  Mean   : 151.3
3rd Qu.: 657.0  3rd Qu.: 180.0
Max.   :370000.0 Max.   :3112.0

      Regionname        EffAge
Length:5124    Min.   : 0.00
Class  :character 1st Qu.: 19.00
Mode   :character  Median : 52.00
                           Mean   : 52.22
                           3rd Qu.: 86.00
                           Max.   :100.00

```

Selanjutnya, akan kita coba untuk simulasi clustering dengan data ini. Clustering dilakukan dengan function kmeans() yang membagi data menjadi 3 cluster dengan nstart = 100.

```

clustermod = kmeans(na.omit(melb2), 3, nstart = 100)
clustermod

K-means clustering with 3 clusters of sizes 2982, 461, 1681

Cluster means:
      ID Rooms Type    Price Distance Car Landsize
1 2807.965 2.817572 1.553991 730317.7 11.734474 3.328974 578.9209
2 2076.766 3.982642 1.021692 2487892.4 8.219306 3.557484 661.4620
3 2355.869 3.376562 1.092802 1435553.7 8.746520 3.431291 497.8531
BuildingArea CouncilArea Regionname EffAge
1 124.9408 15.47317 4.633803 43.17136
2 256.5871 10.98482 5.481562 65.35141
3 169.0529 15.52290 4.814396 64.66686

Clustering vector:
   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  123  124  125  126  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212

[ reached getOption("max.print") -- omitted 4124 entries ]

Within cluster sum of squares by cluster:
 [1] 1.109310e+14 7.684870e+13 9.633754e+13
 [1] [5] [9] 
 [between_SS / total_SS = 83.9 %)

Available components:
 [1] "cluster"   "centers"   "totss"     "withinss"
 [5] "tot.withinss" "betweenss" "size"      "iter"
 [9] "ifault"

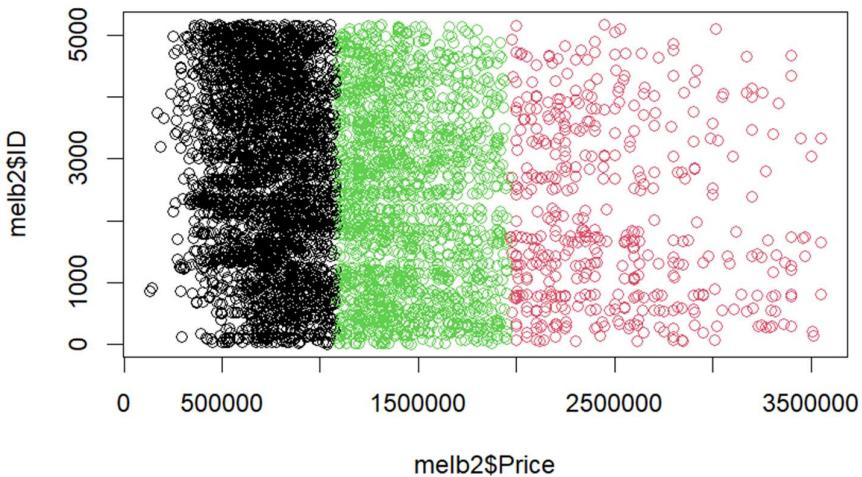
```

Dalam data melb2 tidak ada anggota independent variable yang dikeluarkan.

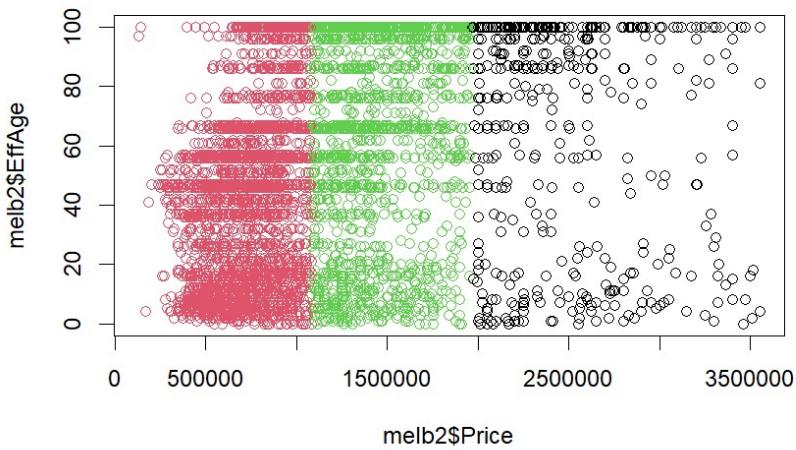
Hal ini karena pada saat disimulasikan dengan semua independent ada, atau dengan tidak ada salah satu variable CouncilArea atau variable Regionname, atau bahkan dengan keduanya tidak ada- tetap menghasilkan hasil yang sama untuk keempat tipe model cluster itu. Sehingga karena hal tersebut tidak ada independent variable yang dikeluarkan dan memilih untuk menggunakan semuanya.

Dapat dilihat dari presentase yang dihasilkan yaitu 83.9% bahwa dalam cluster yang sudah dibagi, besar kemiripan dalam anggota suatu grup yang sama. Dilihat juga dari sum of squares by cluster, bahwa angka kedua dan angka ketiga yang dihasilkan memiliki perbedaan yang tidak terlalu besar, sedangkan angka pertama yang dihasilkan jauh perbedaannya dengan yang lainnya, yang artinya cukup besar ketidaksamaan yang ada diantara anggota suatu grup.

```
plot(melb2$Price, melb2$ID, col = clustermod$cluster)
```



```
plot(melb2$Price, melb2$EffAge, col = clustermod$cluster)
```

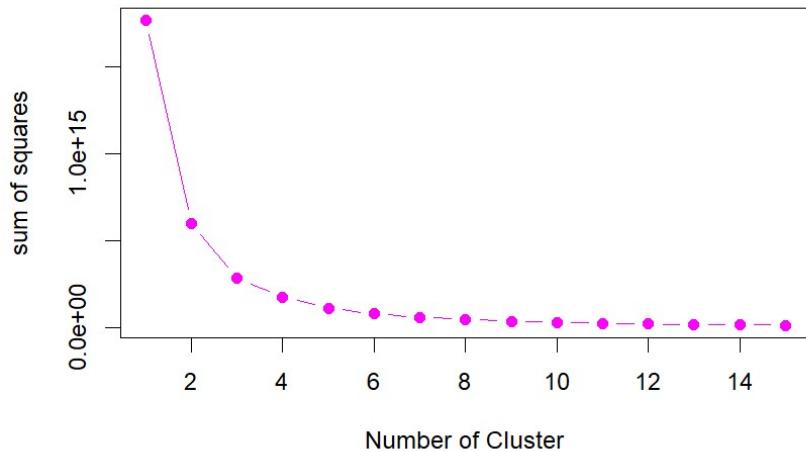


Dengan fungsi `plot()`, yang membantu untuk visualisasi antara dependent variable kita dan independent variable ID serta EffAge, dapat kita lihat bagaimana clustering yang terjadi dengan data kita. Masing-masing cluster dapat dilihat karena sudah di highlight oleh warna yang diperoleh dari hasil `clustermod$cluster`.

```

kmax = 15
wss = sapply(1:kmax, function(k) {kmeans(na.omit(melb2),
k, nstart = 100, iter.max = 15)$tot.withinss})
plot(1:kmax, wss, type = "b", pch = 19, xlab = "Number of
Cluster", ylab = "sum of squares", col = "magenta")

```



Function `sapply()` digunakan untuk mensimulasikan metode elbow yang akan mencari pembagian cluster yang optimum. Di sini, kita gunakan nilai cluster paling besar adalah 15 dengan `nstart` 5. Dapat dilihat dari function `plot()` yang telah disimulasikan bahwa mulai dari 3 cluster adalah pilihan yang tepat untuk data ini.

```

melb2$cluster = clustermod$cluster
summary(melb2)

      ID      Rooms       Type      Price
Min.   : 1   Min.   :1.000   Min.   :1.000   Min.   : 131000
1st Qu.:1297 1st Qu.:3.000  1st Qu.:1.000   1st Qu.: 700000
Median :2594 Median :3.000  Median :1.000   Median : 950000
Mean   :2594 Mean   :3.106  Mean   :1.355   Mean   :1119807
3rd Qu.:3888 3rd Qu.:4.000  3rd Qu.:1.000   3rd Qu.:1400000
Max.   :5179 Max.   :8.000  Max.   :3.000   Max.   :3550000
      Distance     Car      Landsize BuildingArea
Min.   : 0.70  Min.   :1.000   Min.   : 1.0   Min.   : 1.0
1st Qu.: 6.60  1st Qu.:3.000  1st Qu.: 245.8  1st Qu.: 103.0
Median : 9.70  Median :3.000  Median : 480.0  Median : 132.0
Mean   :10.44  Mean   :3.383  Mean   : 559.8  Mean   : 151.3
3rd Qu.:13.00 3rd Qu.:4.000  3rd Qu.: 657.0  3rd Qu.: 180.0
Max.   :47.40  Max.   :5.000  Max.   :37000.0  Max.   :3112.0
      CouncilArea    Regionname   EffAge   cluster
Min.   : 1.00  Min.   :1.000   Min.   : 0.00  Min.   :1.000
1st Qu.: 7.00  1st Qu.:3.000  1st Qu.: 19.00  1st Qu.:1.000
Median :17.00  Median :6.000  Median : 52.00  Median :1.000
Mean   :15.09  Mean   :4.769  Mean   : 52.22  Mean   :1.746
3rd Qu.:23.00 3rd Qu.:6.000  3rd Qu.: 86.00  3rd Qu.:3.000
Max.   :31.00  Max.   :8.000  Max.   :100.00  Max.   :3.000

```

Sekarang, kita akan memasukkan hasil cluster dari clustermod ke dalam data kita yaitu melb2 dan disimpan dalam melb2\$cluster. Dapat dilihat bagaimana data melb2 setelah dimasukkan cluster melalui hasil summary().

#### **4. Linear Based Model / Additive Model**

Linear model adalah suatu metode untuk memvisualisasikan variable respon atau dependent variable dalam hal kombinasi linier variable prediktor. Dalam Linear Model projek ini tersedia pilihan: Generalized Linear Modeling (GLM) atau Generalized Additive Model (GAM).

##### **a. Split Data Training and Data Testing**

Pada bagian clustering sebelumnya, terdapat beberapa data categorical yang harus diubah menjadi numerical agar dapat diolah. Untuk simulasi model Linear atau Additive, akan diubah kembali menjadi categorical menggunakan function factor(). Dapat dilihat summary dari data melb2 dengan function summary() setelah data categorical diubah kembali.

```
melb2$Type = factor(melb2$Type)
melb2$CouncilArea = factor(melb2$CouncilArea)
melb2$Regionname = factor(melb2$Regionname)
melb2$Car = factor(melb2$Car)
```

```

summary(melb2)

      ID      Rooms     Type     Price
Min.   : 1   Min.   :1.000  1:3957  Min.   :131000
1st Qu.:1297 1st Qu.:3.000  2: 516   1st Qu.: 700000
Median :2594 Median :3.000   3: 651   Median : 950000
Mean   :2594 Mean   :3.106   4: 106   Mean   :1119807
3rd Qu.:3888 3rd Qu.:4.000   5: 106   3rd Qu.:1400000
Max.   :5179 Max.   :8.000   6: 106   Max.   :3550000

      Distance     Car     Landsize     BuildingArea
Min.   : 0.70  1: 252   Min.   : 1.0   Min.   : 1.0
1st Qu.: 6.60  2: 376   1st Qu.: 245.8  1st Qu.: 103.0
Median : 9.70  3:1938   Median : 480.0  Median : 132.0
Mean   :10.44  4:2273   Mean   : 559.8  Mean   : 151.3
3rd Qu.:13.00 5: 285   3rd Qu.: 657.0  3rd Qu.: 180.0
Max.   :47.40          Max.   :37000.0  Max.   :3112.0

      CouncilArea    Regionname     EffAge     cluster
23      : 574       6   :1610   Min.   : 0.00  Min.   :1.000
3       : 475       3   :1518   1st Qu.: 19.00  1st Qu.:1.000
22      : 422       7   :1241   Median : 52.00  Median :1.000
7       : 381       1   : 548   Mean   : 52.22  Mean   :1.746
17      : 350       5   : 151   3rd Qu.: 86.00  3rd Qu.:3.000
9       : 348       2   :  23   Max.   :100.00  Max.   :3.000
(Other):2574 (Other): 33

```

Sebelum dapat memulai simulasi model, data akan dibagi dulu ke dalam data training dan data testing menggunakan fungsi `createDataPartition()` dari library(`caret`) dengan rasio 80:20. Data Splitting ini dilakukan berdasarkan variable `Type` sebagai salah satu independent variable yang bersifat categorical yang akan disimulasikan modelnya.

```

set.seed(136)
p = createDataPartition(melb2$type, p = 0.8, list =
FALSE)
train.d = melb2[p, ]
test.d = melb2[-p, ]

```

Dengan function `melb2[]` menggunakan `p`, dapat kita bagi data melb masuk ke dalam `train.d` dan `test.d`. Jumlah yang dihasilkan untuk kedua set data sudah sesuai mengikuti rasio 80:20, hal ini dapat dilihat dari hasil function `dim()` dibawah. Selanjutnya kita bisa memulai mensimulasikan model nya.

```
dim(train.d)
[1] 4100    12
dim(test.d)
[1] 1024    12
```

### b. Model Simulation

Model yang akan disimulasikan dalam projek ini adalah GLM atau Generalized Linear Model. GLM adalah generalisasi yang fleksibel dari regresi linier biasa. GLM menggeneralisasikan regresi linier dengan membiarkan model linier dikaitkan dengan suatu dependent variable dan membiarkan besarnya varian dari setiap independent variable yang menjadi fungsi dari nilai prediksi hasilnya.

GLM Family yang disediakan oleh Rstudio adalah:

- Binomial
- Gaussian
- Gamma
- Inverse Gaussian
- Poisson
- Quasi

Kita akan menggunakan family Gaussian pada projek ini dengan data yang disimulasikan adalah data train.d. Kita gunakan family Gaussian karena variable yang dihasilkan bersifat continuous.

```
set.seed(29)
glm.mod = glm(log(Price)~., family = "gaussian",
data = train.d)
```

```

summary(glm.mod)

Call:
glm(formula = log(Price) ~ ., family = "gaussian", data = train.d)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-2.31967 -0.13050 -0.00551  0.12510  0.89095 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.331e+01  3.823e-02 348.176 < 2e-16 ***
ID           7.560e-06  2.731e-06  2.769 0.005656 **  
Rooms        1.238e-01  5.470e-03 22.623 < 2e-16 ***  
Type2        -9.895e-02  1.397e-02 -7.081 1.68e-12 ***  
Type3        -3.262e-01  1.375e-02 -23.718 < 2e-16 ***  
Distance     -2.818e-02  1.625e-03 -17.342 < 2e-16 ***  
Car2         -9.651e-02  2.234e-02 -4.320 1.60e-05 ***  
Car3         -9.022e-02  1.813e-02 -4.975 6.78e-07 ***  
Car4         -2.138e-02  1.727e-02 -1.238 0.215658    
Car5          8.311e-03  2.231e-02  0.372 0.709584    
Landsize     -4.511e-06  3.881e-06 -1.162 0.245208    
BuildingArea 7.865e-04  4.751e-05 16.557 < 2e-16 ***  
CouncilArea2 4.878e-01  3.783e-02 12.896 < 2e-16 ***  
CouncilArea3 3.818e-01  3.418e-02 11.170 < 2e-16 ***  
CouncilArea4 -2.849e-01  4.922e-02 -5.788 7.68e-09 ***  
CouncilArea5 3.265e-01  1.541e-01 2.118 0.034207 *  
CouncilArea6 1.761e-01  8.686e-02 2.028 0.042641 *  
CouncilArea7 -3.493e-02  3.318e-02 -1.053 0.292558    
CouncilArea8 3.840e-01  7.344e-02 5.229 1.79e-07 ***  
CouncilArea9 2.348e-01  3.531e-02 6.649 3.35e-11 ***  
CouncilArea10 1.407e-01  7.132e-02 1.973 0.048527 *  
CouncilArea11 -4.406e-02  5.002e-02 -0.881 0.378499    
CouncilArea12 -2.226e-01  4.084e-02 -5.450 5.34e-08 ***  
CouncilArea13 2.799e-01  4.412e-02 6.345 2.47e-10 ***  
CouncilArea14 1.705e-01  4.664e-02 3.655 0.000260 ***  
CouncilArea15 4.999e-01  1.560e-01 3.204 0.001368 **  
CouncilArea16 1.133e-01  2.780e-02 4.076 4.67e-05 ***  
CouncilArea17 -2.022e-01  4.959e-02 -4.078 4.63e-05 ***  
CouncilArea18 1.816e-01  4.951e-02 3.669 0.000247 ***  
CouncilArea19 8.149e-02  3.808e-02 2.140 0.032436 *  
CouncilArea20 -3.291e-01  6.288e-02 -5.233 1.75e-07 ***  
CouncilArea21 2.403e-01  3.444e-02 6.979 3.45e-12 ***  
CouncilArea22 -6.457e-02  4.745e-02 -1.361 0.173624 ... 

            Estimate Std. Error t value Pr(>|t|)    
CouncilArea22 -6.457e-02  4.745e-02 -1.361 0.173624  
CouncilArea23 -9.368e-02  3.232e-02 -2.899 0.003766 **  
CouncilArea24  1.026e-01  6.419e-02  1.598 0.110057  
CouncilArea25  2.724e-01  3.789e-02  7.190 7.69e-13 ***  
CouncilArea26  3.252e-01  3.739e-02  8.697 < 2e-16 ***  
CouncilArea27  2.426e-01  2.965e-02  8.183 3.66e-16 ***  
CouncilArea28 -1.115e-01  4.105e-02 -2.717 0.006609 **  
CouncilArea29 -4.165e-01  5.745e-02 -7.249 4.99e-13 ***  
CouncilArea30  6.781e-02  3.761e-02  1.803 0.071476 .  
CouncilArea31  1.972e-01  1.136e-01  1.736 0.082627 .  
Regionname2   2.439e-02  9.504e-02  0.257 0.797446    
Regionname3   -4.872e-02  3.229e-02 -1.509 0.131345  
Regionname4   1.183e-01  8.335e-02  1.419 0.155860  
Regionname5   -3.628e-02  4.014e-02 -0.904 0.366180  
Regionname6   -7.437e-02  2.915e-02 -2.552 0.010757 *  
Regionname7   3.210e-02  4.767e-02  0.673 0.500800  
Regionname8   2.227e-02  9.799e-02  0.227 0.820195  
EffAge        1.383e-03  1.423e-04  9.714 < 2e-16 ***  
cluster       1.479e-01  4.978e-03 29.714 < 2e-16 *** 

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.05117369)

Null deviance: 998.36 on 4099 degrees of freedom
Residual deviance: 207.20 on 4049 degrees of freedom
AIC: -497.78

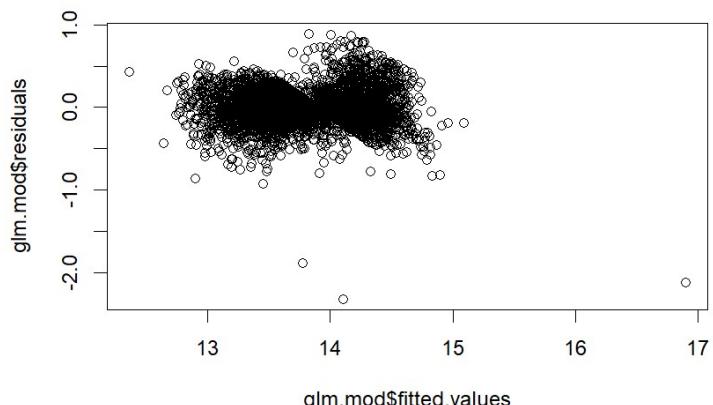
Number of Fisher Scoring iterations: 2

```

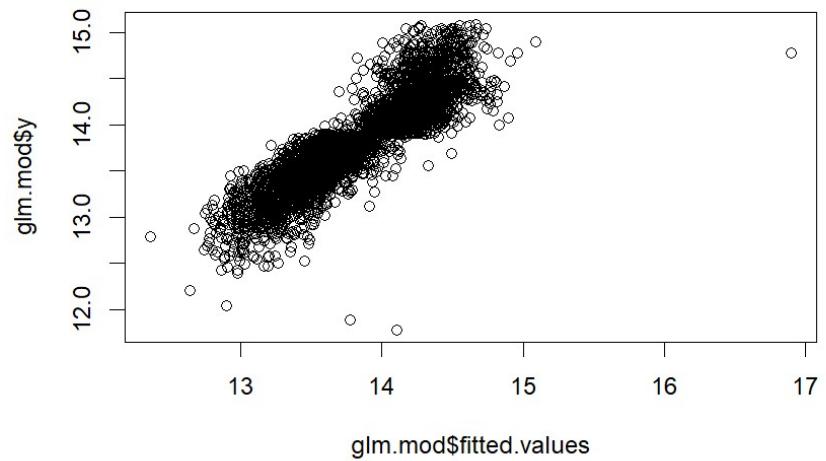
Dari glm.mod, didapatkan nilai AIC -497.78 yang akan digunakan untuk dibandingkan dengan model satunya dengan data train.d yang sudah dihapus outlier nya.

Berikut adalah plot untuk membantu visualisasi hasil glm.mod yang sudah di dapatkan.

```
plot(glm.mod$fitted.values, glm.mod$residuals)
```

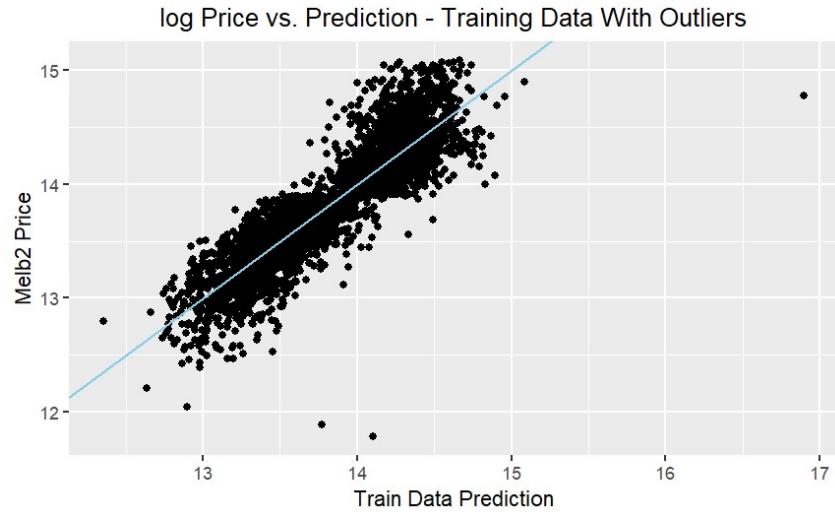


```
plot(glm.mod$fitted.values, glm.mod$y)
```



Berikutnya ini juga merupakan plot visualisasi glm.mod dengan garis abline yang merepresentasikan nilai aslinya.

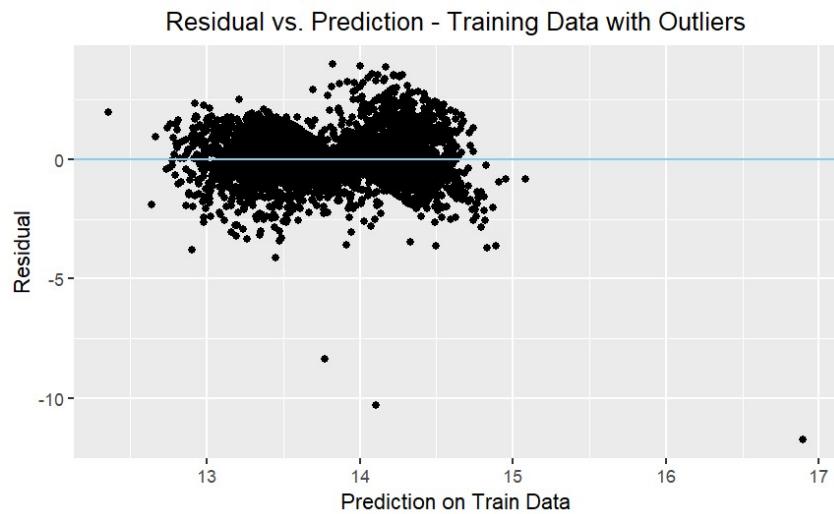
```
ggplot() +  
  geom_point(aes(x = glm.mod$fitted.values, y =  
log(train.d$Price))) +  
  geom_abline(aes(intercept = 0, slope = 1), colour =  
"sky blue") +  
  ggtitle("log Price vs. Prediction - Training Data  
With Outliers") +  
  theme(plot.title = element_text(hjust = 0.5)) +  
  labs(x = "Train Data Prediction", y = "Melb2  
Price")
```



```

residu = data.frame(x = rstandard(glm.mod))
pred = glm.mod$fitted.values
ggplot() +
  geom_point(aes(x = pred, y = residu$x)) +
  geom_abline(aes(intercept = 0, slope = 0), colour =
"sky blue") +
  ggtitle("Residual vs. Prediction - Training Data
with Outliers") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = "Prediction on Train Data", y =
"Residual")

```



Dari plot yang sudah disimulasikan, dapat dilihat bahwa data tersebar pada garis  $y = x$  yang artinya bahwa prediksi dari data ini sudah cukup mendekati atau mirip dengan nilai aslinya. Meskipun begitu, tetap dapat dilihat bahwa terdapat beberapa prediksi yang salah. Hal ini dapat dilihat dari titik-titik yang berada jauh dari garis atau kumpulan data di sekitarnya. Titik-titik ini atau residual ini dapat dilihat di kedua plot, tetapi lebih terlihat pada plot 2.

```

bin1 = which(abs(residu) > 3)
if(length(bin1)>0){
    train.outliers1 = train.d
    train.outliers1$outliers = 0
    train.outliers1$outliers[bin1] = 1
    train.outliers1$pred = glm.mod$fitted.values
    train.outliers1$pred.dollar =
        exp(train.outliers1$pred)
    train.d2 = train.d[-bin1,]
} else{
    train.d2 = train.d
}

```

Variable bin1 dan function which() digunakan untuk menghapus residu yang memiliki nilai lebih dari 3 dan outliers dari model glm yang sudah disimulasikan. Hasil yang tersisa akan dimasukan ke dalam data train.d2 untuk disimulasikan kembali dengan glm.

```

glm.modf = glm(log(Price)~., family = "gaussian",
data = train.d2)
summary(glm.modf)

Call:
glm(formula = log(Price) ~ ., family = "gaussian", data =
train.d2)

Deviance Residuals:
Min      1Q      Median       3Q      Max
-0.96645 -0.12397 -0.00225   0.12226   0.66626

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.328e+01 3.493e-02 380.195 < 2e-16 ***
ID          8.553e-06 2.485e-06  3.434 0.00060 ***
Rooms       1.074e-01 5.281e-03 20.338 < 2e-16 ***
Type2       -8.253e-02 1.273e-02 -6.483 1.01e-10 ***
Type3       -3.068e-01 1.256e-02 -24.436 < 2e-16 ***
Distance    -2.793e-02 1.477e-03 -18.907 < 2e-16 ***
Car2        -9.664e-02 2.050e-02 -4.714 2.51e-06 ***
Car3        -8.875e-02 1.665e-02 -5.330 1.04e-07 ***
Car4        -2.850e-02 1.587e-02 -1.795 0.07266 .
Car5        1.471e-03 0.039e-02  0.072 0.94251
Landsize     -5.254e-06 3.541e-06 -1.484 0.13792
BuildingArea 1.245e-03 5.588e-05 22.283 < 2e-16 ***
CouncilArea2 4.919e-01 3.443e-02 14.285 < 2e-16 ***
CouncilArea3 3.892e-01 3.112e-02 12.505 < 2e-16 ***
CouncilArea4 -2.760e-01 4.466e-02 -6.180 7.04e-10 ***
CouncilArea5 2.839e-01 1.397e-01  2.033 0.04216 *
CouncilArea6 1.836e-01 7.873e-02  2.332 0.01974 *
CouncilArea7 -3.844e-02 3.017e-02 -1.274 0.20265
CouncilArea8 3.926e-01 6.662e-02  5.892 4.12e-09 ***
CouncilArea9 2.561e-01 3.214e-02  7.967 2.10e-15 ***
CouncilArea10 1.521e-01 6.466e-02  2.352 0.01874 *
CouncilArea11 -4.672e-02 4.542e-02 -1.029 0.30372
CouncilArea12 -2.225e-01 3.704e-02 -6.006 2.07e-09 ***
CouncilArea13 2.877e-01 4.007e-02  7.180 8.24e-13 ***
CouncilArea14 1.700e-01 4.232e-02  4.016 6.02e-05 ***
CouncilArea15 4.697e-01 1.414e-01  3.323 0.00090 ***
CouncilArea16 1.185e-01 2.527e-02  4.689 2.84e-06 ***
CouncilArea17 -1.917e-01 4.501e-02 -4.260 2.09e-05 ***
CouncilArea18 1.949e-01 4.490e-02  4.342 1.45e-05 ***
CouncilArea19 6.542e-02 3.478e-02  1.881 0.06001 .
CouncilArea20 -3.409e-01 5.698e-02 -5.983 2.38e-09 ***
CouncilArea21 2.568e-01 3.131e-02  8.201 3.18e-16 ***
CouncilArea22 -6.687e-02 4.305e-02 -1.553 0.12046

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be
0.04196627)

Null deviance: 947.69 on 4058 degrees of freedom
Residual deviance: 168.20 on 4008 degrees of freedom
AIC: -1299

Number of Fisher Scoring iterations: 2

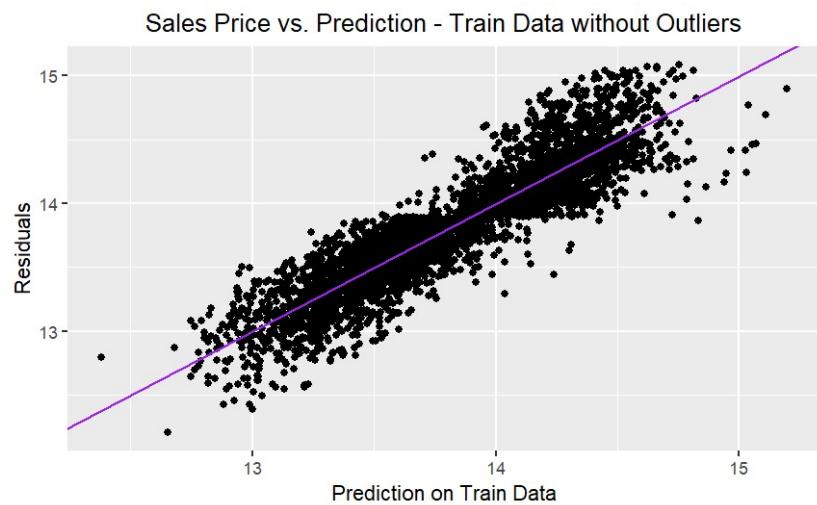
```

Dengan train.d2 yang sudah tidak ada outliers, didapatkan model glm final dengan hasil AIC = -1299. Karena hasil AIC ini lebih kecil dibandingkan model yang pertama, maka kita akan memilih model ini. AIC yang lebih kecil juga menandakan bahwa model sudah lebih akurat.

```
residuf = data.frame(x = rstandard(glm.modf))
predf = glm.modf$fitted.values
```

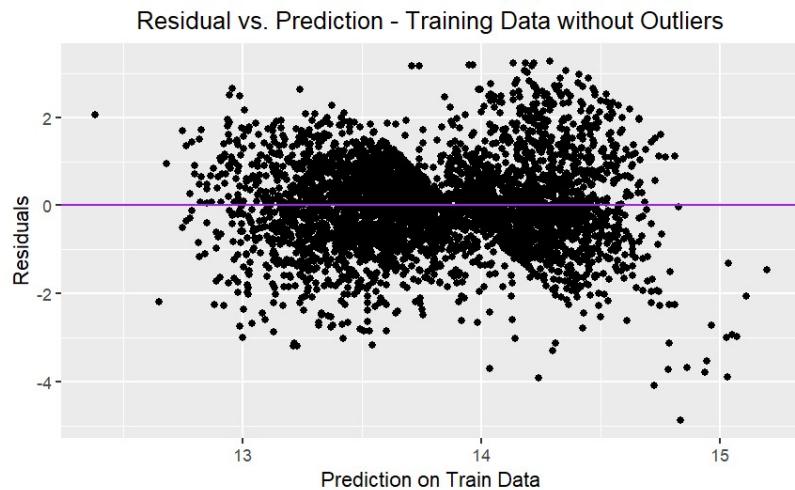
Dari model yang sudah dihasilkan, akan dikumpulkan lagi residu yang masih tersisa dengan function data.frame() dan disimpan dalam residuf. Akan disimulasikan lagi plot dengan nilai fitted.values dari model glm final dengan variable Price dari train.d2.

```
ggplot() +
  geom_point(aes(x = predf, y = log(train.d2$Price)))
  +
  geom_abline(aes(intercept = 0, slope = 1), colour =
  "purple") +
  ggtitle("Sales Price vs. Prediction - Train Data
without Outliers") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = "Prediction on Train Data", y =
  "Residuals")
```



Berikut adalah plot yang memvisualisasikan nilai fitted.values dan model glm final dengan nilai x dari residuf.

```
ggplot() +  
  geom_point(aes(x = predf, y = residuf$x)) +  
  geom_abline(aes(intercept = 0, slope = 0),  
  colour = "purple") +  
  ggtitle("Residual vs. Prediction - Training  
Data without Outliers") +  
  theme(plot.title = element_text(hjust = 0.5)) +  
  labs(x = "Prediction on Train Data", y =  
  "Residuals")
```



Dengan outliers sudah dibersihkan, dapat kita lihat bahwa data menjadi lebih tersebar. Dapat dilihat juga bahwa hasil sekarang lebih mendekati lagi pada garis  $y = x$ . Meskipun begitu, masih terdapat beberapa outliers yang dapat dilihat. Apabila diperhatikan, tidak terdapat pattern tertentu dengan residuals ini, sehingga bisa terbilang bahwa residual ini memiliki distribusi acak.

c. Error and Multi co-linearity test

```
vif(glm.modf)

          GVIF Df GVIF^(1/(2*Df))
ID          1.355367  1      1.164202
Rooms       2.129352  1      1.459230
Type         2.100373  2      1.203855
Distance     6.878347  1      2.622660
Car          1.665254  4      1.065823
Landsize     1.061446  1      1.030265
BuildingArea 1.861401  1      1.364332
CouncilArea  97105.642795 30    1.210935
Regionname   22146.650007 7      2.043521
EffAge       1.814275  1      1.346950
cluster      1.692219  1      1.300853
```

Function vif() digunakan untuk mendeteksi multikolinearitas dari data yang sudah dihasilkan, pada bagian ini yaitu hasil data glm.modf. Nilai GVIF<sup>(1/(2\*Df))</sup> yang kita dapatkan semua berada di bawah nilai 5 atau lebih kecil daripada 5, maka artinya variable yang ada di dalam ini memiliki korelasi yang moderate antara satu sama lainnya.

```
comp = function(pred, obs) {
  n = length(obs)
  rsq = cor(pred, obs)^2
  mse = sum((pred - obs)^2) / n
  semse = sd((pred - obs)^2) / sqrt(n)
  rmse = sqrt(mse)
  se = sd(pred - obs) / sqrt(n)
  mae = sum(abs(pred - obs)) / n
  mape = sum(abs(pred - obs) / obs) / (n*100)
  return(list("n" = n, "R2" = rsq, "MSE" = mse,
             "SEMSE" = semse, "RMSE" = rmse, "SE" = se, "MAE" =
             mae, "MAPE" = mape))
}
```

Comp merupakan kumpulan fungsi yang akan digunakan dalam menguji model untuk mengetahui apakah model sudah akurat atau belum.

```
comp(glm.modf$fitted.values, glm.modf$y)
$n
[1] 4059

$R2
[1] 0.822515

$MSE
[1] 0.04143898

$SEMSE
[1] 0.001089945

$RMSE
[1] 0.2035657

$SE
[1] 0.003195571

$MAE
[1] 0.1560015

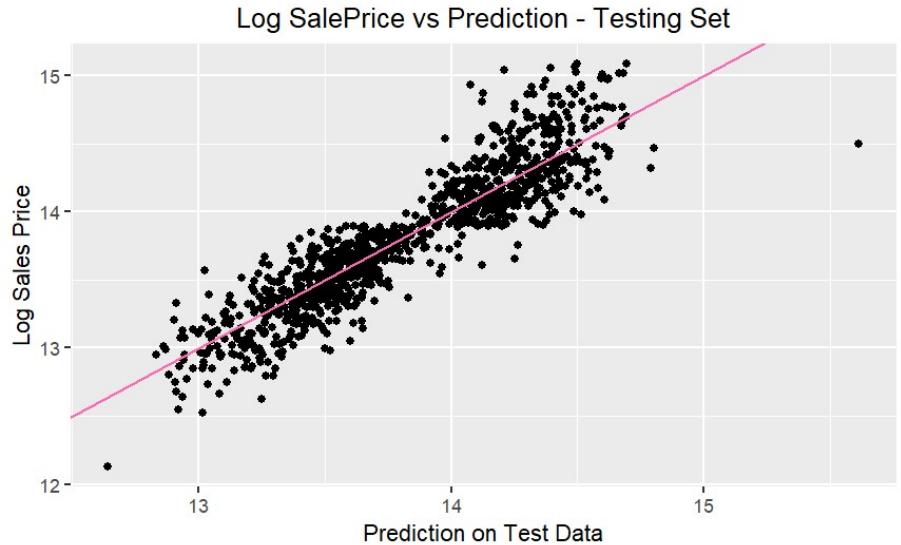
$MAPE
[1] 0.0001129279
```

Fungsi comp digunakan dengan data fitted.values dan y dari glm.modf sehingga menghasilkan nilai-nilai tersebut. Dari 4059 data yang di uji, kita dapatkan nilai MAPE 0.0001129279% dan nilai r-squared sebesar 82.25%. Hasil r-squared sebesar 82.25% berarti bahwa model ini sudah bisa menjelaskan 82.25% dari data yang disimulasikan. Nilai MAPE yang didapatkan berarti bahwa rata-rata perbedaan antara hasil yang diuji dengan nilai aslinya adalah 0.0001129279%.

```

test.d$prediction = predict(glm.modf, newdata =
test.d, type = "response")
ggplot() +
  geom_point(aes(x = test.d$prediction, y =
log(test.d$Price))) +
  geom_abline(aes(intercept = 0, slope = 1),
colour = "hot pink") +
  ggtitle("Log SalePrice vs Prediction - Testing
Set") +
  theme(plot.title = element_text(hjust = 0.5))
+
  labs(x = "Prediction on Test Data", y ="Log
Sales Price")

```



Berikut adalah plot prediksi test data dengan nilai aslinya. Dapat dilihat juga bahwa banyak data yang tersebar pada garis  $y = x$ , yang artinya banyak nilai prediksi yang sudah mendekati dengan nilai aslinya. Tetapi masih terdapat juga beberapa residual yang menandakan bahwa terdapat prediksi yang salah.

```
comp(test.d$prediction, test.d$Price)
$n
[1] 1024

$R2
[1] 0.7079945

$MSE
[1] 1.67253e+12

$SEMSE
[1] 61289823332

$RMSE
[1] 1293263

$SE
[1] 19171.99

$MAE
[1] 1138644

$MAPE
[1] 0.009999845
```

Fungsi comp digunakan dengan data test.d\$prediction dan test.d\$Price sehingga menghasilkan nilai-nilai tersebut. Dari 1024 data yang di uji, kita dapatkan nilai MAPE 0.9999845% dan nilai r-squared sebear 70.8%. Hasil r-squared sebesar 70.8% berarti bahwa model ini sudah bisa menjelaskan 70.8% dari test data. Nilai MAPE yang didapatkan berarti bahwa rata-rata perbedaan antara prediksi test data dengan nilai aslinya adalah 0.9999845%.

## 5. Tree Based Model

*Tree Based Model* adalah metode berbasis pohon keputusan untuk mewakili bagaimana suatu variable input dapat digunakan untuk memprediksi suatu nilai target. Di projek ini, tersedia dua pilihan untuk mensimulasikan *tree based model* yaitu dengan randomForest atau dengan Gradient Boosting Machine (GBM).

### a. Split Data Training and Data Testing

Sebelum memulai untuk mensimulasikan modelnya, yang pertama kali harus dilakukan adalah membagi data menjadi data training dan data testing. Pembagian akan menggunakan function `createFolds()` yang akan membagi data ke dalam 10 folds untuk masing-masing test data dan train data. Kita gunakan function `set.seed()` agar dapat menghasilkan hasil yang sama setiap kali di run.

```
set.seed(127)
n.folds = 10

melb2$folds = createFolds(melb2$Price, k = n.folds,
list = FALSE, returnTrain = FALSE)

test.sd = melb2[melb2$folds == 10, ]
train.val.sd = melb2[melb2$folds != 10, ]
```

Dengan kata lain, disini data akan dibagi menjadi 10 folds seperti misal fold 10 akan menjadi test data atau `test.sd` dan folds 1 hingga 9 akan menjadi train data atau `train.val.sd`. Selanjutnya, akan di cek terlebih dahulu apabila di dalam test data atau train data terdapat nilai NA dengan function `is.na()`. Apabila iya, function `na.omit()` akan mengeluarkan nilai tersebut dan memasukkan yang tidak kembali ke dalam datanya.

```
table(is.na(train.val.sd))
FALSE
59969
train.val.sd = na.omit(train.val.sd)
table(is.na(test.sd))
FALSE
6643
test.sd = na.omit(test.sd)
```

```

train.val.sd$folds2 =
createFolds(train.val.sd$Price, k = n.folds, list =
FALSE, returnTrain = FALSE)

```

Kemudian kita bagi lagi train data kita ke dalam 10 folds dengan function createFolds() dengan jumlah folds k. Folds ini akan digunakan untuk mensimulasikan cross validation dilakukan untuk mengevaluasi model yang akan disimulasikan dengan sampel data yang terbatas.

```



```

Sama dengan sebelumnya, akan dicek terlebih dahulu apabila folds yang baru saja dibuat terdapat nilai NA atau tidak dengan function is.na(). Apabila iya, maka function na.omit() akan mengeluarkannya dan memasukkan yang bukan bernilai NA kembali ke dalam datanya.

#### b. Model Simulation

Model yang akan disimulasikan dalam projek ini adalah model random forest. Random forest adalah suatu metode yang digunakan untuk menyelesaikan masalah klasifikasi dan regresi dengan membangun pohon keputusan pada sampel yang berbeda dan mengambil suara mayoritas untuk melakukan klasifikasi dan rata-rata. Kita akan melakukan simulasi pada train data menggunakan function randomForest().

Pertama, kita akan deklarasi kumpulan function comp() lagi yang akan digunakan di dalam cross validation random forest dan saat mengevaluasi model menggunakan test data.

```

comp = function(pred, obs) {
  n = length(obs)
  rsq = cor(pred,obs)^2
  mse = sum((pred - obs)^2)/n
  semse = sd((pred - obs)^2) / sqrt(n)
  rmse = sqrt(mse)
  se = sd(pred-obs) / sqrt(n)
  mae = sum(abs(pred-obs))/n
  mape = sum(abs(pred-obs)/obs)/n*100
}

```

```
return(list("n"=n, "R2"=rsq, "MSE"=mse, "SEMSE"=semse, "RMSE"=rmse, "SE"=se, "MAE"=mae, "MAPE"=mape))
```

```
ntree = c(1, 3, 5, 7, 10, 15, 20)
mtry = c(11/3)
```

Di sini telah kita tetapkan jumlah ntree yang akan digunakan dengan function c() yang berisi nilai dari 1 hingga 20 untuk dicoba. Selain itu, nilai mtry sudah ditetapkan sesuai dengan ketentuan projek yaitu dengan  $\frac{p}{3}$ , dimana p adalah jumlah independent variable yang digunakan dalam model ini.

Total variable dalam melb2 adalah 13, dengan Price sebagai dependent variable dan terdapat variable folds yang akan digunakan dalam cross validation. Maka dari situ, menyisakan 11 independent variable yang digunakan sehingga didapatkan mtry = 11/3.

```

MAPE = NULL
MAPE.ave = matrix(, nrow = length(ntree), ncol =
length(mtry))
rownames(MAPE.ave) = ntree
colnames(MAPE.ave) = mtry

for(j in 1:length(ntree)){
  t = ntree[j]
  for(k in 1:length(mtry)){
    m = mtry[k]
    for(i in 1:10){
      train.set = train.val.sd[train.val.sd$folds2
!= i, ]
      val.sd = train.val.sd[train.val.sd$folds2 ==
i, ]

      rf = randomForest(formula = Price~., data =
train.set, mtry = m, ntree = t)

      val.sd$pred = predict(rf, val.sd)
      MAPE[i] = comp(val.sd$pred, val.sd$Price)$MAPE
    }
    MAPE.ave[j, k] = mean(MAPE)
  }
}
MAPE.ave

```

	3.66666666666667
1	0.001716829
3	0.001376642
5	0.001316753
7	0.001277943
10	0.001250362
15	0.001218963
20	0.001204629

```

opt.ntree = ntree[which.min(MAPE.ave)]
opt.ntree
[1] 20

```

Model simulasi dilakukan dengan menggunakan function randomForest() dilengkapi dengan fungsi for() yang berguna untuk looping dalam cross validation model ini. Hasil yang didapatkan berbentuk kumpulan nilai MAPE dari ntree yang telah disimulasikan.

Untuk mencari nilai optimum untuk jumlah pohon maka digunakan function ntree[which.min(MAPE.ave)] yang akan disimpan di dalam variable opt.ntree. Hasil yang didapatkan adalah 20 dengan nilai MAPE yang terkecil.

Dengan nilai optimum ntree yang sudah ditemukan, maka akan disimulasikan random forest lagi dengan train data dan mtry 11/3.

```
rf.final = randomForest(formula = Price~., data =
train.val.sd, mtry = 11/3, ntree = opt.ntree)
Call:
randomForest(formula = Price ~ ., data =
train.val.sd, mtry = 11/3,      ntree = opt.ntree)
Type of random forest: regression
Number of trees: 20
No. of variables tried at each split: 4

Mean of squared residuals: 35900459596
% Var explained: 89.63
```

```
test.sd$pred = predict(rf.final, newdata = test.sd)
comp(test.sd$pred, test.sd$Price)

$n
[1] 511

$R2
[1] 0.8953498

$MSE
[1] 35623032831

$SEMSE
[1] 4256241339

$RMSE
[1] 188740.6

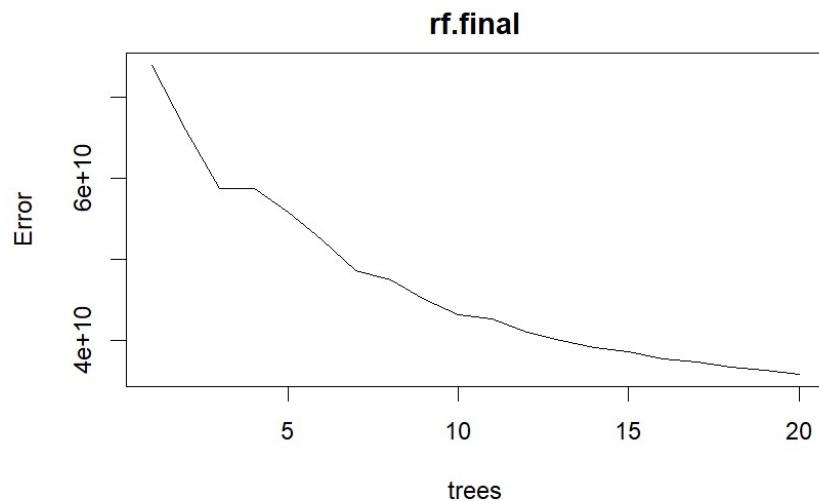
$SE
[1] 8322.448

$MAE
[1] 133432.3

$MAPE
[1] 0.001377825
```

Fungsi comp digunakan dengan data prediksi test.sd dan variable Price dari test.sd sehingga menghasilkan nilai-nilai tersebut. Dari 511 data yang di uji, kita dapatkan nilai MAPE 0.1377825% dan nilai r-squared sebear 89.53%. Hasil r-squared sebesar 89.53% berarti bahwa model ini sudah bisa menjelaskan 89.53% dari data yang disimulasikan. Nilai MAPE yang didapatkan berarti bahwa rata-rata perbedaan antara prediksi test.sd dengan nilai aslinya adalah 0.1377825%.

```
plot(rf.final)
```



Setelah menemukan model random forest final dengan jumlah optimum untuk ntree, akan diuji dengan data test. Menggunakan fungsi predict() akan di uji hasil rf.final dengan data test kemudian disimpan di dalam test.sd\$pred. Kemudian variable pred itu akan diuji kembali dengan variable Price dari test data menggunakan function comp(). Plot menunjukkan bahwa nilai error semakin kecil dengan membesarnya trees yang disimulasikan.

## 6. Kesimpulan

Pada projek ini telah disimulasikan clustering, linear based model/ additive model dan tree based model. Clustering adalah metode pengelompokkan data abstrak ke dalam kelompok dengan data-data yang sifatnya mirip. Pada clustering projek ini, diterapkan nstart = 100 untuk dibagi ke dalam 3 cluster. Saat mencari menggunakan metode elbow, ditemukan bahwa mulai 3 cluster adalah pilihan yang optimum untuk data yang dimiliki.

Data yang digunakan sebagai dependent variable adalah Price dengan independent variable yang melengkapinya adalah Type, Car, CouncilArea, Regionname, ID, Rooms, Distance, Landsize, BuildingArea, dan EffAge ditambah lagi dengan hasil dari clustering. Sehingga dengan ini, terdapat 11 variable independent.

Linear based model/additive model adalah salah satu metode dalam regresi yang dilakukan untuk memvisualisasikan variable respon atau dependent variable dalam hal kombinasi linier variable prediktor. Projek ini menggunakan GLM untuk mensimulasikan data dengan training data dan testing data yang dibagi dalam rasio 80:20. Setelah simulasi, ditemukan bahwa model final memiliki nilai AIC yang lebih rendah sehingga model tersebutlah yang dipilih.

Tree based model adalah metode berbasis pohon keputusan untuk mewakili bagaimana suatu variable input dapat digunakan untuk memprediksi suatu nilai target. Metode yang dipilih adalah random forest. Data dibagi menjadi dalam data testing dan training dengan 10 folds, yang kemudian data training dibagi lagi ke dalam 10 folds untuk kepentingan cross validation. Ditemukan bahwa ntree yang optimum adalah 20 dengan mtry = 11/3.

Tujuan utama projek ini adalah untuk mencari model manakah yang lebih cocok untuk data melb2. Setelah observasi dan melakukan simulasi, didapatkan kesimpulan bahwa tree based model yaitu random forest lebih cocok untuk mensimulasikan data ini. Hal ini dapat dilihat dari hasil comp pada kedua model yang membandingkan prediksi test data dengan nilai aslinya. Nilai MAPE yang dihasilkan oleh random forest memiliki nilai yang lebih rendah sebesar 0.1377825% dibandingkan yang dihasilkan oleh GLM yaitu 0.9999845%. Selain itu juga, R-squared yang dihasilkan oleh random forest memiliki nilai yang lebih tinggi yaitu 89.35% dibandingkan yang dihasilkan oleh GLM yaitu 70.8%. Sehingga, dengan ini kesimpulannya adalah tree based model dengan metode random forest merupakan model yang lebih cocok untuk memvisualisasikan data ini.