Hogar

Miguel Figueira

12 de marzo de 2016

Preprocesamiento

hogares_googleApi.R

En este script se hace la "primera" parte del preprocesamiento que fue usar el api de Google para generar el walking.time , walking.metros , driving.time y driving.metros, además de eliminar features como fotos y piso.

En cuanto al uso del api de Google Maps tuve que hacer un ciclo donde hacia 1 petición por cada origen , debido a que algunos origenes daban error en el formato Direccion,Distrito,Italia en cambio si usaba el formato Direccion,Italia aveces agarraba otro distrito y si usaba solo Direccion o Direccion,Distrito tomaba incluso lugares fuera de Italia por lo tanto mi solución fue ir probando "tipo cascada" es decir primero:

1. Direccion , Distrito, Italia
2. Direccion , Italia
3. Direccion

Luego almaceno esta información en un csv llamado hogares.csv ya que se tarda bastante hacer tantas peticiones al api , además para evitar problemas con el internet. Además tomo solo los metros en caso de distancia y los segundos en casos de tiempo (mayor precisión)

procHogar

En este script contiene el resto del preprocesamiento entre los que se incluye:

1. Crear una columna llamada "sexo" con valores:
   - ragazzi/raggaze = 2
   - ragazzi = 0
   - ragazze = 1
2. Crear varias columnas que contienen los gastos incluidos en el precio
   - condominio
   - aseo
   - agua

- internet
- calefaccion
- internet

3. Separar las instancias , muchas instancias del dataset poseen más de una instancia en una misma fila , por lo tanto separo las filas para diferenciar aquellos que son singola / doppia (por ejemplo) en caso de que sean varios singola(habitaciones simples) simplemente tomo el de mayor valor, esto aumenta el dataset a 113 instancias (10+ de las que tenía originalmente) para esto también cree una columna nueva llamada "precios"

4. Convertir la columna de "Tipo.de.Inmueble" en valores numéricos para poder usarlo en la regresión lineal ya que es un dato importante:

Para esto use levels para saber los distintos formatos en el que estaba escrito los tipo de inmueble y los categorize como: + Apartamento = 3 + Mini Apto = 2 + Apto tipo estudio = 1

```
levels(datos$Tipo.de.Inmueble) #para conocer los tipos de datos
```

levels(datos$Tipo.de.Inmueble) #para conocer los tipos de datos

```
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Apartamento"] =
as.character(3)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Apparrtamento"] =
as.character(3)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Appartameno"] =
as.character(3)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Appartamenti"] =
as.character(3)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Appartamento"] =
as.character(3)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Mini appartamento"] =
as.character(2)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Mini Appartamento"] =
as.character(2)
datos$Tipo.de.Inmueble[datos$Tipo.de.Inmueble == "Monolocale"] =
as.character(1)
```

5. Al final decidí usar solo driving.time el cual lleve al formato de minutos

```
datos$driving.time <- datos$driving.time %/% 60

datos$driving.metros <- NULL
datos$walking.time <- NULL
datos$walking.metros <- NULL
```

6. Elimino los features que no sirven para la regresión lineal:

```
datos$Distrito <- NULL
datos$Notas <- NULL
```

```
datos$X <- NULL
datos$Dirección <- NULL
datos$Descripción <- NULL
```

7. Separo la data entre hombres y mujeres:

```
datos.female <- datos[datos$sexo == 1 | datos$sexo == 2, ]
datos.male <- datos[datos$sexo == 0 | datos$sexo == 2, ]
```

8. Elimino el feature de "sexo"" ya que al ser 2 modelos por separados no tiene mayor influencia:

9. Normalizo todos los features entre 0 y 1 (menos el precio) con la siguiente función

```
normalize <- function(x) {
  return (as.numeric(x - min(x)) / as.numeric(max(x) - min(x)))
}
```

Procesamiento

Ahora ya con los 2 datasets separados , solo falta separar la data entre training & testing para crear el modelo y probarlo

Hombres:

```
## Warning: package 'dplyr' was built under R version 3.2.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#split training & testing for males
sampl <-  sample(nrow(datos.male), floor(nrow(datos.male) * 0.75))
training.male <- datos.male[sampl, ]
testing.male <- datos.male[-sampl, ]

#generate model
model.male <- lm(training.male$precios ~ ., training.male)

#aplying model to testing
testing.male$regresion <- predict(model.male, newdata = testing.male)

#see quality of the model
```

Al usar el testing para probar el training da:

```
##    precios regresion precision
## 21     430 469.56812  39.56812
## 24     750 828.62607  78.62607
## 30     450 477.93575  27.93575
## 34     450 513.45102  63.45102
## 35     550 491.98555  58.01445
## 36     850 916.66268  66.66268
## 37    1000 893.75863 106.24137
## 41     750 833.85872  83.85872
## 42     410 464.53743  54.53743
## 47     340  65.57247 274.42753
## 50     325  59.11542 265.88458
## 54     500 512.74416  12.74416
## 56     650 512.64318 137.35682
## 59     350  83.22002 266.77998
## 60     325  84.63373 240.36627
## 61     300  49.62336 250.37664
```

Y aquí calculamos el error :

```
accuracy.male <- sum(testing.male$precision)
accuracy.male
```

```
## [1] 2026.832
```

Mujeres

```
#split training & testing for females

sampl <-  sample(nrow(datos.female), floor(nrow(datos.female) * 0.75))
training.female <- datos.female[sampl, ]
testing.female <- datos.female[-sampl, ]

#generate model
model.female <- lm(training.female$precios ~ ., training.female)
#aplying model to testing
testing.female$regresion <- predict(model.female, newdata =
testing.female)
```

Al usar el testing para probar el training da:

```
##     precios regresion  precision
## 2       850 821.39287  28.607132
## 5       650 810.78121 160.781212
## 8       450 471.89132  21.891323
## 11      500 455.21296  44.787042
## 14      300  92.06945 207.930552
## 16      450 473.06936  23.069364
```

```
## 17      700 768.15260  68.152596
## 27      600 720.43117 120.431165
## 29      300 157.95963 142.040372
## 34      450 542.19769  92.197688
## 38      450 488.98685  38.986848
## 40     1200 720.85563 479.144369
## 42      450 559.94798 109.947977
## 43     1000 810.88733 189.112671
## 50      700 811.20568 111.205678
## 51     1300 811.20568 488.794322
## 53      750 843.99592  93.995916
## 54      410 484.74183  74.741828
## 62      565 542.14463  22.855370
## 71      325 238.94913  86.050870
## 72      475 560.63773  85.637735
## 87      600 484.15819 115.841813
## 89      550 541.45487   8.545128
## 95      350 215.41549 134.584512
## 100     300 233.96165  66.038348
```

Y aquí calculamos el error :

```
accuracy.female <- sum(testing.female$precision)
accuracy.female
```

```
## [1] 3015.372
```

Eligiendo Hogares:

1. Si el precio es menor al precio que da como resultado la regresión lineal entonces ese es un apartamento(o habitación) con un muy buen precio que debe ser tomado en cuenta.
2. Si el precio es mayor al precio estimado , entonces NO es un buen hogar para tomar en cuenta.
3. Si el precio es igual al precio estimado tampoco debe ser descartado porque tiene un precio justo.

Hombres

Asumiremos que al amigo de un amigo solo le interesa el mejor precio posible es decir el caso 1

```
##    Tipo.de.Inmueble Habitaciones.Disponibles driving.time condominio
aseo
## 47                1                       1   0.06336088          0
0
##    agua internet calefaccion precios regresion precision
## 47    1        0           1     340  65.57247  274.4275
```

Mujeres

```
##    Tipo.de.Inmueble Habitaciones.Disponibles driving.time condominio
aseo
## 51                1                       0   0.02980132          0
0
##    agua internet calefaccion precios regresion precision
## 51    0        0           0    1300  811.2057  488.7943
```