Università della Calabria

Dipartimento di Economia, Statistica e Finanza "*Giovanni Anania*"

CORSO DI LAUREA MAGISTRALE
in

Data Science Per Le Strategie Aziendali.

Progetto Finale Statistical Learning

Docente

Filippo Domma

Candidati

Pierfrancesco Lindia (256641)
Teresa Gallo (252183)
Mariasole Greco (256729)
Giuseppe Macri (256751)
Mario Fidel Tito Accostupa (252418)
Cinzia Gagliardi (256808)
Edlin Villantea Savedra (257218)

Anno Accademico 2023 / 2024

INDICE

1. Abstract

Questo studio analizza i determinanti del prezzo delle proprietà immobiliari di Dubai utilizzando un dataset di 1905 osservazioni e 38 variabili acquisito su Kaggle. Attraverso l'applicazione di tecniche avanzate di regressione e ricampionamento, quali Ridge Regression, Lasso ed Elastic Net, e metodologie di validazione incrociata come Leave-One-Out Cross Validation (LOOCV) e K-Fold Cross Validation. L'obiettivo è costruire un modello predittivo robusto e accurato. Il modello ottimale è selezionato in base al Mean Squared Error (MSE), dimostrando la robustezza e l'accuratezza delle metodologie applicate.

2. Introduzione

Il mercato immobiliare di Dubai ha sperimentato una crescita straordinaria negli ultimi decenni, trasformando l'emirato in uno dei centri urbani più dinamici e moderni del mondo. Originariamente un piccolo villaggio di pescatori e commercianti di perle, Dubai ha visto un'espansione significativa a partire dagli anni '70 grazie alla scoperta del petrolio. L'implementazione di politiche economiche lungimiranti, incentivi fiscali e legislazioni favorevoli agli investimenti esteri hanno alimentato una rapidissima urbanizzazione e lo sviluppo di progetti immobiliari iconici come il Burj Khalifa, Palm Jumeirah e Dubai Marina. Di conseguenza, il mercato immobiliare di Dubai è diventato uno dei più competitivi e diversificati al mondo, con una vasta gamma di proprietà di lusso con confort incredibili, naturalmente presenti all'interno del nostro dataset. Questo studio di propone di analizzare i determinanti dei prezzi delle proprietà immobiliari a Dubai, con particolare attenzione ai quartieri più prestigiosi come Downtown Dubai e Dubai Marina.

3. Descrizione dei Dati e Gestione Variabili.

In questa fase si studia il dataset con le relative variabili, facendo tutto il necessario per Il dataset "*Dubai Properties*" è composto da 38 variabili e 1905 osservazione le variabili includono:

I. ***ID***: Identificativo dell'appartamento (rimosso dall'analisi per ridondanza).
II. ***Neighborhood***: Quartieri (Downtown Dubai, Dubai Marina e altre località).
III. ***Latitude e Longitude***: Coordinate geografiche.
IV. ***Price***: Prezzo della proprietà (*variabile dipendente*).
V. ***Size in sqft***: Dimensione in metri quadrati.
VI. ***Price in sqft***: Prezzo per metro quadrato.
VII. ***No of bedrooms***: Numero di camere da letto.
VIII. ***No of bathrooms***: Numero di bagni. Quality:
IX. ***Qualità dell'immobile*** (*High, Medium, Low, Ultra*).
X. ***Altre variabili:*** includono la presenza di balconi, aree barbecue, armadi a muro, aria condizionata centralizzata, area giochi per bambini, piscina per bambini, servizio di portineria, parcheggio coperto, elettrodomestici da cucina, atrio, servizio di pulizia, connessione internet, ammissione di animali domestici, giardino privato, palestra privata, jacuzzi, piscina privata, servizio di sicurezza,

palestra comune, piscina comune, spa comune, area studio, conformità al vastu, vista su punti di riferimento e sull'acqua, cabina armadio.

La variabile *ID*, considerando la variabile dipendente (*Price*), deve essere eliminata perché superflua ai fini predittivi. Questo perché *ID* è un identificatore univoco per ogni osservazione nel dataset e non contiene informazioni utili per la predizione de prezzo delle proprietà immobiliari. Mantenere la variabile *ID* nel modello può portare a problemi di *overfitting.* I modelli predittivo potrebbe cercare di trovare schemi nell' identificativo anche se non esistono, peggiorando la capacità del modello di generalizzare su nuovi dati. Naturalmente, diminuire il numero di variabili non necessarie aiuta a semplificare il modello, riducendo a sua volta il numero di parametri da stimare migliorando inevitabilmente l'interpretabilità del modello.

Per agevolare l'analisi e l'inserimento delle variabili all'interno del modello, sono state effettuati alcuni accorgimenti sulle variabili del data set. In particolare, alcune variabili inizialmente rappresentate come valori *Booleani (True – False)* sono state trasformate in *variabili dummy (0 - 1).*
La variabile "quality" originariamente presentava 4 livelli: "Medium", "High", "Low". Per codificare questa variabile, sono state create tre nuove Variabili Dummy "*qualityhigh*" "*qualitymedium*" "*qualitylow*". Queste variabili assumono valore 1 quando si verifica la rispettiva categoria e 0 altrimenti. In questo modo ciascuna delle c-1 categorie della variabile originaria, sono state create c-1 variabili dummy.
Per la variabile "Neighborhood" è stata trasformata. Sono stati assegnati i seguenti valori numerici:

- 0 per il quartiere "*Downtown Dubai*"
- 1 per il quartiere "*Dubai Marina*"
- 2 per tutti gli altri quartieri

Tab. 1 Trasformazioni Variabili

Variabile	Tipologia Dato Originario	Trasformazione Effettuata
ID	Numerico	Eliminata
Latitude	Numerico	Nessuna
Longitude	Numerico	Nessuna
Price	Numerico	Nessuna
Size in sqft	Numerico	Nessuna
Price per sqft	Numerico	Nessuna
N° of BedRooms	Numerico	Nessuna
N° of BathRooms	Numerico	Nessuna
Neighborhood ()	*Categoriale*	*Downtown Dubai (0),* *Dubai Marina (1)* *Altre località (2)*

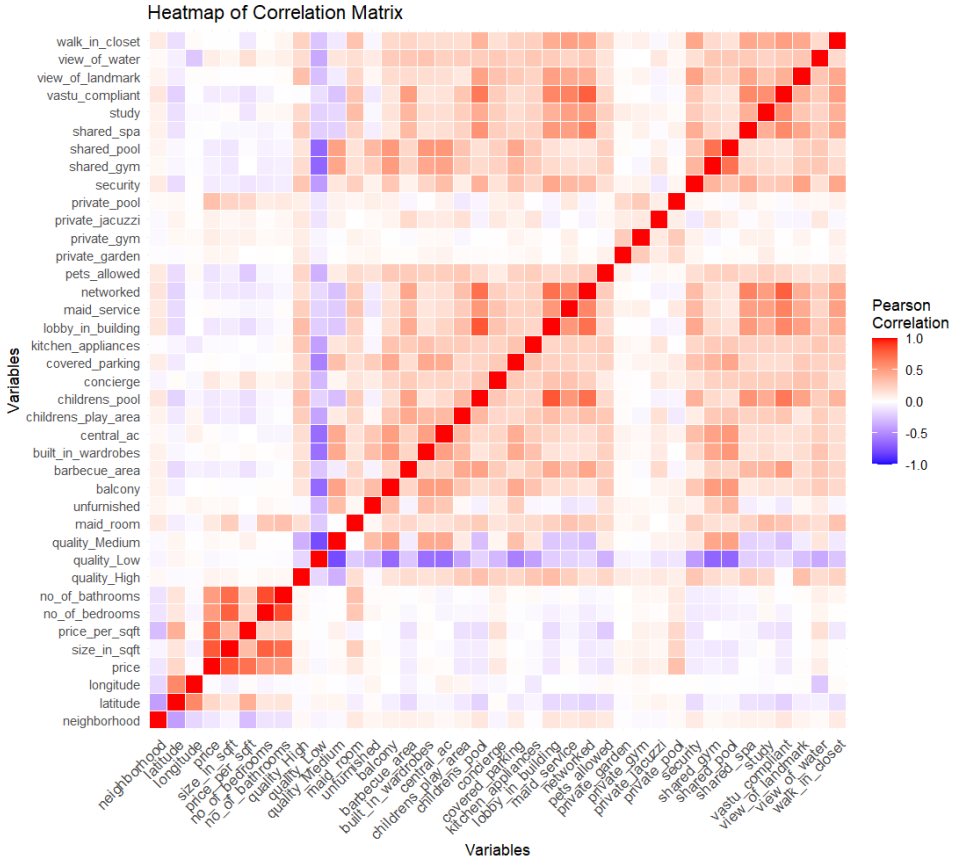
Quality	*Categoriale*	*Codifica in Dummy:* - ***qualityhigh*** (1 se "*High*", 0 altrimenti) -***qualitymedium*** (1 se *"Medium"*, 0 altrimenti) -***qualitylow*** (1 se *"Low"*, 0 altrimenti)

Maid room	*Logica (True – False)*	*Binaria (0 - 1)*
Balcony	*Logica (True – False)*	*Binaria (0 - 1)*
Barbecue_area	*Logica (True – False)*	*Binaria (0 - 1)*
Built_in_wardrobes	*Logica (True – False)*	*Binaria (0 - 1)*
Bentral_ac	*Logica (True – False)*	*Binaria (0 - 1)*
Childrens_play_area	*Logica (True – False)*	*Binaria (0 - 1)*
Childrens_pool	*Logica (True – False)*	*Binaria (0 - 1)*
Concierge	*Logica (True – False)*	*Binaria (0 - 1)*
Covered_parking	*Logica (True – False)*	*Binaria (0 - 1)*
Kitchen_appliances	*Logica (True – False)*	*Binaria (0 - 1)*
Lobby_in_building	*Logica (True – False)*	*Binaria (0 - 1)*
Maid_service	*Logica (True – False)*	*Binaria (0 - 1)*
Networked	*Logica (True – False)*	*Binaria (0 - 1)*
Pets_allowed	*Logica (True – False)*	*Binaria (0 - 1)*
Private_garden	*Logica (True – False)*	*Binaria (0 - 1)*
Private_gym	*Logica (True – False)*	*Binaria (0 - 1)*
Private_jacuzzi	*Logica (True – False)*	*Binaria (0 - 1)*
Private_pool	*Logica (True – False)*	*Binaria (0 - 1)*
Security	*Logica (True – False)*	*Binaria (0 - 1)*
Shared_gym	*Logica (True – False)*	*Binaria (0 - 1)*
Shared_pool	*Logica (True – False)*	*Binaria (0 - 1)*
Shared_spa	*Logica (True – False)*	*Binaria (0 - 1)*
Study	*Logica (True – False)*	*Binaria (0 - 1)*
Vastu_compliant	*Logica (True – False)*	*Binaria (0 - 1)*
View_of_landmark	*Logica (True – False)*	*Binaria (0 - 1)*
View_of_water	*Logica (True – False)*	*Binaria (0 - 1)*
Walk_in_closet	*Logica (True – False)*	*Binaria (0 - 1)*

4. Stima del modello e verifica Multicollinearità.

4.1 Analisi Correlazione.

Per analizzare la forza e la direzione della relazione lineare tra le variabili facciamo riferimento al coefficiente di correlazione di Pearson, che, come noto, varia tra -1 e 1. Dato il numero elevato di variabili, il modo più efficace per avere una visione chiara della correlazione è l'utilizzo di una Heatmap. La Heatmap permette di rappresentare graficamente le relazioni tra tutte le coppie di variabili, rendendo immediatamente visibili i pattern di correlazione, soprattutto per l'identificazione di eventuale multicollinearità nelle variabili indipendenti.



La scala di colori varia dal rosso al blue: il *rosso scuro* indica una forte correlazione positiva (*vicina a 1*), mentre il *blue scuro* indica una forte correlazione negativa (*vicina a -1*). Le variabili che presentano colori meno accesi indicano correlazioni meno intense. In ogni caso, in base alla tonalità del colore, possiamo capire se la correlazione è negativa, positiva o molto vicina allo zero. La multicollinearità si verifica quando due o più variabili indipendenti in un modello di regressione sono altamente correlate tra loro, il che può rendere difficile stimare i coefficienti di regressione. Ecco alcune coppie di variabili che mostrano potenziali problemi di multicollinearità.

Le variabili fortemente correlate positivamente:

- *no_of_bedrooms e size_in_sqft*: È logico aspettarsi che la dimensione di una proprietà in metri quadri sia positivamente correlata con il numero di camere da letto.
- *price e size_in_sqft*: Anche in questo caso, è prevedibile che il prezzo di una proprietà sia positivamente correlato con la sua dimensione.
- *price e no_of_bathrooms:* Un aumento del numero di bagni tende ad aumentare il prezzo.
- *price_per_sqft* e price: Il prezzo per metro quadro è positivamente correlato con il prezzo totale della proprietà.

Le variabili fortemente correlate negativamente:

- *quality_Low e quality_High*: C'è una forte correlazione negativa, il che è atteso poiché una qualità bassa è l'opposto di una qualità alta.
- *latitude e longitude con neighborhood*: Questo può riflettere la posizione geografica specifica dei quartieri, con variazioni di latitudine e longitudine che identificano diverse aree.

4.2 Stima del modello

Stimiamo il modello lineare considerando come variabile dipendente ***price.***
Per far ciò facciamo riferimento al seguente script R:

```
m1<-lm(price~., data=data1)
summary(m1)
```

```
m1<-lm(price~., data=data1)
summary(m1)

##
## Call:
## lm(formula = price ~ ., data = data1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8308956  -340378    79585   434239  9635100
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -6.439e+07  1.829e+07  -3.520 0.000442 ***
## neighborhood       3.080e+04  3.168e+04   0.972 0.331112
## latitude          -7.131e+06  5.305e+05 -13.443  < 2e-16 ***
## longitude          4.343e+06  4.387e+05   9.899  < 2e-16 ***
## size_in_sqft       2.653e+03  4.091e+01  64.842  < 2e-16 ***
## price_per_sqft     2.352e+03  4.050e+01  58.072  < 2e-16 ***
## no_of_bedrooms    -5.540e+05  4.778e+04 -11.595  < 2e-16 ***
## no_of_bathrooms   -8.686e+04  3.938e+04  -2.205 0.027545 *
## quality            5.851e+04  4.558e+04   1.284 0.199405
## maid_room         -2.791e+05  7.132e+04  -3.913 9.43e-05 ***
## unfurnished        2.946e+04  5.183e+04   0.568 0.569874
## balcony            9.628e+04  6.375e+04   1.510 0.131147
```

```
## barbecue_area         1.121e+04  8.104e+04   0.138 0.889960
## built_in_wardrobes    3.201e+04  6.282e+04   0.510 0.610432
## central_ac            1.903e+04  6.129e+04   0.311 0.756163
## childrens_play_area   5.569e+04  5.623e+04   0.990 0.322118
## childrens_pool       -1.498e+04  1.502e+05  -0.100 0.920562
## concierge            -4.171e+04  5.075e+04  -0.822 0.411274
## covered_parking       3.833e+04  5.382e+04   0.712 0.476420
## kitchen_appliances   -4.226e+04  5.153e+04  -0.820 0.412260
## lobby_in_building     9.302e+04  1.324e+05   0.703 0.482386
## maid_service          2.003e+05  1.122e+05   1.786 0.074337 .
## networked            -2.834e+05  1.693e+05  -1.674 0.094362 .
## pets_allowed          1.810e+05  5.265e+04   3.437 0.000601 ***
## private_garden       -3.003e+05  1.716e+05  -1.750 0.080311 .
## private_gym          -1.532e+05  2.551e+05  -0.600 0.548348
## private_jacuzzi       1.862e+05  1.029e+05   1.810 0.070427 .
## private_pool          8.784e+05  1.160e+05   7.572 5.73e-14 ***
## security             -8.743e+04  6.192e+04  -1.412 0.158157
## shared_gym           -7.261e+03  6.767e+04  -0.107 0.914562
## shared_pool          -8.680e+04  7.684e+04  -1.130 0.258788
## shared_spa            2.001e+04  9.919e+04   0.202 0.840119
## study                 5.429e+04  9.127e+04   0.595 0.552002
## vastu_compliant       2.274e+05  1.918e+05   1.186 0.235845
## view_of_landmark     -1.218e+04  6.907e+04  -0.176 0.860078
## view_of_water        -1.821e+05  5.353e+04  -3.401 0.000686 ***
## walk_in_closet        1.262e+05  7.701e+04   1.639 0.101467
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 904600 on 1868 degrees of freedom
## Multiple R-squared:  0.9054, Adjusted R-squared:  0.9036
## F-statistic: 496.7 on 36 and 1868 DF,  p-value: < 2.2e-16
```

'*' indica un p-value < 0.001 '**' indica un p-value < 0.01 '*' indica un p-value < 0.05 '.' indica un p-value < 0.1 ' ' (spazio vuoto) indica un p-value >= 0.1**

Residual standard error:

902100 on 1866 degrees of freedom:

$$RSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-k-1}}$$

L'errore standard residuo (RSE) è una misura della dispersione dei residui (le differenze tra i valori osservati e i valori predetti dal modello) intorno all'iperpiano. In altre parole, indica quanto i dati effettivi si discostano dai valori stimati dal modello di regressione.

Multiple R-squared:

0.906

Adjusted (Gradi di libertà) R-squared:

0.9041

F-statistic:

473.4 on 38 and 1866 DF, p-value: < 2.2e-16

La statistica F testa l'ipotesi globale che almeno uno dei coefficienti delle variabili indipendenti sia diversa da zero. In questo caso il p-value associato è molto basso.

La precedente tabella fa riferimento a:

- *Estimate:* Questa colonna mostra il coefficiente stimato per ciascun predittore. Indica l'effetto previsto di una unità di incremento della variabile predittiva sul prezzo. Un valore positivo significa che l'aumento della variabile predittiva porta ad un aumento del prezzo, mentre un valore negativo indica una diminuzione.
- *Standard Error:* Questa colonna ci dà informazioni in merito alla precisione con cui è stato stimato il coefficiente. Valori più piccoli indicano stime più precise.
- *t-value:* Il valore t è il risultato del test t per ciascun coefficiente. È calcolato come il rapporto tra il coefficiente stimato e suo errore standard. Un valore t elevato in valore assoluto indica che il coefficiente è significativamente diverso da zero.
- *Pr(>|t|):* Il valore p associato al test t. Indica la possibilità di ottenere un valore t almeno altrettanto estremo, supponendo che il valore del coefficiente sia zero. Valori p bassi (*tipicamente meno di 0.05*) suggeriscono che il coefficiente è significativamente diverso da zero.

Uno dei segnali che indica la presenza di multicollinearità e la presenza di numerosi regressori statisticamente non significativi, nonostante la presenza di un indice di Determinazione elevato.

4.3 VIF E TOLLERANCE

Per quanto riguarda invece la multicollinearità, per identificarla bisogna procedere con la valutazione di indicatori come VIF e TOLLERANCE.

La *Variance Inflaction Factor* (*VIF*) ci dice quanto la varianza di un coefficiente di regressione è gonfiata a causa della multicollinearità con le altre variabili

$$VIF(Xi) = \frac{1}{1 - R_i^2}$$

Un VIF pari a:

- ***1:*** Nessuna correlazione tra la variabile Xi e le altre variabili indipendenti.
- ***1< VIF<5:*** Moderata correlazione, generalmente accettabile.
- ***VIF>5:*** Alta correlazione, indica presenza di multicollinearità
- ***VIF>10:*** Altissima correlazione, problemi di multicollinearità gravi.

Il *Tollerance* è il reciproco del VIF e rappresenta la proporzione della variabilità di una variabile indipendente che non è spiegata dalle altre variabili indipendenti

$$Tollerance(Xi) = \frac{1}{Vif(Xi)} = 1 - R_i^2$$

Interpretazione:

- *Tollerance vicino a 1:* Bassa Multicollinearità.
- *Tollerance < 0.2:* Alta Multicollinearità
- *Tollerance < 0.1:* Altissima Multicollinearità

Variabile	VIF	Tollerance	R_i^2
neighborhood	1.323836	0.755381	0.244619
latitude	2.578874	0.387766	0.612234
longitude	2.124766	0.47064	0.52936
size_in_sqft	3.102229	0.322349	0.677651
price_per_sqft	1.709128	0.585094	0.414906
no_of_bedrooms	4.808209	0.207978	0.792022
no_of_bathrooms	4.089501	0.244529	0.755471
quality_High	20.347619	0.049146	0.950854
quality_Low	86.993856	0.011495	0.988505
quality_Medium	92.224777	0.010843	0.989157
maid_room	1.469006	0.680732	0.319268
unfurnished	1.456703	0.686482	0.313518
balcony	1.974336	0.506499	0.493501
barbecue_area	1.887665	0.529755	0.470245
built_in_wardrobes	1.938693	0.515811	0.484189
central_ac	1.96799	0.508133	0.491867
childrens_play_area	1.733601	0.576834	0.423166
childrens_pool	4.244053	0.235624	0.764376
concierge	1.411692	0.70837	0.29163
covered_parking	1.641358	0.609251	0.390749
kitchen_appliances	1.498059	0.667531	0.332469
lobby_in_building	3.916719	0.255316	0.744684

maid_service	2.237395	0.446948	0.553052
networked	4.40637	0.226944	0.773056
pets_allowed	1.399765	0.714406	0.285594
private_garden	1.141768	0.875834	0.124166
private_gym	1.189323	0.840814	0.159186
private_jacuzzi	1.192631	0.838482	0.161518
private_pool	1.303462	0.767188	0.232812
security	2.147655	0.465624	0.534376
shared_gym	2.542122	0.393372	0.606628
shared_pool	2.687622	0.372076	0.627924
shared_spa	2.115375	0.472729	0.527271
study	1.8432	0.542535	0.457465
vastu_compliant	9.86063	0.101413	0.898587
view_of_landmark	1.86129	0.537262	0.462738
view_of_water	1.528584	0.6542	0.3458
walk_in_closet	1.815827	0.550713	0.449287

Come si evince dalla tabella, ci sono molte variabili che indicano la presenza di multicollinearità, con valori di VIF e Tollerance che superano ampiamente le soglie definite. Alcune variabili presentano valori molto vicini ai limiti di accettabilità. Pertanto, si ritiene necessario applicare tecniche di regolarizzazione per risolvere questo problema.

5. Tecniche di Ricampionamento

Se l'analisi fosse di tipo inferenziale, dovremmo considerare la significatività di tutti i regressori. Molti di questi non risultano statisticamente significativi, poiché i p-value dei test marginali sono elevati. Tuttavia, dato che la nostra analisi e predittiva, non ci concentriamo sulla significatività dei singoli regressori.

In un contesto didattico, avremmo potuto suddividere il dataset in un training set e un test set. Tuttavia, poiché utilizzeremo tecniche di ricampionamento, questo passaggio sarà superfluo. Tipicamente, avremmo effettuato la suddivisione del dataset fissando il seme a 100 per inizializzare l'algoritmo della riproduzione casuale, inserendo poi il 70% delle osservazioni del training set e le restanti nel test set. Per il nostro studio sarà fondamentale definire le funzioni MSE (*Mean Squared Error*) e RMSE (*Root Mean Squared Error*). In altri contesti, avremmo calcolato l'MSE e l'RMSE sia nel training

set che nel test set, per poi introdurre la funzione di complessità al fine di incrementarla, sapendo che incrementando la complessità del modello, tramite interazioni e termini polinomiali, l'errore di addestramento tende a diminuire. Tuttavia, l'errore di test seguirà un andamento convesso: Inizialmente si ridurrà, per poi aumentare, a causa del Trade-Off tra distorsione e varianza.

5.1 Leave-One-Out Cross Validation

La LOOCV funziona nel seguente modo:

- *Divisione del dataset:* Il dataset viene diviso in *n* sottoinsiemi, dove *n* è il numero totale di osservazioni nel dataset. Ciascun sottoinsieme contiene esattamente una singola osservazione.
- *Addestramento del modello:* Il modello viene addestrato su *n-1* sottoinsiemi (*tutte le osservazioni tranne una*) e testato sull'osservazione restante.
- *Ripetizione:* Questo passaggio viene ripetuto *n* volte, cambiando ogni volta l'osservazione utilizzata per il Test.

```
library(boot)

m0glm<-glm(price~., data=data1)
m1glm<-glm(price~.+latitude*longitude, data=data1)
m2glm<-glm(price~.+latitude*longitude+size_in_sqft*price_per_sqft, data=data1)
m3glm<-glm(price~.+latitude*longitude+size_in_sqft*price_per_sqft+poly(latitude
,2)-latitude, data=data1)
m4glm<-glm(price~.+latitude*longitude+size_in_sqft*price_per_sqft+poly(latitude
,2)-latitude+poly(longitude,2)-longitude, data=data1)
```

Estraiamo il vettore δ da ogni modello, il quale corrisponde alla media dell'MSE della Cross Validation.

```
n<-nrow(data1)
cv.errorL0<-cv.glm(data1,m0glm, K=n)$delta
cv.errorL1<-cv.glm(data1,m1glm, K=n)$delta
cv.errorL2<-cv.glm(data1,m2glm, K=n)$delta
cv.errorL3<-cv.glm(data1,m3glm, K=n)$delta
cv.errorL4<-cv.glm(data1,m4glm, K=n)$delta

ddLOOCV<-cbind(cv.errorL0, cv.errorL1, cv.errorL2, cv.errorL3, cv.errorL4)
ddLOOCV

##         cv.errorL0   cv.errorL1 cv.errorL2 cv.errorL3 cv.errorL4
## [1,] 873318399744 874268197717   23.58169   23.58157   23.57966
## [2,] 873299166423 874248361068   23.58129   23.58116   23.57925
```

Dai risultati ottenuti possiamo vedere che il valore più piccolo si trova in corrispondenza del modello 4.

5.2 K-Fold Cross Validation.

La K-Fold Cross Validation funziona nel seguente modo:

- *Divisione del Dataset:* Il dataset viene in *k* sottoinsiemi o "*fold*" di dimensione approssimativamente uguali.
- *Addestramento e test:* Per ogni k, il modello viene addestrato sui *k-1 fold* e testato sul *fold* rimanente.
- *Ripetizione:* Questo processo viene ripetuto k volte, cambiando ogni volta il *fold* utilizzato per il test.

```
set.seed(31012024)
cv.errorK0<-cv.glm(data1,m0glm, K=10)$delta
cv.errorK1<-cv.glm(data1,m1glm, K=10)$delta
cv.errorK2<-cv.glm(data1,m2glm, K=10)$delta
cv.errorK3<-cv.glm(data1,m3glm, K=10)$delta
cv.errorK4<-cv.glm(data1,m4glm, K=10)$delta
ddKfold<-cbind(cv.errorK0, cv.errorK1, cv.errorK2, cv.errorK3, cv.errorK4)
ddKfold

##         cv.errorK0   cv.errorK1 cv.errorK2 cv.errorK3 cv.errorK4
## [1,] 883520032675 888566311008   23.82226   23.83987   23.28557
## [2,] 879057638318 883766165421   23.72810   23.74373   23.21865
```

Anche in questo caso il valore più piccolo si trova in corrispondenza dell'ultimo modello. Mettiamo a confronto le due tecniche, nella prima riga troviamo le medie degli MSE e nella seconda riga i valori della distorsione.

```
##         cv.errorL0   cv.errorL1 cv.errorL2 cv.errorL3 cv.errorL4   cv.errorK0
## [1,] 873318399744 874268197717   23.58169   23.58157   23.57966 883520032675
## [2,] 873299166423 874248361068   23.58129   23.58116   23.57925 879057638318
##         cv.errorK1 cv.errorK2 cv.errorK3 cv.errorK4
## [1,] 888566311008   23.82226   23.83987   23.28557
## [2,] 883766165421   23.72810   23.74373   23.21865
```

La LOOCV offre una stima leggermente migliore del MSE ma con una distorsione maggiore rispetto a K-Fold.

La K-Fold presenta una distorsione minore, indicando una maggiore stabilità nelle stime dell'errore del modello. Entrambe le tecniche forniscono risultati molto simili.

6. Ridge Regression.

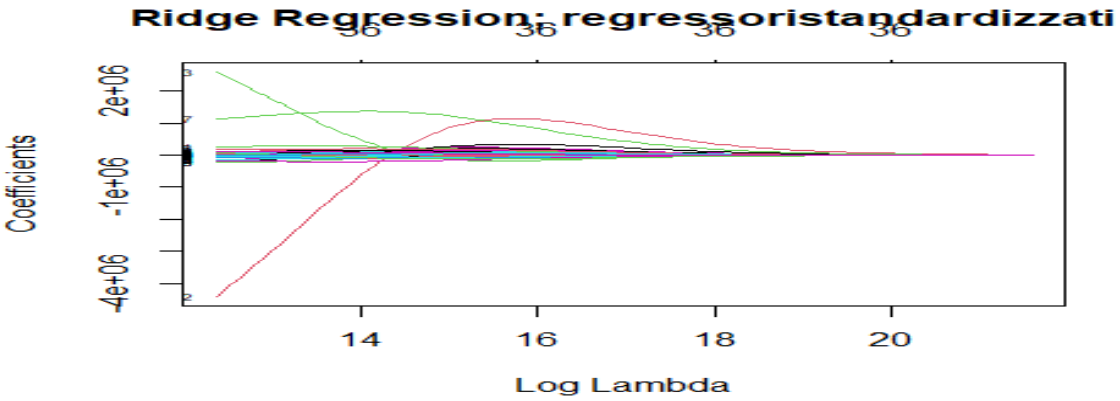
La Ridge Regression è una tecnica di regolarizzazione utilizzata per migliorare la prestazione dei modelli predittivi, specialmente quando si affrontano problemi di multicollinearità o quando il numero di predittori è maggiore del numero di osservazioni. La Ridge Regression aggiunge un termine di penalizzazione alla funzione obiettivo della regressione lineare, con l'obiettivo di ridurre l'overfitting e migliorare la generalizzazione del modello.

Il parametro λ gioca un ruolo cruciale nella Ridge Regression. Quando $\lambda = 0$, il modello si riduce a una regressione lineare ordinaria, senza alcuna penalizzazione. All'aumentare di λ, la penalizzazione sui coefficienti cresce, riducendo l'ampiezza dei coefficienti stessi. Un valore di λ molto grande può portare i coefficienti vicino allo zero, semplificando il modello ma aumentando il bias.

Come prima cosa andiamo a impostare il parametro $\lambda = 0$

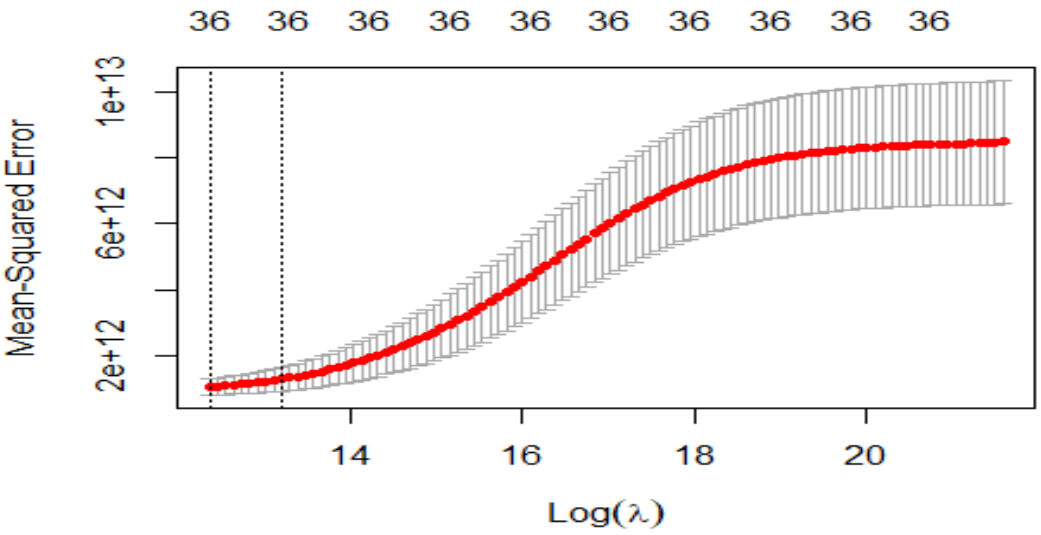
Prima di eseguire la regressione suddividiamo il dataset in due componenti: La matrice X che contiene tutte le variabili indipendenti, e il vettore y che contiene la variabile dipendente (prezzo) esclusa dalla matrice x.
Stimiamo la Ridge Regression con il dataset completo.

```
ridge.mods.ALL=glmnet(x,y,alpha=0, lambda=NULL)
dim(coef(ridge.mods.ALL))

## [1]  37 100
```



Il grafico in questione mostra l'evoluzione dei coefficienti dei predittori della Ridge Regression al variare del parametro di regolarizzazione log di λ. Sull'asse orizzontale è riportato il logaritmo dei valori di λ, mentre sull'asse verticale sono riportati i valori dei coefficienti dei predittori. Ricordo che, la linea che sembra dividere in due il grafico, è l'intercetta che non è influenzata dal parametro lambda.
Per individuare il livello di penalizzazione che bilancia al meglio il compromesso tra bias e varianza, migliorando la robustezza e la capacità predittiva del modello, utilizziamo la Cross Validation per trovare il parametro λ. Questo parametro λ viene scelto in base al valore minimo di errore ottenuto durante la Cross Validation.

6.1 Best λ con K-Fold Cross Validation



```
bestLambda<-cv.outK10$lambda.min
bestLambda

## [1] 235498
```

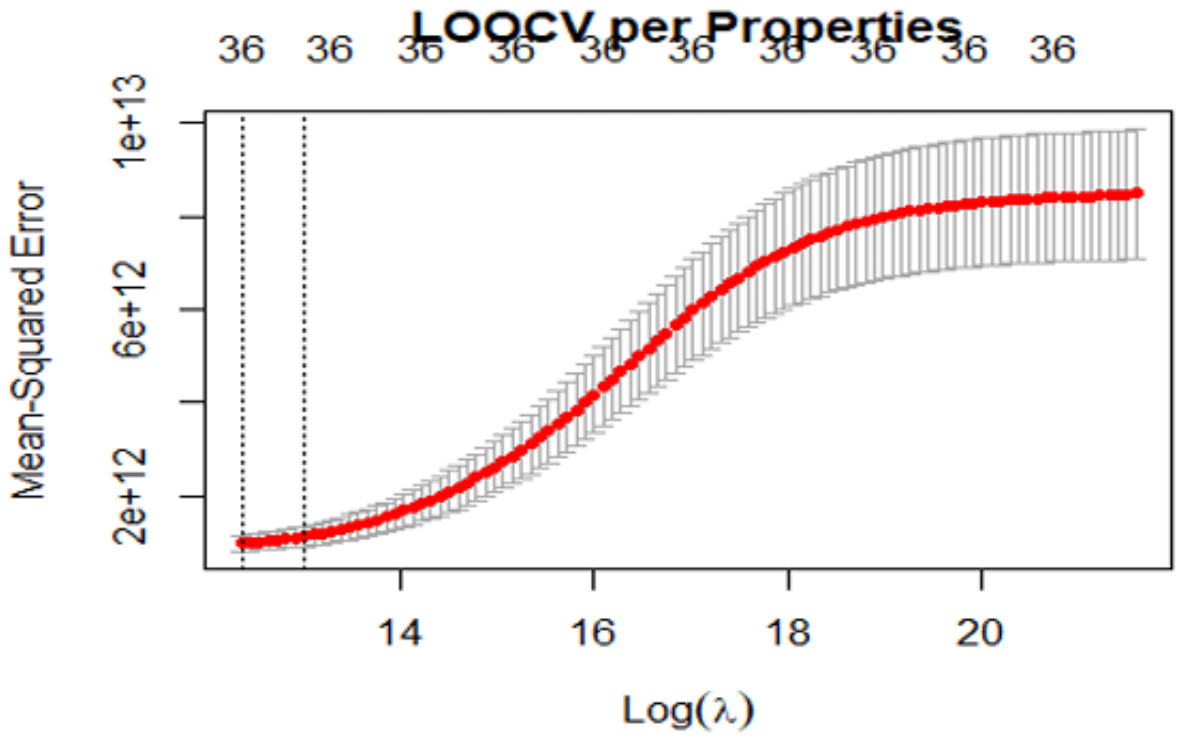
Nel grafico precedente è possibile identificare il punto in cui la linea rossa (MSE) raggiunge il minimo. Questo valore di λ, evidenziato in giallo, è quello che minimizza l'errore di previsione del modello e lo stesso valore su cui andremo a costruire il modello di Ridge Regression:

```
#best lambda con MSE minimo per stimanre modello
ridge.mod.kCV=glmnet(x,y,alpha=0, lambda = bestLambda)
coef(ridge.mod.kCV)[,1]
```

```
##          (Intercept)         neighborhood             latitude            longitude
##        -3.610959e+07         7.493477e+04        -4.414499e+06         2.598101e+06
##         size_in_sqft       price_per_sqft       no_of_bedrooms      no_of_bathrooms
##         2.150880e+03         2.148509e+03        -2.286488e+05        -3.181148e+04
##              quality            maid_room          unfurnished              balcony
##         4.621224e+04        -2.362700e+05         2.742314e+04         7.624249e+04
##       barbecue_area  built_in_wardrobes          central_ac  childrens_play_area
##        -2.695305e+03         6.274901e+04         1.007845e+05         1.753675e+04
##      childrens_pool            concierge      covered_parking   kitchen_appliances
##        -3.861561e+04         6.286502e+02         3.900771e+04        -5.031249e+04
##   lobby_in_building         maid_service            networked         pets_allowed
##         1.504489e+04         2.671695e+05        -1.611928e+05         1.079282e+05
##      private_garden          private_gym      private_jacuzzi         private_pool
##        -1.587950e+05        -1.616281e+05         1.950079e+05         1.127418e+06
##             security           shared_gym          shared_pool           shared_spa
##        -8.907912e+04        -1.759106e+04        -1.671161e+05         9.217418e+04
##                study      vastu_compliant     view_of_landmark        view_of_water
```

```
##         3.287907e+04         7.425807e+04         1.999268e+03        -1.521953e+05
##      walk_in_closet
##         1.208763e+05
```

6.2 Best λ con Leave-One-Out Cross Validation



```
bestLambdaLOOCV<-cv.outLOOCV$lambda.min
bestLambdaLOOCV

## [1] 235498
```

Costruiamo il modello con la Ridge Regression sfruttando il parametro λ:

```
ridge.mod.kCV=glmnet(x,y,alpha=0, lambda = bestLambdaLOOCV)
coef(ridge.mod.kCV)

## 37 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)        -3.610959e+07
## neighborhood        7.493477e+04
## latitude           -4.414499e+06
## longitude           2.598101e+06
## size_in_sqft        2.150880e+03
## price_per_sqft      2.148509e+03
## no_of_bedrooms     -2.286488e+05
## no_of_bathrooms    -3.181148e+04
## quality             4.621224e+04
## maid_room          -2.362700e+05
## unfurnished         2.742314e+04
## balcony             7.624249e+04
## barbecue_area      -2.695305e+03
## built_in_wardrobes  6.274901e+04
## central_ac          1.007845e+05
## childrens_play_area 1.753675e+04
## childrens_pool     -3.861561e+04
## concierge           6.286502e+02
## covered_parking     3.900771e+04
## kitchen_appliances -5.031249e+04
## lobby_in_building   1.504489e+04
## maid_service        2.671695e+05
## networked          -1.611928e+05
## pets_allowed        1.079282e+05
## private_garden     -1.587950e+05
## private_gym        -1.616281e+05
## private_jacuzzi     1.950079e+05
## private_pool        1.127418e+06
## security           -8.907912e+04
## shared_gym         -1.759106e+04
## shared_pool        -1.671161e+05
## shared_spa          9.217418e+04
## study               3.287907e+04
## vastu_compliant     7.425807e+04
## view_of_landmark    1.999268e+03
## view_of_water      -1.521953e+05
## walk_in_closet      1.208763e+05
```

In questa analisi abbiamo utilizzato due tecniche di Cross Validation, la K-fold (con K=10) e la LOOCV, per determinare il parametro ottimale di λ n per la Ridge Regression. Entrambe le tecniche hanno prodotto un valore di λ simile, indicando una notevole coerenza nei risultati. La K-Fold ha il vantaggio di essere meno costosa a livello computazionale rispetto alla LOOCV, soprattutto per dataset di grandi dimensioni.

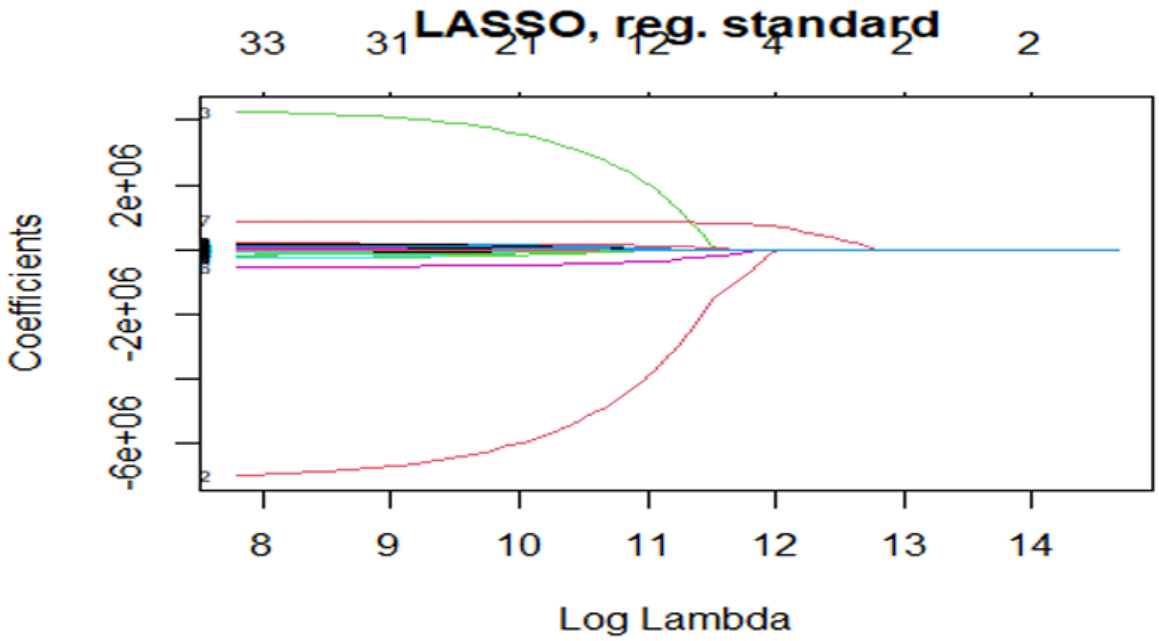
7. LASSO REGRESSION.

Si procede ora con la tecnica di regolarizzazione Lasso. Essa, a differenza della Ridge Regression, che riduce i coefficienti ma non li annulla mai completamente, la Lasso può ridurre esattamente a zero i coefficienti di alcune variabili. Questo significa che la Lasso anche una selezione automatica delle variabili, mantenendo solo quelle che contribuiscono maggiormente alla predizione. Proprio per questo impostiamo $\alpha = 1$.

```
LASSO.mods.ALL=glmnet(x,y,alpha=1, lambda=NULL)
dim(coef(LASSO.mods.ALL))
```

```
## [1] 37 75
```

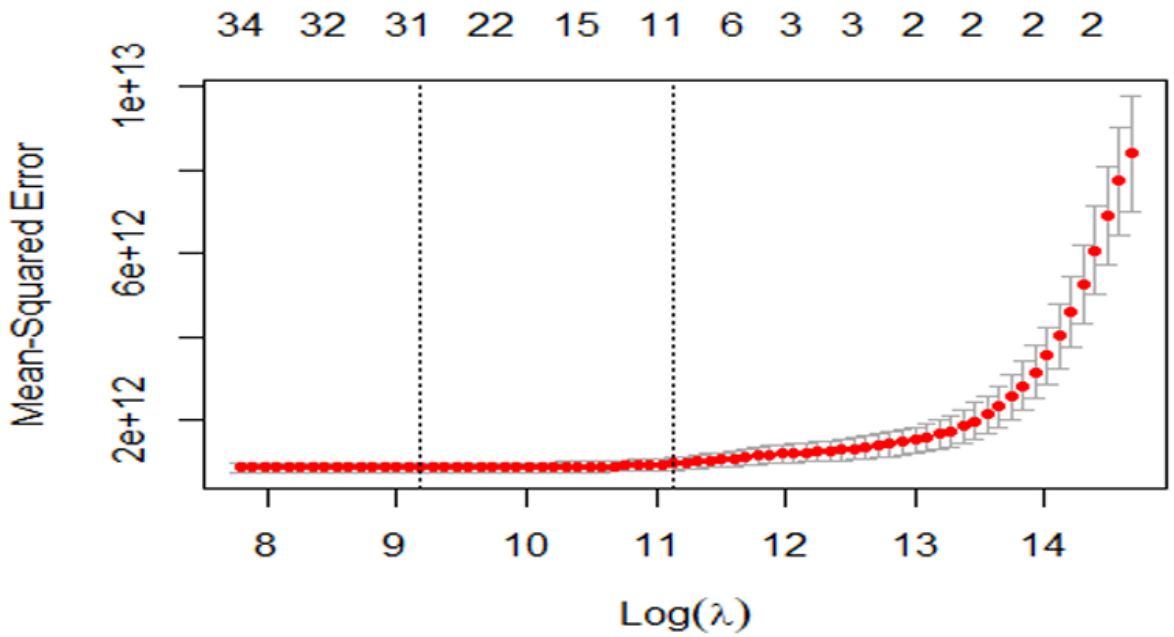
Nel seguente script andiamo tramite il pacchetto *glmnet* andiamo a definire *x* la matrice delle variabili indipendenti, *y* la matrice della variabile dipendente, $\alpha = 1$ specifichiamo che stiamo utilizzando la penalizzazione L1 tipica della Lasso Regression.
I risultati ottenuti restituiscono la matrice dei coefficienti stimati. In questo caso, l'output indica che ci sono 37 variabili inclusa l'intercetta e 75 diversi valori di λ considerati nel processo di stima. Successivamente viene mostrato il come l'andamento dei coefficienti in funzione del parametro di penalizzazione λ.



7.1 Lasso con K-Fold Cross Validation

In questa parte, viene utilizzata la tecnica di validazione incrociata K-Fold (con K=10) per determinare il miglior parametri di λ per la Lasso Regression. La linea rossa mostra l'andamento dell'MSE medio per i diversi valori di log(λ), mentre le barre mostrano l'intervallo di confidenza attorno all'MSE medio. Il minimo MSE è considerato ottimale perché minimizza l'errore di predizione del modello.

```
cv.outK10.Lasso=cv.glmnet(x,y,lambda=NULL,alpha=1, grouped=FALSE)
plot(cv.outK10.Lasso)
```

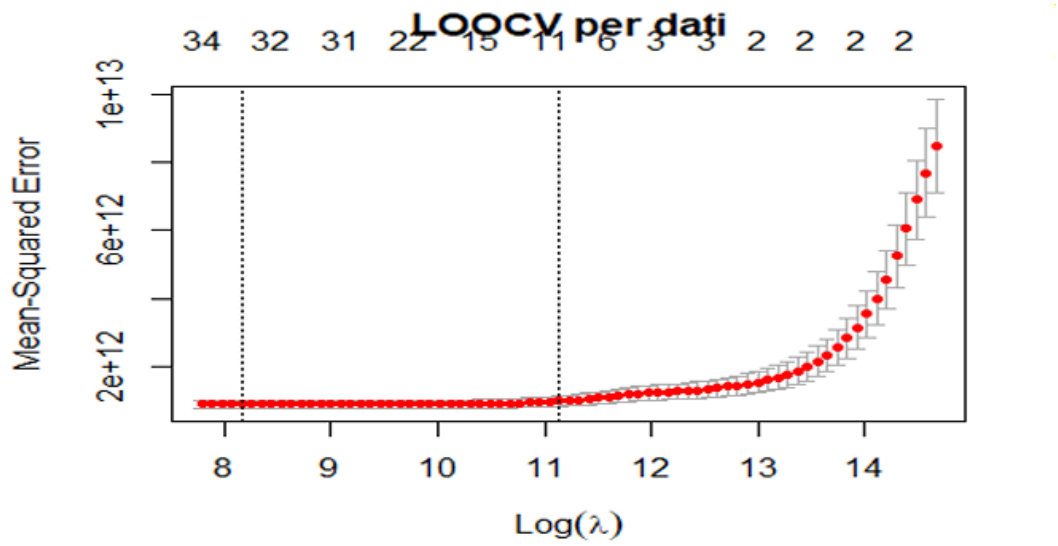



```
lambdaminL<-cv.outK10.Lasso$lambda.min
lambdaminL

## [1] 9730.807

LASSO.mod.KCV=glmnet(x,y, alpha=1, lambda=lambdaminL)
coef(LASSO.mod.KCV)[,1]

##         (Intercept)        neighborhood            latitude           longitude
##       -5.944434e+07        2.879995e+04       -6.666042e+06        4.042699e+06
##        size_in_sqft      price_per_sqft      no_of_bedrooms     no_of_bathrooms
##        2.606246e+03        2.308296e+03       -5.232974e+05       -7.223108e+04
##             quality           maid_room         unfurnished             balcony
##        3.152560e+04       -2.376415e+05        5.888573e+03        6.459585e+04
##       barbecue_area  built_in_wardrobes          central_ac childrens_play_area
##        4.008143e+03        4.511807e+03        0.000000e+00        3.963520e+04
##      childrens_pool           concierge     covered_parking  kitchen_appliances
##        0.000000e+00        0.000000e+00        9.132392e+03       -1.547640e+04
##   lobby_in_building        maid_service           networked        pets_allowed
##        0.000000e+00        1.675315e+05        0.000000e+00        1.528216e+05
##      private_garden         private_gym     private_jacuzzi        private_pool
##       -2.016274e+05       -3.415497e+02        1.413664e+05        8.673483e+05
##            security          shared_gym         shared_pool          shared_spa
##       -4.600819e+04        0.000000e+00       -4.029874e+04        0.000000e+00
##               study     vastu_compliant    view_of_landmark       view_of_water
##        1.587566e+04        6.613324e+04        0.000000e+00       -1.528736e+05
##      walk_in_closet
##        9.326188e+04
```

La Lasso Regression ha ridotto a zero i coefficienti delle variabili non significative, migliorando così l'interpretabilità del modello. Il modello risultante è più semplice e concentra l'attenzione sulle variabili che hanno un impatto significativo sulla variabile dipendente, riducendo il rischio di overfitting e migliorando la capacità generalizzazione del modello.

7.2 Lasso con Leave-One-Out Cross Validation

In questa parte dell'analisi viene utilizzata la Leave-One-Out Cross Validation (*LOOCV*) per determinare il miglior parametro λ per la Lasso Regression.




```
bestLambdaLOOCV.LASSO<-cv.outLOOCV.LASSO$lambda.min
bestLambdaLOOCV.LASSO

## [1] 3497.071
```

Successivamente con il Best λ ottenuto si stima il modello finale per l'ottenimento dei coefficienti.

```
LASSO.mod.kCV=glmnet(x,y,alpha=1, lambda=bestLambdaLOOCV)
coef(LASSO.mod.kCV)[,1]

##         (Intercept)        neighborhood            latitude           longitude
##       -3031684.126              0.000              0.000              0.000
##        size_in_sqft      price_per_sqft     no_of_bedrooms    no_of_bathrooms
##           1883.152           1831.126              0.000              0.000
##             quality          maid_room         unfurnished            balcony
##               0.000              0.000              0.000              0.000
##       barbecue_area  built_in_wardrobes         central_ac  childrens_play_area
##               0.000              0.000              0.000              0.000
##      childrens_pool          concierge     covered_parking  kitchen_appliances
##               0.000              0.000              0.000              0.000
##   lobby_in_building       maid_service          networked       pets_allowed
##               0.000              0.000              0.000              0.000
##      private_garden        private_gym     private_jacuzzi       private_pool
##               0.000              0.000              0.000         433113.253
##            security         shared_gym        shared_pool         shared_spa
##               0.000              0.000              0.000              0.000
##               study     vastu_compliant    view_of_landmark       view_of_water
##               0.000              0.000              0.000              0.000
##      walk_in_closet
##               0.000
```

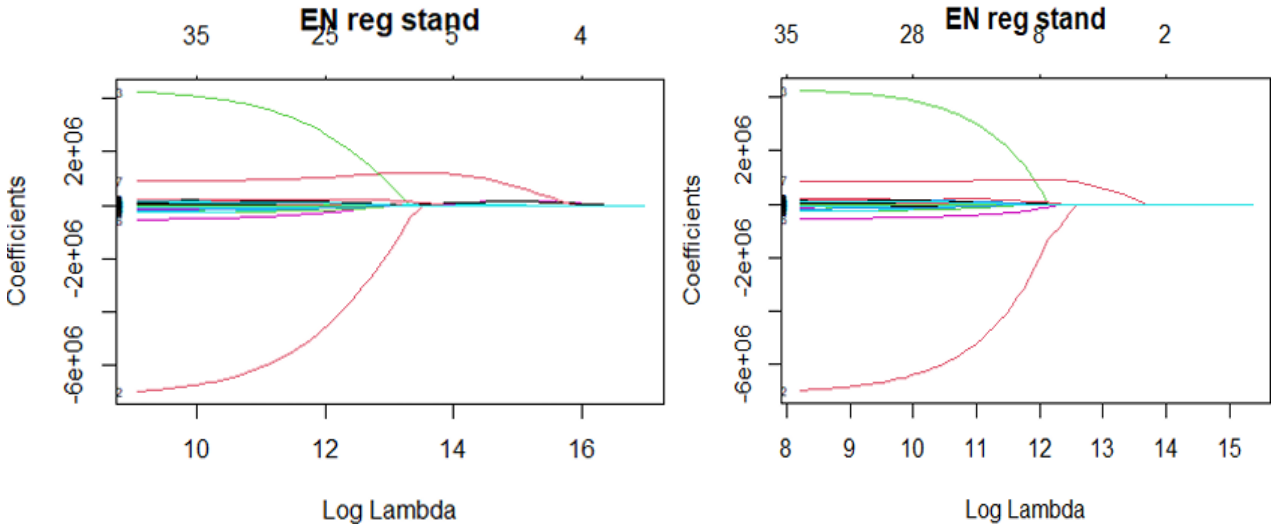
Confrontiamo i coefficienti ottenuti tra la Lasso e la Ridge Regression:

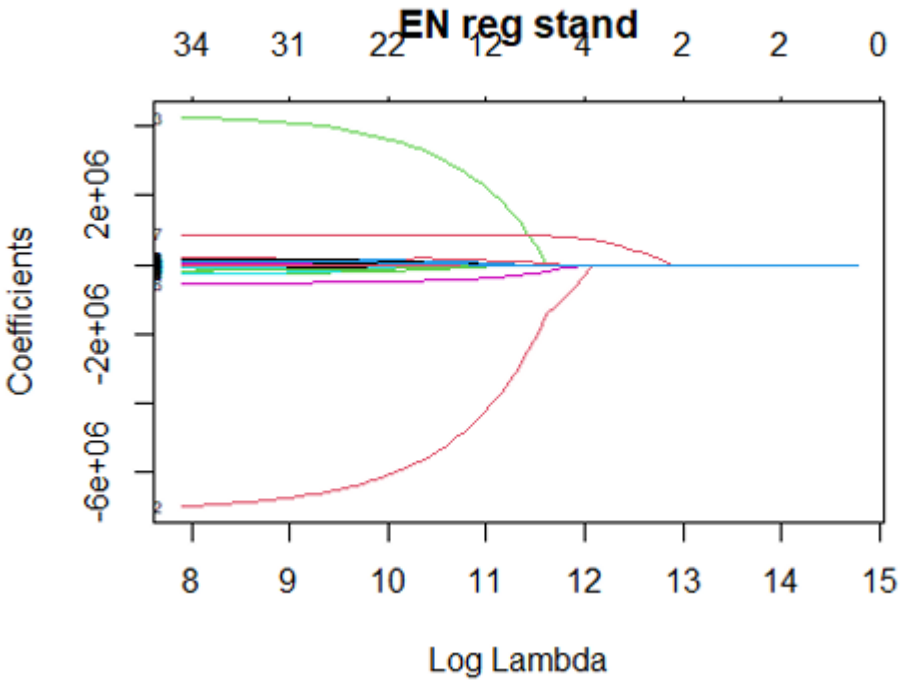
```
cbind(coef(LASSO.mod.kCV)[,1],coef(ridge.mod.kCV)[,1])

##                              [,1]          [,2]
## (Intercept)         -3031684.126 -3.610959e+07
## neighborhood               0.000  7.493477e+04
## latitude                   0.000 -4.414499e+06
## longitude                  0.000  2.598101e+06
## size_in_sqft            1883.152  2.150880e+03
## price_per_sqft          1831.126  2.148509e+03
## no_of_bedrooms             0.000 -2.286488e+05
## no_of_bathrooms            0.000 -3.181148e+04
## quality                    0.000  4.621224e+04
## maid_room                  0.000 -2.362700e+05
## unfurnished                0.000  2.742314e+04
## balcony                    0.000  7.624249e+04
## barbecue_area              0.000 -2.695305e+03
## built_in_wardrobes         0.000  6.274901e+04
## central_ac                 0.000  1.007845e+05
## childrens_play_area        0.000  1.753675e+04
## childrens_pool             0.000 -3.861561e+04
## concierge                  0.000  6.286502e+02
## covered_parking            0.000  3.900771e+04
## kitchen_appliances         0.000 -5.031249e+04
## lobby_in_building          0.000  1.504489e+04
## maid_service               0.000  2.671695e+05
## networked                  0.000 -1.611928e+05
## pets_allowed               0.000  1.079282e+05
## private_garden             0.000 -1.587950e+05
## private_gym                0.000 -1.616281e+05
## private_jacuzzi            0.000  1.950079e+05
## private_pool          433113.253  1.127418e+06
## security                   0.000 -8.907912e+04
## shared_gym                 0.000 -1.759106e+04
## shared_pool                0.000 -1.671161e+05
## shared_spa                 0.000  9.217418e+04
## study                      0.000  3.287907e+04
## vastu_compliant            0.000  7.425807e+04
## view_of_landmark           0.000  1.999268e+03
## view_of_water              0.000 -1.521953e+05
## walk_in_closet             0.000  1.208763e+05
```

Come possiamo vedere la Lasso riduce molti coefficienti a zero rispetto alla Ridge Regression, proprio per questo l'ultima tecnica di regolarizzazione che andremo ad implementare e l'Elastic Net che cerca di mitigare la rigorosità della Lasso nell'esclusione di alcuni coefficienti (*mantenendo sempre la selezione automatica dei Regressori poco significativi*) ma allo stesso tempo ridurre i coefficienti come succede nella Ridge Regression.

8. ELASTIC NET

Nella tecnica Elastic Net abbiamo due penalità ponderate rispettivamente con i propri parametri $\lambda 1$ e $\lambda 2$. L'Elastic Net si propone di unificare il meccanismo della Ridge Regression con quello della Lasso.
Impostiamo il parametro α nei vari modelli rispettivamente pari a 0.1, 0.5, 0.9. Ricordo che con α = 0 abbiamo la Ridge Regression, con α = 1 la Lasso. I valori menzionati, cercano di unire i vantaggi e mitigare i svantaggi delle due tecniche.





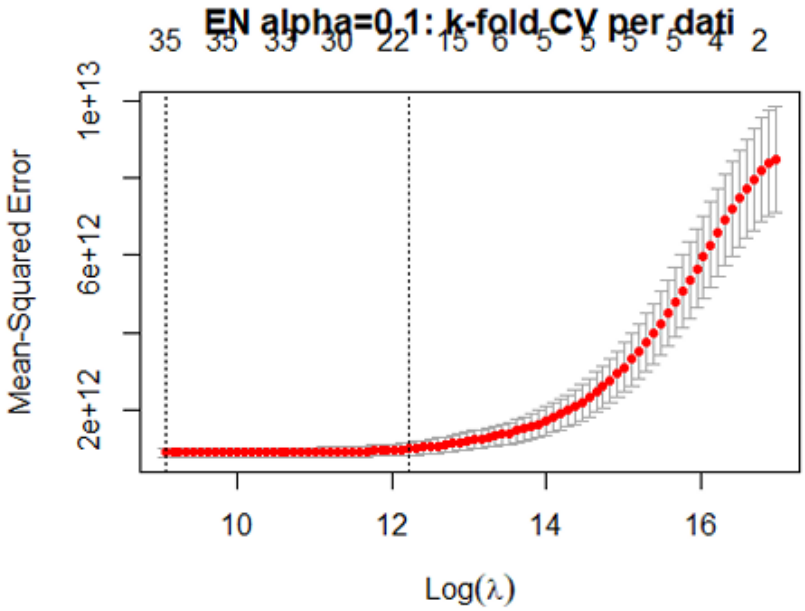
Come sempre i seguenti grafici mostrano l'andamento dei coefficienti al variare del parametri $\lambda 1$ e $\lambda 2$.

8.1 Elastic Net K-Fold Cross Validation

Applichiamo la K-Fold Cross Validation per i diversi valori di α:

- ***Alfa = 0.1:***

```
cv.outK10.EN01=cv.glmnet(x,y,lambda=NULL, alpha=0.1,grouped=FALSE)
plot(cv.outK10.EN01, main="EN alpha=0.1: k-fold CV per dati")
```




```
bestLambda.EN01<-cv.outK10.EN01$lambda.min
bestLambda.EN01
```

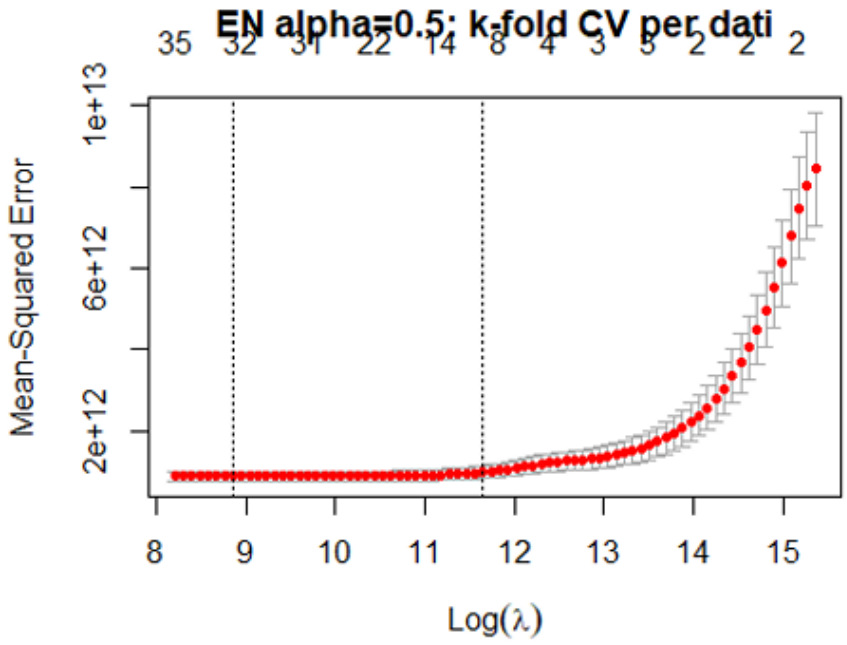
```
## [1] 8662.511
```

```
EN01.mod.kCV=glmnet(x,y,alpha=0.1,lambda=bestLambda.EN01, grouped=FALSE)
coef(EN01.mod.kCV)[,1]
```

```
##         (Intercept)        neighborhood            latitude           longitude
##       -62724920.236           33143.877       -6967606.384         4238319.018
##        size_in_sqft      price_per_sqft      no_of_bedrooms     no_of_bathrooms
##            2626.733            2340.309         -535275.202          -84312.618
##             quality          maid_room        unfurnished             balcony
##           54457.322        -274772.825          27282.792           90688.850
##       barbecue_area  built_in_wardrobes        central_ac childrens_play_area
##            9172.477           30156.647          21573.155           52394.828
##      childrens_pool           concierge     covered_parking  kitchen_appliances
##               0.000          -36228.753          35679.191          -40352.387
##   lobby_in_building        maid_service          networked        pets_allowed
##           68871.371          200263.555       -244725.879          175813.146
##      private_garden         private_gym     private_jacuzzi        private_pool
##         -286201.043        -139386.362          182946.033          889269.126
##            security          shared_gym         shared_pool          shared_spa
##          -83673.997           -5158.518          -87501.599           16279.046
##               study     vastu_compliant    view_of_landmark       view_of_water
##           51388.973          201690.326          -7999.508         -178501.365
##      walk_in_closet
##          123907.482
```

- ***ALFA = 0.5***

```
cv.outK10.EN02=cv.glmnet(x,y,lambda=NULL, alpha=0.5,grouped=FALSE)
plot(cv.outK10.EN02, main="EN alpha=0.5: k-fold CV per dati")
```



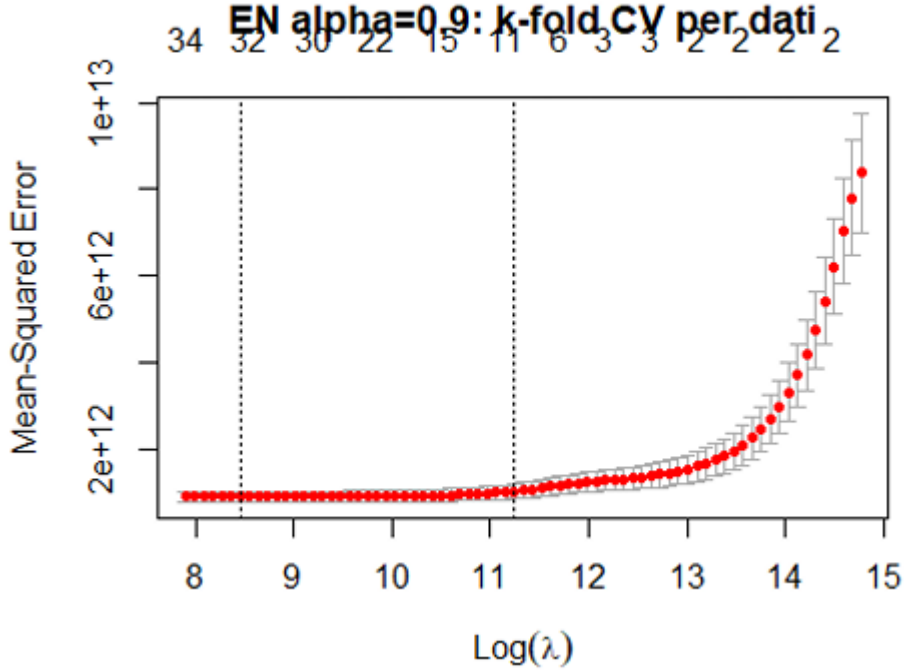

```
bestLambda.EN02<-cv.outK10.EN02$lambda.min
bestLambda.EN02
```

```
## [1] 6994.141
```

```
##         (Intercept)        neighborhood            latitude           longitude
##       -62084079.101           31058.560        -6906436.049         4199145.100
##        size_in_sqft      price_per_sqft      no_of_bedrooms     no_of_bathrooms
##             2627.070            2331.770         -536958.078          -80453.682
##             quality           maid_room        unfurnished             balcony
##           48843.435         -264233.721           20617.414          82008.801
##       barbecue_area  built_in_wardrobes         central_ac childrens_play_area
##             9233.486           21943.295           14859.565          49517.315
##      childrens_pool           concierge     covered_parking  kitchen_appliances
##                0.000          -24688.796           28121.228         -32962.160
##   lobby_in_building        maid_service          networked        pets_allowed
##           37339.544          190261.662        -154915.697          169306.592
##      private_garden         private_gym     private_jacuzzi        private_pool
##          -263229.663          -95330.920          169145.541          880279.837
##             security          shared_gym         shared_pool          shared_spa
##          -71589.277               0.000          -74411.124               0.000
##                study     vastu_compliant     view_of_landmark       view_of_water
##            40295.640          157990.461                0.000         -171434.232
##      walk_in_closet
##           113823.260
```

- ***ALFA = 0.9***

```
cv.outK10.EN03=cv.glmnet(x,y,lambda=NULL, alpha=0.9,grouped=FALSE)
plot(cv.outK10.EN03, main="EN alpha=0.9: k-fold CV per dati")
```




```
bestLambda.EN03<-cv.outK10.EN03$lambda.min
bestLambda.EN03

## [1] 4680.26
```

```
EN03.mod.kCV=glmnet(x,y,alpha=0.9,lambda=bestLambda.EN03, grouped=FALSE)
coef(EN03.mod.kCV)[,1]

##         (Intercept)        neighborhood            latitude           longitude
##       -62252322.378           29934.335       -6918594.423         4207771.080
##        size_in_sqft      price_per_sqft      no_of_bedrooms     no_of_bathrooms
##            2632.331            2331.194         -541190.967          -79627.272
##             quality          maid_room         unfurnished             balcony
##           47225.126        -261669.866           18947.204           80317.871
##       barbecue_area  built_in_wardrobes         central_ac childrens_play_area
##            9097.192           19276.267           11936.884           49045.021
##      childrens_pool           concierge     covered_parking  kitchen_appliances
##               0.000          -21754.230           25979.761          -30936.363
##   lobby_in_building        maid_service           networked        pets_allowed
##           29688.892          186624.113        -133552.842          168150.534
##      private_garden         private_gym     private_jacuzzi        private_pool
##         -258402.629          -83617.261          165452.995          874632.493
##            security          shared_gym         shared_pool          shared_spa
##          -68157.525               0.000          -68994.510               0.000
##               study     vastu_compliant    view_of_landmark       view_of_water
##           37549.882          147062.788               0.000        -169554.985
##      walk_in_closet
##          111160.038
```

Guardando i vari risultati, possiamo notare che ci sono molti meno coefficienti pari a zero rispetto alla Lasso e altrettanti vengono ridotti come succedeva nella Ridge Regression.

9. Confronto Lasso Ridge ed Elastic Net

Tramite il comando "*cbind*" mettiamo a confronto le diverse tecniche di regolarizzazione attuate:

```
cbind(coef(LASSO.mod.kCV)[,1], coef(ridge.mod.kCV)[,1],coef(EN01.mod.kCV)[,1],coef
(EN02.mod.kCV)[,1],
      coef(EN03.mod.kCV)[,1])
```

	Lasso	Ridge	EN α=0.1	EN α=0.5	EN α=0.9
(Intercept)	-3031684.126	-36109590.0	-62724920.236	-62084079.101	-62252322.378
neighborhood	0.0	74934.77	33143.877	31058.56	29934.335
latitude	0.0	-4414499.0	-6967606.384	-6906436.049	-6918594.423
longitude	0.0	2598101.0	4238319.018	4199145.1	4207771.08
size_in_sqft	1883.152	2150.88	2626.733	2627.07	2632.331
price_per_sqft	1831.126	2148.509	2340.309	2331.77	2331.194
no_of_bedrooms	0.0	-228648.8	-535275.202	-536958.078	-541190.967
no_of_bathrooms	0.0	-31811.48	-84312.618	-80453.682	-79627.272
quality	0.0	46212.24	54457.322	48843.435	47225.126
maid_room	0.0	-236270.0	-274772.825	-264233.721	-261669.866
unfurnished	0.0	27423.14	27282.792	20617.414	18947.204
balcony	0.0	76242.49	90688.85	82008.801	80317.871
barbecue_area	0.0	-2695.305	9172.477	9233.486	9097.192
built_in_wardrobes	0.0	62749.01	30156.647	21943.295	19276.267
central_ac	0.0	100784.5	21573.155	14859.565	11936.884
childrens_play_area	0.0	17536.75	52394.828	49517.315	49045.021
childrens_pool	0.0	-38615.61	0.0	0.0	0.0
concierge	0.0	628.6502	-36228.753	-24688.796	-21754.23
covered_parking	0.0	39007.71	35679.191	28121.228	25979.761
kitchen_appliances	0.0	-50312.49	-40352.387	-32962.16	-30936.363
lobby_in_building	0.0	15044.89	68871.371	37339.544	29688.892
maid_service	0.0	267169.5	200263.555	190261.662	186624.113
networked	0.0	-161192.8	-244725.879	-154915.697	-133552.842
pets_allowed	0.0	107928.2	175813.146	169306.592	168150.534
private_garden	0.0	-158795.0	-286201.043	-263229.663	-258402.629
private_gym	0.0	-161628.1	-139386.362	-95330.92	-83617.261
private_jacuzzi	0.0	195007.9	182946.033	169145.541	165452.995
private_pool	433113.253	1127418.0	889269.126	880279.837	874632.493
security	0.0	-89079.12	-83673.997	-71589.277	-68157.525
shared_gym	0.0	-17591.06	-5158.518	0.0	0.0
shared_pool	0.0	-167116.1	-87501.599	-74411.124	-68994.51
shared_spa	0.0	92174.18	16279.046	0.0	0.0
study	0.0	32879.07	51388.973	40295.64	37549.882
vastu_compliant	0.0	74258.07	201690.326	157990.461	147062.788
view_of_landmark	0.0	1999.268	-7999.508	0.0	0.0
view_of_water	0.0	-152195.3	-178501.365	-171434.232	-169554.985
walk_in_closet	0.0	120876.3	123907.482	113823.26	111160.038

Dalla tabella precedente si evince che la Lasso riduce a zero quasi tutti i coefficienti mentre la Ridge li riduce soltanto, l'Elastic Net coerente con il suo meccanismo in cui unisce la Lasso e Ridge ne azzera meno della Lasso, in questo caso solo le variabili "*shared_gym*", "*shared_spa", "view_of_landmark" e "childrens_pool"*.

10. Miglior Modello con MSE minore.

Di tutti i modelli costruiti con le varie tecniche, dobbiamo scegliere quello con il minore MSE:

```
mse.minLASSO<-cv.outK10.Lasso$cvm[cv.outK10.Lasso$lambdainL==cv.outK10.Lasso$lambd
a.min]
mse.minRR<-cv.outK10$cvm[cv.outK10$lambda==cv.outK10$lambda.min]
mse.minEN01<-cv.outK10.EN01$cvm[cv.outK10.EN01$lambda==cv.outK10.EN01$lambda.min]
mse.minEN02<-cv.outK10.EN02$cvm[cv.outK10.EN02$lambda==cv.outK10.EN02$lambda.min]
mse.minEN03<-cv.outK10.EN03$cvm[cv.outK10.EN03$lambda==cv.outK10.EN03$lambda.min]


cbind(mse.minLASSO,mse.minRR,mse.minEN01,mse.minEN02,mse.minEN03)

##        mse.minRR  mse.minEN01  mse.minEN02  mse.minEN03
## s85 1.036417e+12 878780246389 875229879665 899860513666

min(cbind(mse.minLASSO,mse.minRR,mse.minEN01,mse.minEN02,mse.minEN03))

## [1] 875229879665
```

Possiamo concludere affermando che il miglior modello di apprendimento è quello relativo alla tecnica Elastic Net con valore di α pari a 0.5, poiché presenta il valore MSE più piccolo tra tutti.