機器學習與商業應用

期中報告_RT 組

Melbourne Housing Price Prediction

指導老師： 周雨田

組員 ：莊高閔、陳政廷

民國１１０年６月16日

目錄

一、動機..2

二、資料結構..2

三、資料前處理..6

1.缺失值處理 6

2.類別欄位處理 7

3.刪除共線性高的變數 8

四、模型選用..8

五、期中後修正..9

1.預期修改內容 9

2.過程疑難排解 9

六、結論..10

1.結果比較 10

2.研究發現 11

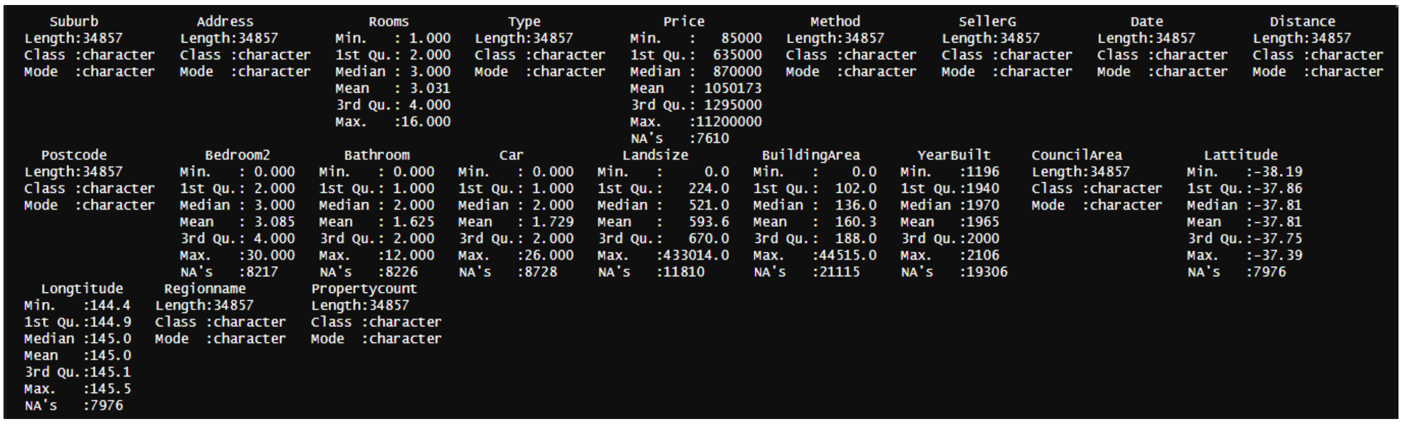
附錄 :R 程式碼...12

一、動機

房屋這項議題始終與人類的生活息息相關， 如何找到符合自身條件的物件更為重要，其中房價便是一項不可不考慮的因素之一，其取決了人們今天是否要買地蓋房、脫手物件、進場投資...等重大抉擇，但今天影響房價的因子眾多，且市面上房價沒有訂定明確的價格規範，導致房價高低變化大，如此一來不管是以房東還是房客的角度出發都會遇到買高賣低的問題，故我們這次打算在這塊複雜的領域中預測出相對準確的房屋價格，希望助於大家在後續進行買房賣房等重大抉擇時能透過資料科學的角度，多一種評估方式，避免被知情人士套利，談定更為滿意的價格。

二、資料結構

此資料集收集了從 2016~2018 年三年間的房屋交易相關資料，總共 34857 筆。



```
    Suburb             Address              Rooms             Type               Price              Method             SellerG             Date              Distance
 Length:34857       Length:34857       Min.   : 1.000   Length:34857       Min.   :   85000   Length:34857       Length:34857       Length:34857       Length:34857
 Class :character   Class :character   1st Qu.: 2.000   Class :character   1st Qu.:  635000   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Median : 3.000   Mode  :character   Median :  870000   Mode  :character   Mode  :character   Mode  :character   Mode  :character
                                       Mean   : 3.031                      Mean   : 1050173
                                       3rd Qu.: 4.000                      3rd Qu.: 1295000
                                       Max.   :16.000                      Max.   :11200000
                                                                           NA's   :7610
   Postcode           Bedroom2          Bathroom            Car            Landsize          BuildingArea        YearBuilt     CouncilArea          Lattitude
 Length:34857       Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :     0.0   Min.   :    0.0   Min.   :1196   Length:34857       Min.   :-38.19
 Class :character   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:   224.0   1st Qu.:  102.0   1st Qu.:1940   Class :character   1st Qu.:-37.86
 Mode  :character   Median : 3.000   Median : 2.000   Median : 2.000   Median :   521.0   Median :  136.0   Median :1970   Mode  :character   Median :-37.81
                    Mean   : 3.085   Mean   : 1.625   Mean   : 1.729   Mean   :   593.6   Mean   :  160.3   Mean   :1965                      Mean   :-37.81
                    3rd Qu.: 4.000   3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.:   670.0   3rd Qu.:  188.0   3rd Qu.:2000                      3rd Qu.:-37.75
                    Max.   :30.000   Max.   :12.000   Max.   :26.000   Max.   :433014.0   Max.   :44515.0   Max.   :2106                      Max.   :-37.39
                    NA's   :8217     NA's   :8226     NA's   :8728     NA's   :11810      NA's   :21115     NA's   :19306                     NA's   :7976
   Longtitude      Regionname         Propertycount
 Min.   :144.4   Length:34857       Length:34857
 1st Qu.:144.9   Class :character   Class :character
 Median :145.0   Mode  :character   Mode  :character
 Mean   :145.0
 3rd Qu.:145.1
 Max.   :145.5
 NA's   :7976
```

利用 summary 我們可以分別看到基礎的資料樣態，可以發現本資料集以 character 及 numeric 占大宗，其中最為重要的問題也是資料普遍充斥著 NA，由於此資料的取材自現實，故狀況無法盡善盡美，此時，資料前處理就顯得十分重要，

預測模型通常只是最後根據資料結構或目的做不同取用，且方法就算再複雜也可以透過網路上參考或自學知曉，但資料前處理卻不是這麼一回事，需要對房屋的背景有足夠的認知並審慎的針對缺值進行排除與填補，而非隨意將空值補 0 或放上整體平均數便可解決，資料前處理的好壞將會在後續建立模型時放大，錯誤或太過隨便的處理會導致後續建立模型時，在損失函數上有較大的結果，造成使用者誤判模型選用的問題，因此，這也是選取本篇作為學習的目的之一，建立對房屋資料的了解並建立適當的模型。在了解資料樣態後，我們可以接著查看資料欄位中的內容與意義為何：

Suburb：該房屋所處區域

Address：地址

Rooms：房間數

Price：以澳幣計算的價格

Method：房屋出售的方式(預售、法拍...)

Type：房屋類型(套房、雅房...)

SellerG：房屋仲介

Date：成交日期

Distance：距離市中心距離

Regionname：廣泛的地理位置(在澳洲的東西南北)

Propertycount：於該區中所擁有務業數量

Bedroom2：當前尚可使用的房間

Bathroom：廁所數量

Car：汽車車位數 W

Landsize：土地面積(平方公尺)

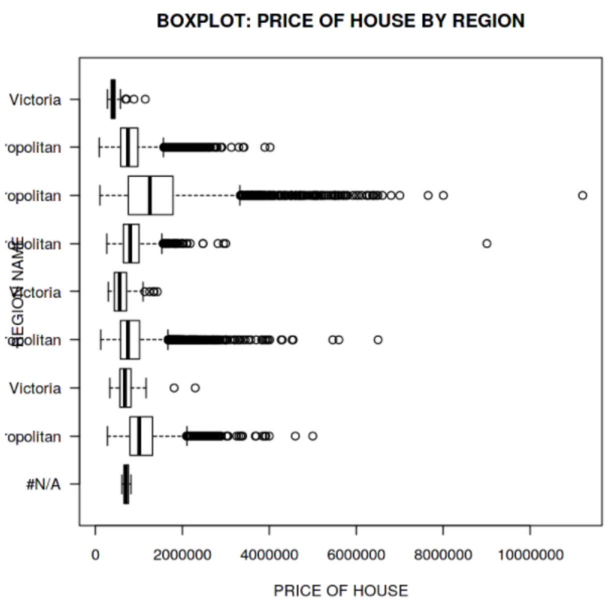
BuildingArea：房屋面積(平方公尺)

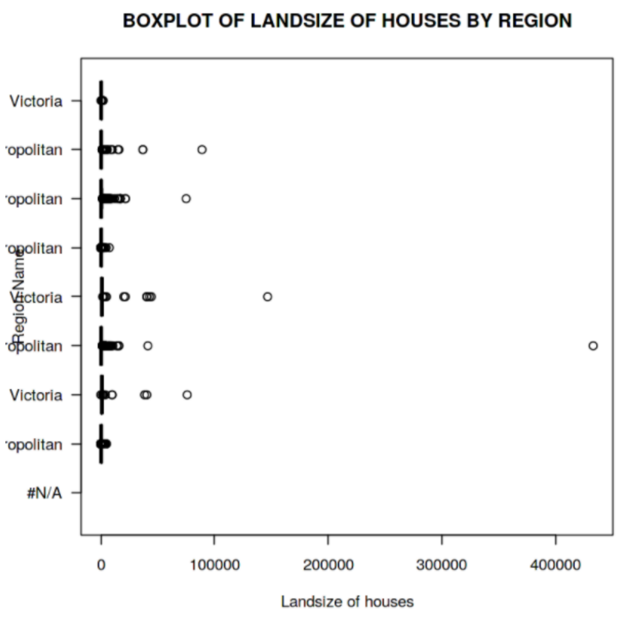
YearBuilt：建造日期

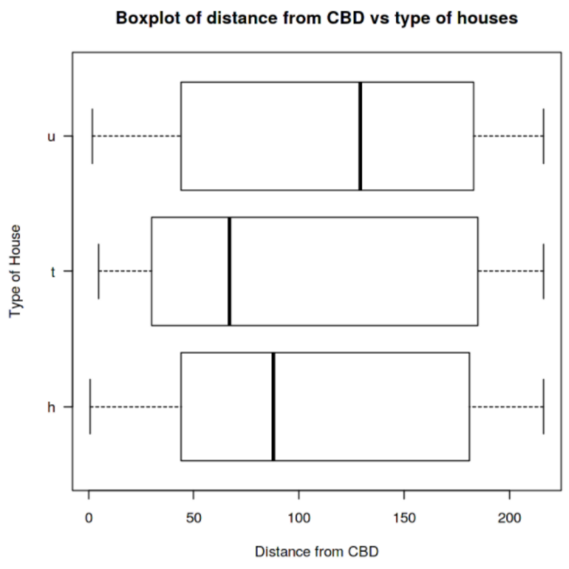
CouncilArea：該地區的市議會

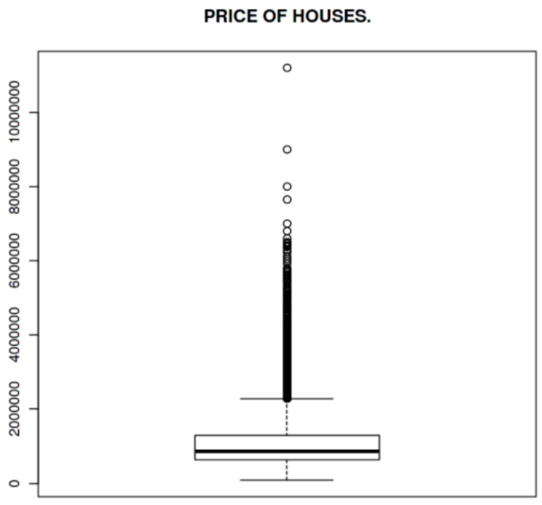
Lattitude：緯度

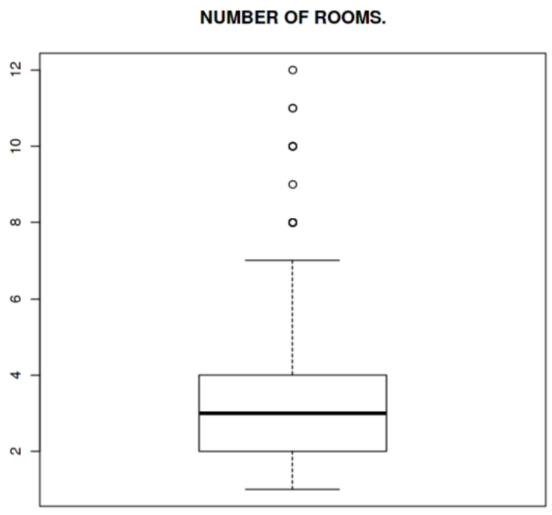
Longtitude：經度

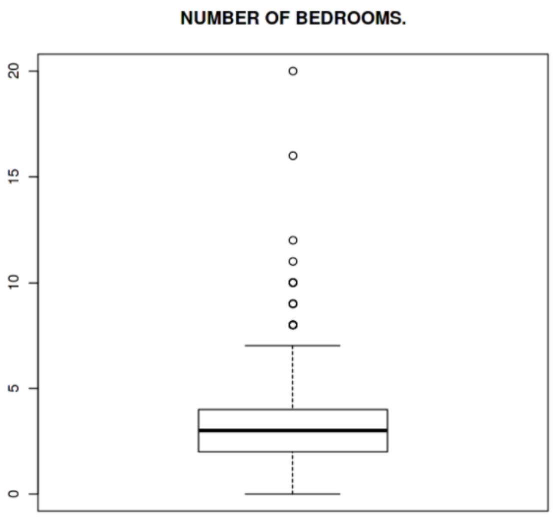


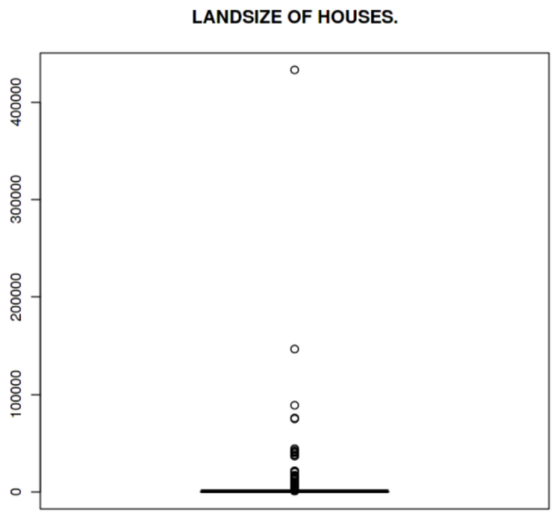



Boxplot of distance from CBD vs type of houses
Type of House
u
t
h
0
50
100
150
200
Distance from CBD


PRICE OF HOUSES.
0
2000000
4000000
6000000
8000000
10000000


NUMBER OF ROOMS.
2
4
6
8
10
12

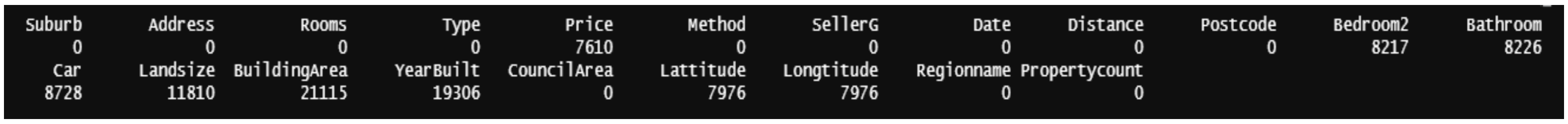

NUMBER OF BEDROOMS.
0
5
10
15
20


LANDSIZE OF HOUSES.
0
100000
200000
300000
400000

三、資料前處理

1.缺失值處理

首先，我們再次針對缺失值進行查看。



```
  Suburb     Address        Rooms       Type        Price      Method      SellerG        Date    Distance     Postcode    Bedroom2    Bathroom
       0           0            0          0         7610           0            0           0           0            0        8217        8226
     Car    Landsize BuildingArea  YearBuilt  CouncilArea   Lattitude   Longtitude  Regionname Propertycount
    8728       11810        21115      19306            0        7976         7976           0           0
```

在此將刪去缺失值佔總比例大於 60%以上的欄位，以防後續進行填補時造成資料失真，盡管不作此步驟在訓練及驗證資料集上表現良好，仍有可能是因為人為行為所導致，使最後真正的測試資料集得到的結果反而是較糟，故我們根據上述原則排除 BuildingArea，接續進行缺失值填補，這邊發現 Bedroom2 及 Rooms 兩欄位有 73%左右的相似度，由此可推可能房東在填寫時並沒有特別將這兩欄位區別出來，故這裡將 Bedroom2 欄位中的缺失值以對應 Rooms 的欄位值做填補。

處理 Landsize 缺失值的方式為透過評估房間數量，通常房間數量越多，該該地面積越大，藉由房間數量對整個資料集分群，並針對不同群當中的缺失值填入該群土地面積的中位數，來達成該欄的缺失值處理。因汽車車位的差距普遍不大，故 Car 欄位的缺失值則以整個資料集的中位數進行填補。至於 Bathrooms 的缺失值則根據此網站所提供的最適房間數目與廁所數目比例進行處理，該網站中定義最佳的比例為 3:2，也就是 3 間房間對應 2 間廁所，這邊

利用上述已完成缺失值填補的 Bedroom2 欄位為對照，替 Bathroom2 進行缺失值填補。

2.類別欄位處理

雖然其餘文字欄位較沒有缺失值的問題發生，但我們也無法直接利用，需要進行欄位上的轉換及 one hot encoding(建立屬於該條件為 1，不屬於該條件為 0 的虛擬變數欄位)的方式處理，才可進行後續建模的動作，下列呈現欄位的轉換方式，完成轉換後均會使用 one hot encoding 的技巧，且最後均會排除其中一個分類以避免虛擬變數陷阱產生。

a. Regionname： 針對欄位內八個地區編列 1~8 ，了解房價是否會因為地區不同而有高低之差。

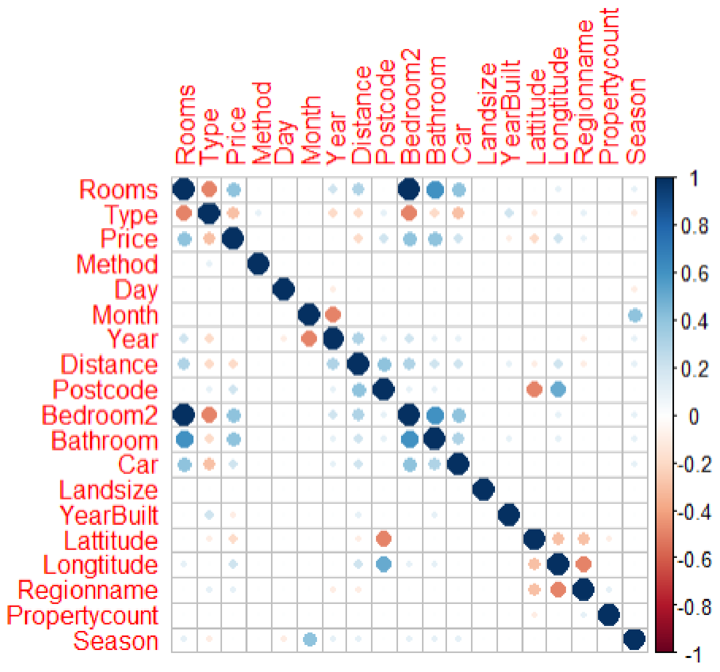
b. Method： 針對該欄提到的九種銷售方式編列 1~9，查看不同的銷售方式是否會導致房價有別。

c. Type： 將不同的房屋類型轉換為數字 1~3，控制不同的房型對於房價的影響。

d. Date： 將交易日期的年月日拆開並定義春夏秋冬四個季節，探討房價是否會因季節而有所起伏。

e. Council Area： 對一共 33 個市議進行編列，瞭解不同地區的議會會如何反映在房價上。

3.刪除共線性高的變數



由上圖可以發現，Bedroom2 與 Rooms 會具有共線性的問題。因此，我們將 Rooms 刪除，留下 Bedroom2。

四、模型選用

完成資料前處理後，我們便可套用模型進行預測，在此預測的對象為 Price，故將會以回歸類型的模型為主要考量，以 Linear Regression 作為 baseline，接續使用 Lasso Regression、Random Forest 和廣為使用的 XGBoost 來建立模型，並將 MSE 列為損失函數，以在最後比較各模型間的優劣。

五、期中後修正

1.預期修改內容

在此先謝謝教授的反饋，學生受益良多。於期中報告時教授點出了 BuildingArea 或許與 selection bias 有關以及與其透過 VIF 並主觀的二選一刪除特徵變數，可以試著用完整的 data 來跑跑看，故我們在後續打算建立進行資料前處理與否對於房價預測影響的比較，將未進行資料前處理的資料集放入模型中做為 baseline 來查看前述資料前處理之必要性。

2.過程疑難排解

雖然說要將原始資料集直接丟入模型中，但是我們在這邊還是有做一最基本的類別欄位處理以順利進行建模，因無法直接將文字欄位套入模型中。而在建模過程中，尚未進行前處理的資料集當中有許多 NA，甚至有幾列樣本呈現全部 NA 的狀況，這樣也導致後續模型預測結果為 NA，為解決此狀況我們這邊便使用了 na.omit 語法，故整個樣本數剩下 8887 筆，雖然少了許多但仍可進行後續建模。

而最後一點改變則是，原本 CouncilArea(當地市議會)、Postcode(郵遞區號)於模型中捨棄，其中 CoucilArea 原本會以 32 個 dummy 欄位的形式放入，而 Postcode 則以 R 當中的 factor 型別輸入，但此時產生兩個問題，第一個為

欄位過多導致 Linear regression 在預測時出現了可能會 misleading 的 warning，若執行則會發生訓練資料集的損失函數大於驗證資料集的問題，第二個則是 Lasso regression 無法順利執行，這兩件問題代表資料當中可能出現過多欄位，且在 8887 筆相對較少(與原本近 25000 筆相比)，容易在分割資料時產生訓練資料集欄位與驗證資料集不對稱的情形。舉例來說，訓練資料集分別有 A、B、C 三個欄位，而驗證資料集可能為 A、B、C、D 又或者是 A、D、E...等狀況，故 Linear regression 將會發生問題，而 Lasso regression 更會因資料轉成矩陣形式不對稱而無法進行，故最後捨棄兩欄位。

六、結論

1.結果比較

	Linear Test RMSE	Lasso Test RMSE	RandomForest Test RMSE	XGBoost Test RMSE
不做資料前處理	444,346	444,211	315,328	261,925
進行資料前處理	405,151	405,162	287,265	266,672

可由以上結果發現以下兩件事：

(1). XGBoost 模型的表現最佳：

不管有沒有進行資料前處理，使用 XGBoost 能得到最低的 RMSE，也就是

說 XGBoost 的預測能力與其他三個機器學習方法相比是較佳的，其中未進行資料前處理的預測力為最好。

(2). 多數有進行前處理模型之損失函數比不做還來的低：

雖然未進行前處理在 XGBoost 的表現最好，但其實損失函數相差不大，反觀其他三個模型，進行資料前處理確實可以幫助我們達到預測更準確的效果。換句話說，進行資料前處理稍加貼近現實狀況，若無視缺失值，將其全部刪除並不納入模型中讓資料筆數變少，不僅容易造成欄位過多問題也將導致預測的表現下降，然而，這樣的情況並不一定符合所有的場合，也許再其他場合中將 NA 值都刪除會比進行資料填補的表現來得好。

再次感謝老師在期中報告時給予的回饋，讓我們查覺到應該先以原始資料進行建模這一點，以其作為 baseline，更能反應出資料處理的必要性，讓整體模型效果更佳。

2.研究發現

在本項研究發現，使用 XGBoost 機器學習預測模型最能夠達到當初所設立的目標，即幫助未來需要買房的消費者和欲賣房的房東，透過此一模型預測買賣方所提供的條件是否合理，輔助雙方進行買賣房之重大決策，以免被有心的知情人士所欺騙，花了冤枉錢買下或賣了不符合價格的房屋。反之，也能藉由此一模型輔助判斷該房屋是否物超所值，挑選到物美價廉好物件。

附錄 ：R 程式碼

~~~
---
title: "ML_midterm_proposal"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```{r}
#import package
library(glmnet)
library(lava)
library(stringr)
library(dummies)
library(corrplot)
library(VIF)
library(MASS)
library(rpart.plot)
library(tree)
library(Metrics)
library(randomForest)
library(gvlma)
library(car)
library(dplyr)
library(tidyr)
library(xgboost)
library(caret)
library(tidyverse)
```
```{r}
data <- read.csv("Melbourne_housing_FULL.csv")
```
```{r}
#data preproccessing
data$Suburb<- as.factor(data$Suburb)
~~~

~~~
data$Type<- as.factor(data$Type)
data$Method<- as.factor(data$Method)
data$SellerG<- as.factor(data$SellerG)
data$Date<-as.factor(data$Date)
data$Postcode<-as.factor(data$Postcode)
data$CouncilArea<-as.factor(data$CouncilArea)
data$Regionname<- as.factor(data$Regionname)
data$Propertycount<-as.factor(data$Propertycount)
data$Address <- as.character(data$Address)
data$Rooms <- as.factor(data$Rooms)
data$Distance<- as.integer(data$Distance)
data$Bathroom <- as.factor(data$Bathroom)
data$Car <- as.numeric(data$Car)
```
```{r}
#check data type
sapply(data, class)
```


```{r}
#Remove NA values of Price as its dependent variable
mel1 <- subset(data,(!is.na(data[,5])))
colSums(is.na(mel1))
dim(mel1)
```
```{r}
#Removing >60%
mel2 <- mel1[,c(1:14,16:21)]
colSums(is.na(mel2))
dim(mel2)
```


```{r}
mel4 <- mel2
# no preproccessing
mel5 <- mel1

#73% of the data for the rooms and Bedrooms is same i.e example if rooms==2 then
~~~

~~~
bedroom2 ==2
temp <- mel4[,c("Rooms","Bedroom2")]
bedroom2 <- temp[which(temp$Rooms == temp$Bedroom2),]
(length(bedroom2$Rooms) / length(mel4$Rooms)) * 100

#thus assigning the NA's of Bedrooms with the values of rooms.
my.na <- is.na(mel4$Bedroom2)
mel4$Bedroom2[my.na] <- mel4$Rooms[my.na]
colSums(is.na(mel4))
```
```{r}
bed.land.df <- mel4[,c("Bedroom2","Landsize")]
unique(bed.land.df$Bedroom2)
colSums(is.na(bed.land.df))
bed.land.df <- na.omit(bed.land.df)
bed.land.df <- bed.land.df[which(bed.land.df$Landsize > 0),]

colSums(is.na(bed.land.df))

bed.land.df_0 <- bed.land.df[which(bed.land.df$Bedroom2 == 0),]
bed.land.df_1 <-    bed.land.df[which(bed.land.df$Bedroom2 == 1),]
bed.land.df_2 <- bed.land.df[which(bed.land.df$Bedroom2 == 2),]
bed.land.df_3 <-    bed.land.df[which(bed.land.df$Bedroom2 == 3),]
bed.land.df_4 <- bed.land.df[which(bed.land.df$Bedroom2 == 4),]
bed.land.df_5 <-    bed.land.df[which(bed.land.df$Bedroom2 == 5),]
bed.land.df_6 <- bed.land.df[which(bed.land.df$Bedroom2 == 6),]
bed.land.df_7 <-    bed.land.df[which(bed.land.df$Bedroom2 == 7),]

#Replacing Na values with 0
mel4$Landsize[which(is.na(mel4$Landsize))] <- 0
```
```{r}
#Replacing 0 values with median values
mel4$Landsize[which(mel4$Bedroom2 == 0 & mel4$Landsize== 0)] <-
median(bed.land.df_0$Landsize[which(bed.land.df_0$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 1 & mel4$Landsize== 0)] <-
median(bed.land.df_1$Landsize[which(bed.land.df_1$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 2 & mel4$Landsize== 0)] <-
~~~

~~~
median(bed.land.df_2$Landsize[which(bed.land.df_2$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 3 & mel4$Landsize== 0) ] <-
median(bed.land.df_3$Landsize[which(bed.land.df_3$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 4 & mel4$Landsize== 0) ] <-
median(bed.land.df_4$Landsize[which(bed.land.df_4$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 5 & mel4$Landsize== 0) ] <-
median(bed.land.df_5$Landsize[which(bed.land.df_5$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 6 & mel4$Landsize== 0) ] <-
median(bed.land.df_6$Landsize[which(bed.land.df_6$Landsize > 1)])
mel4$Landsize[which(mel4$Bedroom2 == 7 & mel4$Landsize== 0) ] <-
median(bed.land.df_7$Landsize[which(bed.land.df_7$Landsize > 1)])

#Checking if all the value got atleast 100 those are zero
mel4$Landsize[which(mel4$Landsize < 120)]
summary(mel4)

#Car Column
#Putting median in all the NA values of Car column
mel4$Car <- as.numeric(mel4$Car)
mel4$Car[is.na(mel4$Car)] <- median(mel4$Car[which(!is.na(mel4$Car))])
colSums(is.na(mel4))

#Putting 0 in all the NA values of YearBuilt column
mel4$YearBuilt <- as.numeric(mel4$YearBuilt)
mel4$YearBuilt[which(is.na(mel4$YearBuilt))] <- 0
```
```{r}
#  按照最佳比例比對
mel4$Bathroom <- as.integer(mel4$Bathroom)
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 1)] <- 1
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 2)] <- 1
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 3)] <- 2
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 4)] <- 2.5
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 5)] <- 3
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 6)] <- 4
mel4$Bathroom[which(is.na(mel4$Bathroom) & mel4$Bedroom2== 7)] <- 4.5

summary(mel4)
~~~

```
mel4 <- mel4[which(mel4$CouncilArea != '#N/A'),]
colSums(is.na(mel4))
dim(mel4)

#reverting back the datatypes which were changed in order to calculate the median.
mel4$Car <- as.numeric(mel4$Car)
mel4$Bathroom <- as.factor(mel4$Bathroom)
mel4$YearBuilt <- as.factor(mel4$YearBuilt)
mel4$Rooms <- as.factor(mel4$Rooms)
```
```{r}
#omitting the missing values from lattitude and longitude
mel4 <- na.omit(mel4)

#------------
mel4$Postcode <- droplevels(mel4$Postcode)
mel4$CouncilArea <- droplevels(mel4$CouncilArea)
mel4$Regionname <- droplevels(mel4$Regionname)
mel4$Propertycount <- droplevels(mel4$Propertycount)
str(mel4)
#dropping unused levels from the dataframe.
mel4$Postcode<- as.factor(mel4$Postcode)
mel4$CouncilArea<- as.factor(mel4$CouncilArea)
mel4$Regionname<- as.factor(mel4$Regionname)
mel4$Propertycount<- as.factor(mel4$Propertycount)
mel4$Postcode <- droplevels(mel4$Postcode)
mel4$CouncilArea <- droplevels(mel4$CouncilArea)
mel4$Regionname <- droplevels(mel4$Regionname)
mel4$Propertycount <- droplevels(mel4$Propertycount)
str(mel4)

table(mel4$CouncilArea)
#-------------
#converting RegionName into numeric
mel4$Regionname <- as.character(mel4$Regionname)
mel4$Regionname[mel4$Regionname == 'Eastern Metropolitan'] <- 1
mel4$Regionname[mel4$Regionname == 'Eastern Victoria'] <- 2
```

```
mel4$Regionname[mel4$Regionname == 'Northern Metropolitan'] <- 3
mel4$Regionname[mel4$Regionname == 'Northern Victoria'] <- 4
mel4$Regionname[mel4$Regionname == 'South-Eastern Metropolitan'] <- 5
mel4$Regionname[mel4$Regionname == 'Southern Metropolitan'] <- 6
mel4$Regionname[mel4$Regionname == 'Western Metropolitan'] <- 7
mel4$Regionname[mel4$Regionname == 'Western Victoria'] <- 8

#converting method into numeric
mel4$Method = as.character(mel4$Method)
mel4$Method[mel4$Method == 'PI'] <- 1
mel4$Method[mel4$Method == 'PN'] <- 2
mel4$Method[mel4$Method == 'S'] <- 3
mel4$Method[mel4$Method == 'SA'] <- 4
mel4$Method[mel4$Method == 'SN'] <- 5
mel4$Method[mel4$Method == 'SP'] <- 6
mel4$Method[mel4$Method == 'SS'] <- 7
mel4$Method[mel4$Method == 'VB'] <- 8
mel4$Method[mel4$Method == 'W'] <- 9

#converting type into numeric
mel4$Type <- as.character(mel4$Type)
mel4$Type[mel4$Type == 'h'] <- 1
mel4$Type[mel4$Type == 't'] <- 2
mel4$Type[mel4$Type == 'u'] <- 3
```
```{r}
mel4 <- mel4 %>% separate(Date,sep = "/",into = c("Day","Month","Year"))

#converting month into season.
#spring (March, April, May),
#summer (June, July, August),
#autumn (September, October, November)
#winter (December, January, February).

mel4$Season <- mel4$Month
mel4$Season <- as.numeric(mel4$Season)
mel4$Season[which(mel4$Season == 3 | mel4$Season == 4 | mel4$Season == 5)] =
"Spring"
```

```
mel4$Season[which(mel4$Season == 6 |mel4$Season == 7 | mel4$Season == 8)] =
"Summer"
mel4$Season[which(mel4$Season == 9 | mel4$Season == 10 | mel4$Season == 11)]
= "Autumn"
mel4$Season[which(mel4$Season == 12 | mel4$Season == 1 | mel4$Season == 2)] =
"Winter"

mel4$Season <- as.character(mel4$Season)
mel4$Season[mel4$Season == 'Spring'] <- 1
mel4$Season[mel4$Season == 'Summer'] <- 2
mel4$Season[mel4$Season == 'Autumn'] <- 3
mel4$Season[mel4$Season == 'Winter'] <- 4

#correlation checking of data
my_corrdata <- mel4[,-c(1,2,7,18)]
#converting the datacolumn into numeric
my_corrdata$Regionname <- as.numeric(my_corrdata$Regionname)
my_corrdata$Method <- as.numeric(my_corrdata$Method)
my_corrdata$Type <- as.numeric(my_corrdata$Type)
my_corrdata$Rooms <- as.numeric(my_corrdata$Rooms)
my_corrdata$Distance <- as.numeric(my_corrdata$Distance)
my_corrdata$Postcode <- as.numeric(my_corrdata$Postcode)
my_corrdata$Bedroom2 <- as.numeric(my_corrdata$Bedroom2)
my_corrdata$Bathroom <- as.numeric(my_corrdata$Bathroom)
my_corrdata$Car <- as.numeric(my_corrdata$Car)
my_corrdata$YearBuilt <- as.numeric(my_corrdata$YearBuilt)
my_corrdata$Day <- as.numeric(my_corrdata$Day)
my_corrdata$Month <- as.numeric(my_corrdata$Month)
my_corrdata$Year <- as.numeric(my_corrdata$Year)
my_corrdata$Propertycount <- as.numeric(my_corrdata$Propertycount)
my_corrdata$Season <- as.numeric(my_corrdata$Season)
corr <- round(cor(my_corrdata),1)

corrplot(corr)
```

```{r}
```

```
# no preproccessing
mel5 <- na.omit(mel5)
#------------
#dropping unused levels from the dataframe.
mel5$Postcode<- as.factor(mel5$Postcode)
mel5$CouncilArea<- as.factor(mel5$CouncilArea)
mel5$Regionname<- as.factor(mel5$Regionname)
mel5$Propertycount<- as.factor(mel5$Propertycount)
mel5$Postcode <- droplevels(mel5$Postcode)
mel5$CouncilArea <- droplevels(mel5$CouncilArea)
mel5$Regionname <- droplevels(mel5$Regionname)
mel5$Propertycount <- droplevels(mel5$Propertycount)
str(mel5)
#
#converting RegionName into numeric
mel5$Regionname <- as.character(mel5$Regionname)
mel5$Regionname[mel5$Regionname == 'Eastern Metropolitan'] <- 1
mel5$Regionname[mel5$Regionname == 'Eastern Victoria'] <- 2
mel5$Regionname[mel5$Regionname == 'Northern Metropolitan'] <- 3
mel5$Regionname[mel5$Regionname == 'Northern Victoria'] <- 4
mel5$Regionname[mel5$Regionname == 'South-Eastern Metropolitan'] <- 5
mel5$Regionname[mel5$Regionname == 'Southern Metropolitan'] <- 6
mel5$Regionname[mel5$Regionname == 'Western Metropolitan'] <- 7
mel5$Regionname[mel5$Regionname == 'Western Victoria'] <- 8

#converting method into numeric
mel5$Method = as.character(mel5$Method)
mel5$Method[mel5$Method == 'PI'] <- 1
mel5$Method[mel5$Method == 'PN'] <- 2
mel5$Method[mel5$Method == 'S'] <- 3
mel5$Method[mel5$Method == 'SA'] <- 4
mel5$Method[mel5$Method == 'SN'] <- 5
mel5$Method[mel5$Method == 'SP'] <- 6
mel5$Method[mel5$Method == 'SS'] <- 7
mel5$Method[mel5$Method == 'VB'] <- 8
mel5$Method[mel5$Method == 'W'] <- 9

#converting type into numeric
```

```
mel5$Type <- as.character(mel5$Type)
mel5$Type[mel5$Type == 'h'] <- 1
mel5$Type[mel5$Type == 't'] <- 2
mel5$Type[mel5$Type == 'u'] <- 3

#
mel5$Regionname <- as.factor(mel5$Regionname) #one hot needed
mel5$Method <- as.factor(mel5$Method) #one hot needed
mel5$Type <- as.factor(mel5$Type) #one hot needed
mel5$Rooms <- as.numeric(mel5$Rooms)
mel5$Car <- as.numeric(mel5$Car)
mel5$Bathroom <- as.numeric(mel5$Bathroom)
mel5$YearBuilt <- as.numeric(mel5$YearBuilt)
mel5$Propertycount <- as.character(mel5$Propertycount)
mel5$Propertycount <- as.numeric(mel5$Propertycount)
#one hot encoding of type
type_ <- factor(mel5$Type)
dumm <- as.data.frame(model.matrix(~type_)[,-1])
mel5 <- cbind(dumm,mel5)

#one hot encoding of Method
Method_ <- factor(mel5$Method)
dumm <- as.data.frame(model.matrix(~Method_)[,-1])
mel5 <- cbind(dumm,mel5)
#
mel5$CouncilArea = as.character(mel5$CouncilArea)
save<- list()
list<- mel5$CouncilArea
u_list<- unique(list)
count<-1
for (i in u_list){
  mel5$CouncilArea[mel5$CouncilArea == i]<- count
  count<-count+1
}
#
raw_df<-mel5[,-c(7,8,10,12,13,14,16,23)]
#
#check data type
```

```
sapply(raw_df, class)
```

```{r}
#one hot encoding and datatype corrections
mel4$Regionname <- as.factor(mel4$Regionname) #one hot needed
mel4$Method <- as.factor(mel4$Method) #one hot needed
mel4$Type <- as.factor(mel4$Type) #one hot needed
mel4$Rooms <- as.factor(mel4$Rooms)
mel4$Car <- as.numeric(mel4$Car)
mel4$Bedroom2 <- as.numeric(mel4$Bedroom2)
mel4$Bathroom <- as.numeric(mel4$Bathroom)
mel4$YearBuilt <- as.numeric(mel4$YearBuilt)
mel4$Day <- as.numeric(mel4$Day)
mel4$Month <- as.factor(mel4$Month) #as we have create a new variable season
using Month we will not be using month
mel4$Propertycount <- as.character(mel4$Propertycount)
mel4$Propertycount <- as.numeric(mel4$Propertycount)
mel4$Season <- as.numeric(mel4$Season) #one hot encoding needed

#one hot encoding of type
type_ <- factor(mel4$Type)
dumm <- as.data.frame(model.matrix(~type_)[,-1])
mel4 <- cbind(dumm,mel4)

#one hot encoding of Method
Method_ <- factor(mel4$Method)
dumm <- as.data.frame(model.matrix(~Method_)[,-1])
mel4 <- cbind(dumm,mel4)

#one hot encoding of season
Season_ <- factor(mel4$Season)
dumm <- as.data.frame(model.matrix(~Season_)[,-1])
mel4 <- cbind(dumm,mel4)

mel4$CouncilArea <- str_replace_all(mel4$CouncilArea,c(" "="_"))
```

```
Council_ <- factor(mel4$CouncilArea)
dumm <- as.data.frame(model.matrix(~Council_)[,-1])
mel4 <- cbind(dumm,mel4)

#test_df <- mel4[,-c(40,41,42,43,44,45,47,48,51,54,55,56,58,59,60,61)]

colnames(mel4)
```
```{r}
df <- mel4[,-c(42,43,44,45,47,48,49,50,51,53,58,59,62,63,64)]
colnames(df)
see <-mel4[,c(42,43,44,45,47,48,49,50,51,53,58,59,62,63,64)]
```

```{r}
# with preproccessing
set.seed(1)
## 75% of the sample size

smp_size <- floor(0.7 * nrow(df))

train_per <- sample(seq_len(nrow(df)), size = smp_size)

train <- df[train_per, ]
test <- df[-train_per, ]
```
```{r}
# linear
linear_model <- lm(Price ~ .,data = train)
predicted_ys <- predict(linear_model,test)
observed_ys <- test$Price
RMSE_linear <- sqrt(sum((observed_ys - predicted_ys) ^ 2)/dim(test)[1])
RMSE_linear
```
```{r}
# lasso
```

~~~
# cv
x <- model.matrix(Price~.,data=train)
x_train <- x[,-1]
y_train <- train$Price
cv.lasso <-    cv.glmnet(x = x_train, y = y_train,type.measure = "mse")
best.lambda <- cv.lasso$lambda.min
RMSE_lasso_train<-sqrt(cv.lasso$cvm[cv.lasso$lambda == best.lambda])
# fit
fit1 <-glmnet(x = x_train, y = y_train, alpha = 1, lambda = best.lambda )
```
```{r}
# predict
x_1 <-model.matrix(Price~.,data=test)
x_test_lasso <- x_1[,-1]

predicted_lasso <- predict(fit1, s = best.lambda, x_test_lasso)
observed_lasso <- test$Price
RMSE_lasso_test <- sqrt(sum((observed_lasso - predicted_lasso) ^ 2)/dim(test)[1])
```
```{r}
# rf
rf <- randomForest(Price ~ .,data = train,.multicombine=TRUE)
rf.pred <- predict(rf,test)
RMSE_rf_test <- sqrt(sum((test[,"Price"]-rf.pred)^2)/dim(test)[1])
```


```{r}
# xgb

set.seed(1)    # For reproducibility

# Create index for testing and training data
inTrain <- createDataPartition(y = df$Price, p = 0.7, list = FALSE)

# subset power_plant data to training
training <- df[inTrain,]
~~~

~~~
# subset the rest to test
testing <- df[-inTrain,]
```


```{r}
xgb_train_x<- xgb.DMatrix(as.matrix(training%>% select(-Price)))
xgb_train_y<-training$Price
xgb_test_x<-xgb.DMatrix(as.matrix(testing%>% select(-Price)))
xgb_test_y<-testing$Price
```
```{r}
#cv
xgb_trcontrol = trainControl(
   method = "cv",
   number = 3,
   allowParallel = TRUE,
   verboseIter = FALSE,
   returnData = FALSE
)

#grid
xgbGrid <- expand.grid(nrounds = c(100,200),    # this is n_estimators in the python
code above
                                   max_depth = c(10, 15, 20),
                                   colsample_bytree = seq(0.5, 0.9, length.out = 5),
                                   ## The values below are default values in the sklearn-
api.
                                   eta = 0.1,
                                   gamma=0,
                                   min_child_weight = 1,
                                   subsample = 1
                                 )
```
```{r}
set.seed(1)

xgb_model = train(
   xgb_train_x, xgb_train_y,
~~~

```
    trControl = xgb_trcontrol,
    tuneGrid = xgbGrid,
    method = "xgbTree"
)
predicted = predict(xgb_model, xgb_test_x)
residuals = xgb_test_y - predicted
RMSE_xgb_test = sqrt(mean(residuals^2))
```

```{r}
#  不做資料前處理  (老師 Feedback)
set.seed(1)

smp_size_rawdata <- floor(0.7* nrow(raw_df))
train_per_raw <- sample(seq_len(nrow(raw_df)), size = smp_size_rawdata)

train_rawdata <- raw_df[train_per_raw, ]
test_rawdata <- raw_df[-train_per_raw, ]

# linear
linear_model_raw <- lm(Price ~ .,data = train_rawdata)

predicted_raw <- predict(linear_model_raw,test_rawdata)
observed_raw <- test_rawdata$Price
RMSE_linear_raw <- sqrt(sum((observed_raw - predicted_raw) ^
2)/dim(test_rawdata)[1])
RMSE_linear_raw
# lasso
# cv
x_raw <- model.matrix(Price~.,data=train_rawdata)
x_train_raw <- x_raw[,-1]
y_train_raw <- train_rawdata$Price
cv.lasso_raw <-    cv.glmnet(x = x_train_raw, y = y_train_raw,type.measure = "mse")
best.lambda_raw <- cv.lasso_raw$lambda.min
RMSE_lasso_train_raw<-sqrt(cv.lasso_raw$cvm[cv.lasso_raw$lambda ==
best.lambda_raw])
# fit
fit_raw_lasso <-glmnet(x = x_train_raw, y = y_train_raw, alpha = 1, lambda =
best.lambda_raw)
# predict
x_raw <-model.matrix(Price~.,data=test_rawdata)
x_test_lasso_raw <- x_raw[,-1]

predicted_lasso_raw <- predict(fit_raw_lasso, s = best.lambda_raw,
x_test_lasso_raw)
observed_lasso_raw <- test_rawdata$Price
RMSE_lasso_test_raw <- sqrt(sum((observed_lasso_raw - predicted_lasso_raw) ^
2)/dim(test_rawdata)[1])

# rf
rf_raw <- randomForest(Price ~ .,data = train_rawdata)
rf.pred_raw <- predict(rf_raw,test_rawdata)
RMSE_rf_raw <- sqrt(sum((test_rawdata[,"Price"]-
rf.pred_raw)^2)/dim(test_rawdata)[1])
```

```{r}
# xgb
set.seed(1)

# Create index for testing and training data
inTrain <- createDataPartition(y = df$Price, p = 0.7, list = FALSE)

# subset power_plant data to training
training <- df[inTrain,]



# subset the rest to test
testing <- df[-inTrain,]

#split
xgb_train_x_raw<- xgb.DMatrix(as.matrix(training%>% select(-Price)))
xgb_train_y_raw<-training$Price
xgb_test_x_raw<-xgb.DMatrix(as.matrix(testing%>% select(-Price)))
xgb_test_y_raw<-testing$Price

#cv
```

```
xgb_trcontrol = trainControl(
  method = "cv",
  number = 3,
  allowParallel = TRUE,
  verboseIter = FALSE,
  returnData = FALSE
)

#grid
xgbGrid <- expand.grid(nrounds = c(100,200),  # this is n_estimators in the python
code above
                       max_depth = c(10, 15, 20),
                       colsample_bytree = seq(0.5, 0.9, length.out = 5),
                       ## The values below are default values in the sklearn-
api.
                       eta = 0.1,
                       gamma=0,
                       min_child_weight = 1,
                       subsample = 1
                      )

#train
xgb_model_raw = train(
  xgb_train_x_raw, xgb_train_y_raw,
  trControl = xgb_trcontrol,
  tuneGrid = xgbGrid,
  method = "xgbTree"
)
#predict
predicted_raw = predict(xgb_model_raw, xgb_test_x_raw)
residuals_raw = xgb_test_y_raw - predicted_raw
RMSE_xgb_test_raw = sqrt(mean(residuals_raw^2))
```