

**STATISTICAL DATA ANALYSIS**

**DOCUMENTAZIONE PROGETTO**

**Membri:**

* **Gargiulo Michele, mat. 0622701137**
* **Marchesano Riccardo, mat. 0622701198**
* **Sabini Pietro, mat. 0622701112**

**ANNO ACCADEMICO 2019-2020**

Sommario

[Introduzione 3](#_Toc58491870)

[Composizione del dataset 3](#_Toc58491871)

[Google Colab – Classificazione 4](#_Toc58491872)

[Naïve Bayes 4](#_Toc58491873)

[Dataset sintetico 4](#_Toc58491874)

[Dataset reale 6](#_Toc58491875)

[KNN 8](#_Toc58491876)

[Dataset sintetico 8](#_Toc58491877)

[Dataset reale 10](#_Toc58491878)

[Logistic Regression 13](#_Toc58491879)

[Dataset sintetico 13](#_Toc58491880)

[Dataset reale 14](#_Toc58491881)

[Naïve Kernel 17](#_Toc58491882)

[Dataset sintetico 17](#_Toc58491883)

[Dataset reale 17](#_Toc58491884)

[RStudio 20](#_Toc58491885)

[Regressione Multipla 20](#_Toc58491886)

[Metodi di ricampionamento 23](#_Toc58491887)

[Validation set 23](#_Toc58491888)

[K-Fold Cross Validation 23](#_Toc58491889)

[Bootstrap 24](#_Toc58491890)

[Subset Selection 25](#_Toc58491891)

[Best Subset Selection 25](#_Toc58491892)

[Stepwise Selection 29](#_Toc58491893)

[Metodi di regolarizzazione 31](#_Toc58491894)

[Ridge 31](#_Toc58491895)

[LASSO 33](#_Toc58491896)

[PCR 34](#_Toc58491897)

[PLS 35](#_Toc58491898)

# Introduzione

Il presente progetto è stato realizzato utilizzando un dataset reperito da Kaggle, riguardante alcune valutazioni in merito ai servizi di diverse linee ferroviarie. A tale dataset sono stati applicati i diversi metodi di regressione e classificazione analizzati durante il corso, corredati di grafici e opportune valutazioni e osservazioni finali per ognuno. In ognuna delle due parti, sono stati cambiati leggermente i ruoli delle features, in accordo all’interesse dei nostri calcoli.

La documentazione presentata è suddivisa in due macro-sezioni, una dedicata alla parte di RStudio (per i metodi di regressione), l’altra dedicata alla parte di Google Colab/Python, utilizzata per i metodi di classificazione.

## Composizione del dataset

Il dataset reperito è composto di 79576 entries e 13 colonne, di cui solo alcune sono state considerate rilevanti ai fini del progetto. Le colonne, o dimensioni delle features, sono le seguenti:

* **indice**: un semplice indice progressivo, che indica la row corrispondente;
* **company**: rappresenta la linea ferroviaria;
* **traveller\_type**: rappresenta la categoria del viaggiatore. Questa colonna contiene molti dati mancanti, ed essendo l’unica a presentare tale problematica, è stata semplicemente esclusa;
* **class**: rappresenta il tipo di vagone (Economy, Business, Premium);
* **final\_vote**: è il voto finale assegnato al particolare treno (da 1 a 10);
* **comfort\_score**: è il voto assegnato alla comodità del sedile e del viaggio in generale (da 1 a 5);
* **service\_score**: è il voto assegnato al servizio offerto nel vagone (da 1 a 5);
* **food\_score**: è il voto assegnato alla qualità delle vivande offerte dal servizio di ristorazione (da 1 a 5);
* **enjoyment\_score**: è il voto assegnato ai servizi di intrattenimento, come ad esempio film proiettati, riviste da poter leggere, etc. (da 1 a 5);
* **station\_score**: è il voto assegnato al servizio offerto alla stazione (da 1 a 5);
* **internet\_connection\_score**: è il voto assegnato alla qualità della connessione internet durante il viaggio (da 1 a 5);
* **expendiency\_score**: è il voto assegnato al rapporto qualità/prezzo (da 1 a 5);
* **approved**: è un valore booleano, indicante se il treno in questione sia consigliato (1) o meno (0);

Per i nostri scopi, abbiamo ritenuto opportuno utilizzare per la classificazione/regressione solamente le colonne dei voti, più la colonna ‘approved’, in quanto queste erano le colonne con meno valori mancanti (i quali si è scelto di sostituire con valori casuali, preferendo non perdere entries del dataset data la quantità di dimensioni). Nondimeno, tali colonne sono risultate molto più convenienti ai calcoli (contenendo perlopiù valori numerici) e, secondo l’opinione dei membri, più significative, considerando le labels utilizzate nelle due parti del progetto.

# Google Colab – Classificazione

In questa parte del progetto, è stato scelto come paramentro da stimare ‘approved’, mentre come parametri di ingresso sono stati scelti, come già detto, tutti quelli indicanti un voto (7 predittori).  
I metodi applicati sono essenzialmente quattro:

* Naïve Bayes
* K-NN
* Logistic Regression
* Naïve Kernel

È stata eseguita una normalizzazione sui voti (‘final\_vote’ è in scala da 1 a 10, mentre gli altri da 1 a 5).

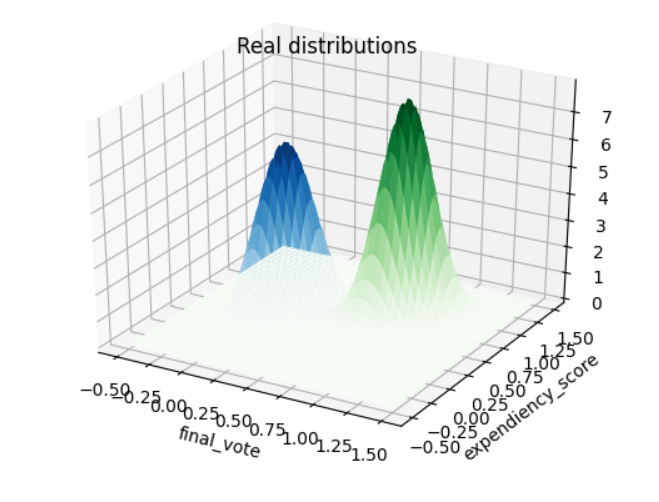
Per l’analisi dei dati, è stato inizialmente prodotto un dataset sintetico, con lo scopo di valutare la distribuzione dei dati all’interno del dataset, e sono stati utilizzati come parametri ‘final\_vote’ e ‘expendiency\_score’; si è supposto, inoltre, che la distribuzione dei valori seguisse una multivariata normale.

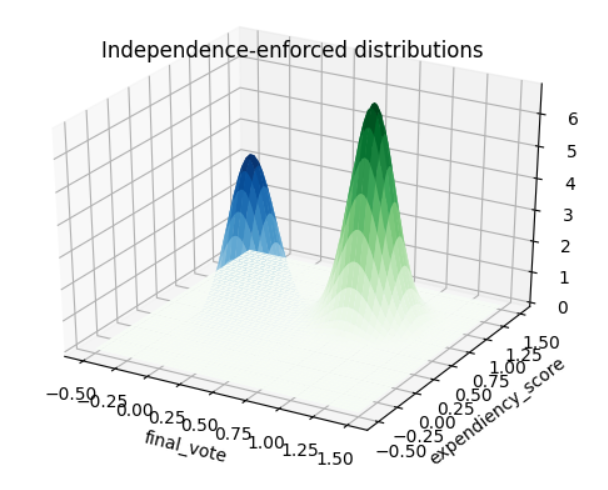
## Naïve Bayes

Tale classificatore assume che le features siano tra loro indipendenti, e cerca di massimizzare la probabilità a posteriori.

### Dataset sintetico

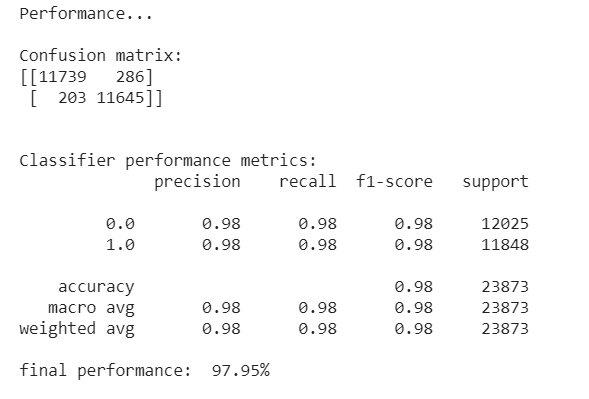
Come già accennato, sono state utilizzate le colonne ‘final\_vote’ e ‘expendiency\_score’ per il dataset sintetico, e sono stati calcolati i grafici relativi a queste due colonne per entrambe le classi di ‘approved’, sia nel caso reale che nell’ipotesi di indipendenza:





La scala di blu rappresenta la classe 0, mentre la scala di verde rappresenta la classe 1. Notiamo una chiara separazione delle due classi, inoltre il caso reale e quello delle variabili indipendenti non differiscono in modo sostanziale, e ciò indica una bassa correlazione tra i due parametri ‘final\_vote’ e ‘expendiency\_score’. Ciò dovrebbe consentirci di proseguire con l’analisi Naïve Bayes senza particolari timori riguardo l’accuratezza delle stime finali.

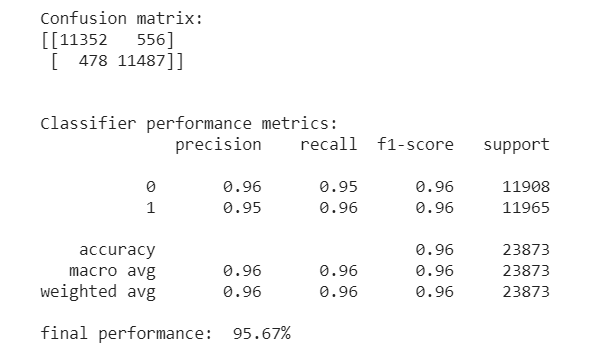
Di fatto, la performance è stata del 97.95% in questo caso:



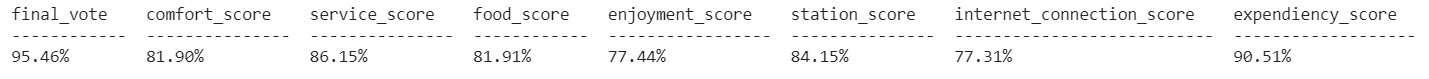
### Dataset reale

È stato applicato il metodo Naïve Bayes utilizzando varie combinazioni di parametri.

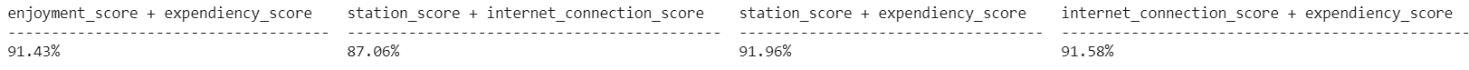
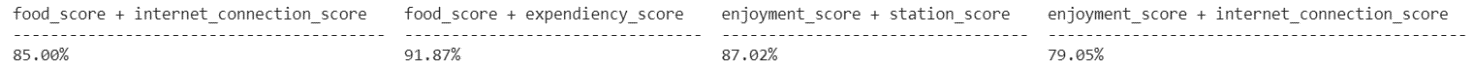
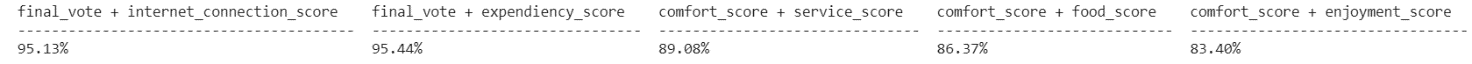
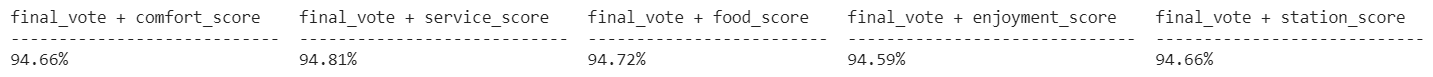
#### ‘final\_vote’ e ‘expendiency\_score’



#### Tutti i parametri singolarmente

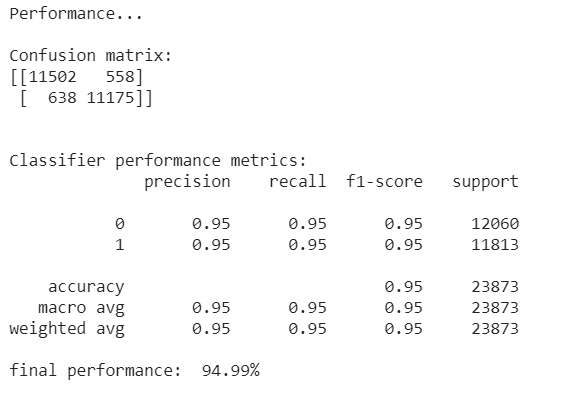


#### Tutte le coppie di parametri



Notiamo che i risultati più alti sono dati dalle coppie contenenti ‘final\_vote’ o ‘expendiency’, e la coppia contenente questi due parametri dà la performance migliore.

#### Tutti i parametri insieme



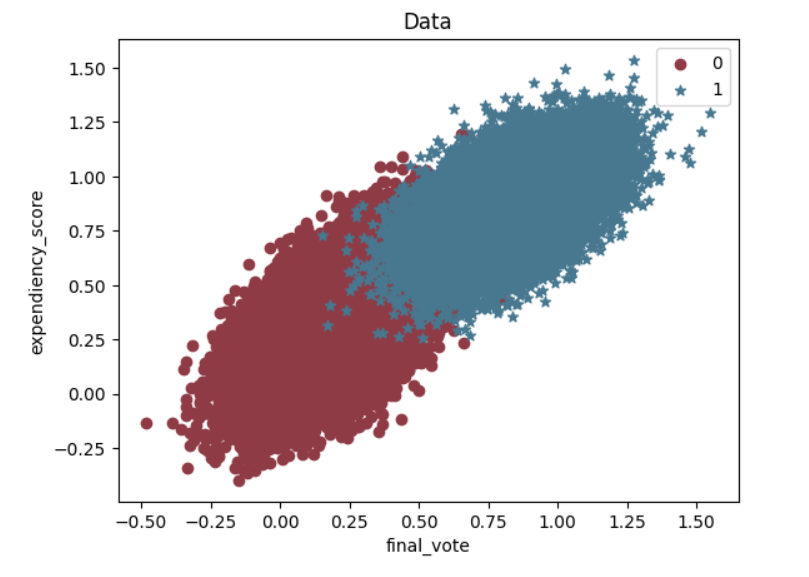
## KNN

Tale metodo richiede la regolazione dell’iperparametro K, a cui segue il fitting vero e proprio. A tale scopo è stato partizionato il dataset nel seguente modo:

* Training set: con questa parte si esegue il fitting, dunque si considerano i K elementi più vicini ad ogni sample;
* Test set: con questa parte si valutano le performance;
* Validation set: con questa parte si valuta il miglior K per la classificazione, sulla base del minor errore quadratico medio MSE.

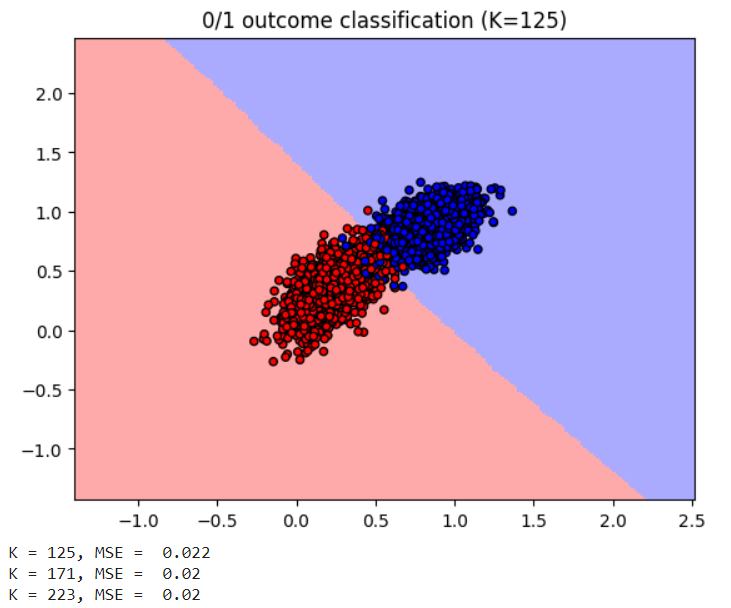
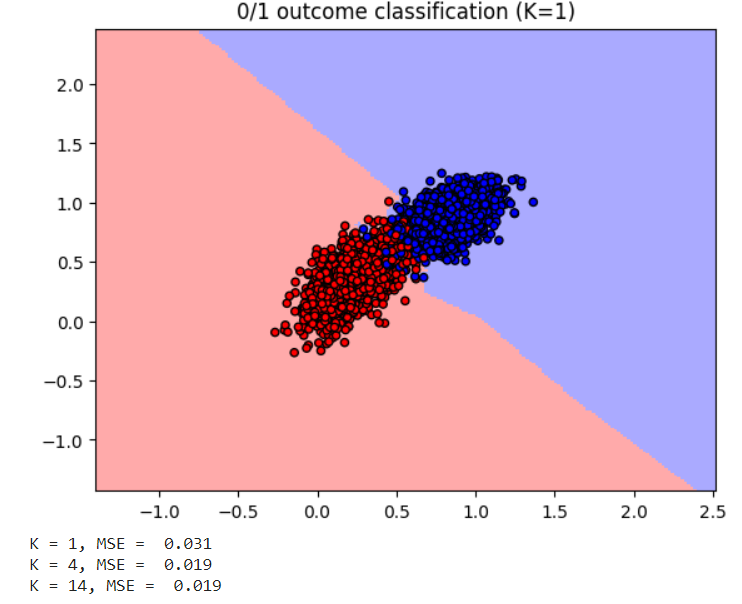
### Dataset sintetico

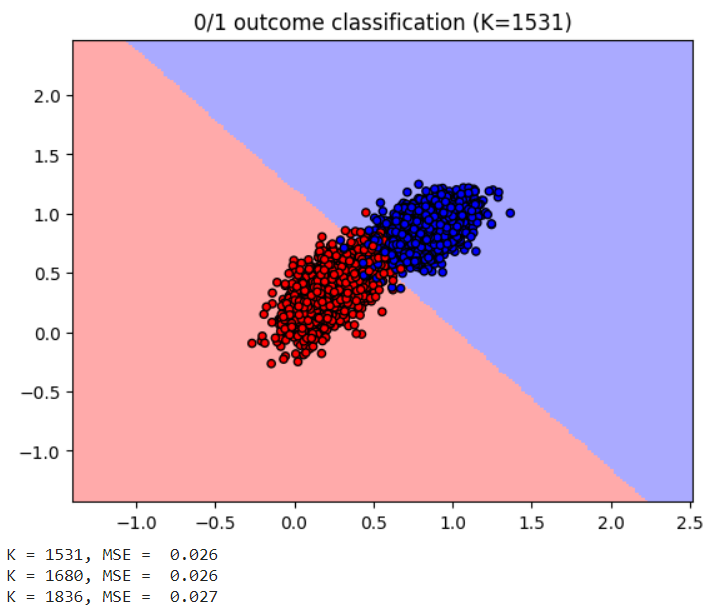
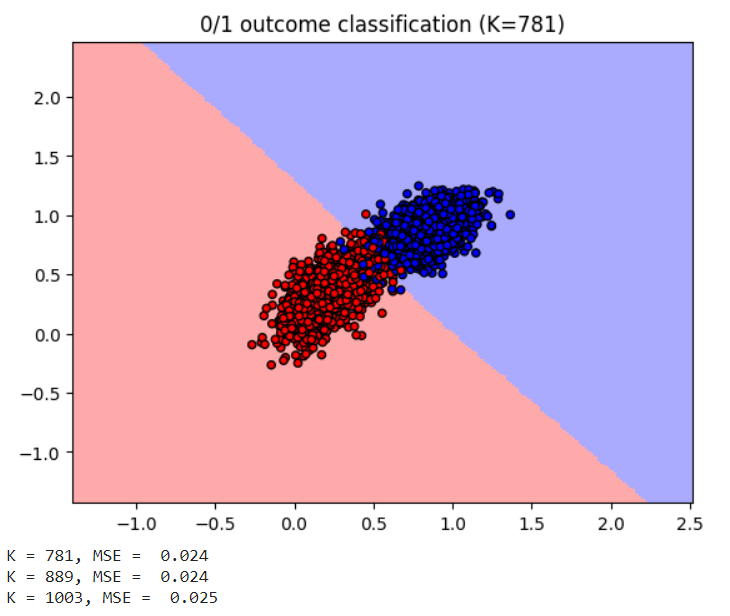
La distribuzione dei dati, sempre riguardo ai due parametri di esempio, è la seguente:

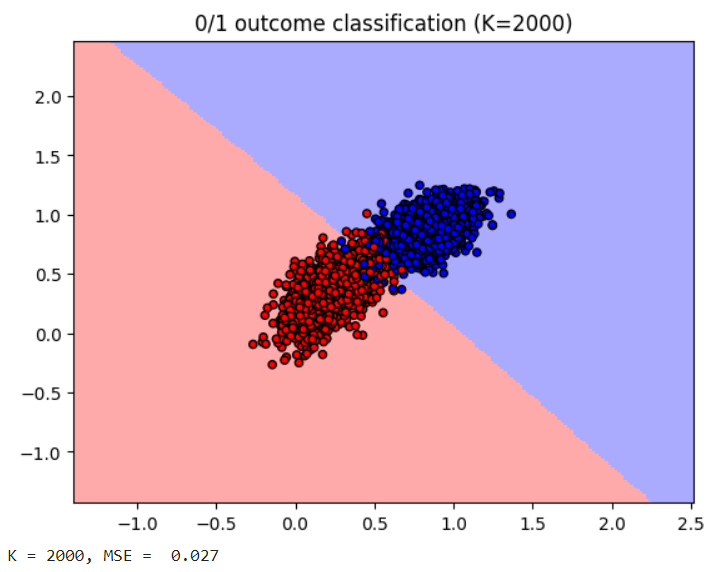


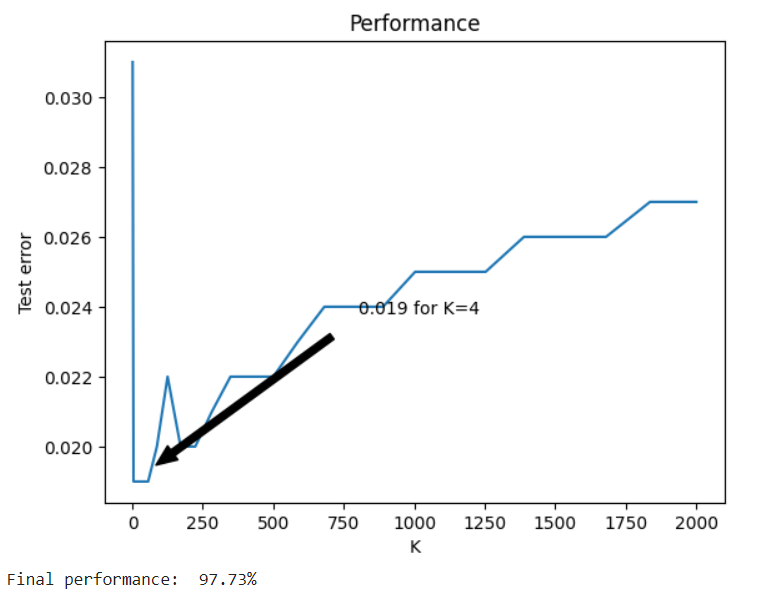
Notiamo una bassa sovrapposizione delle due classi (0 rosso, 1 blu), e ciò è conveniente alla buona riuscita di tale metodo.

Sono state utilizzate le funzioni della libreria ‘sklearn’ per l’applicazione di questo metodo. Inoltre, il framework Spark è stato adoperato per parallelizzare il calcolo. Sono stati testati 25 valori di K nell’intervallo [1,2000], di cui mostriamo nel seguito alcuni grafici delle performance:







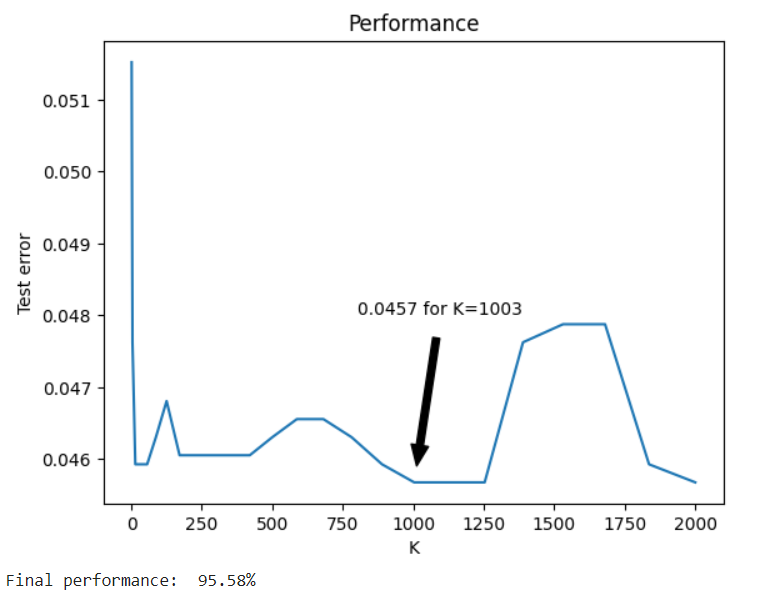
Di seguito viene riportato il grafico delle prestazioni dei vari valori testati di K:

In generale, la suddivisione abbastanza netta delle classi dei campioni permette di ottenere buone prestazioni con quasi tutti i valori di K testati, di cui il migliore risulta 4.

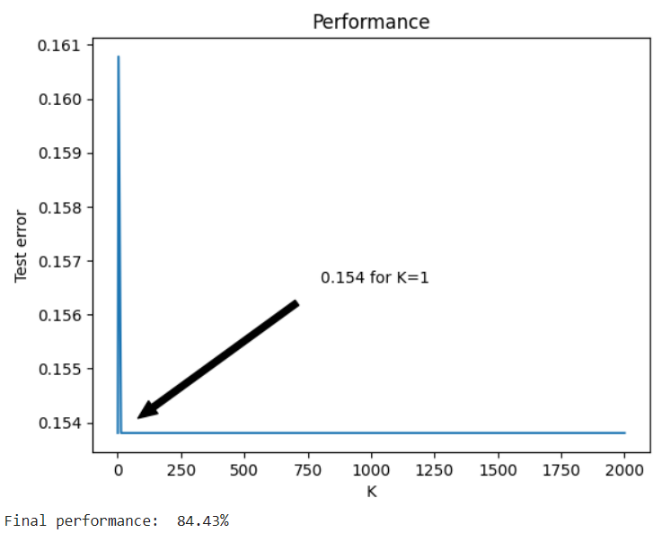
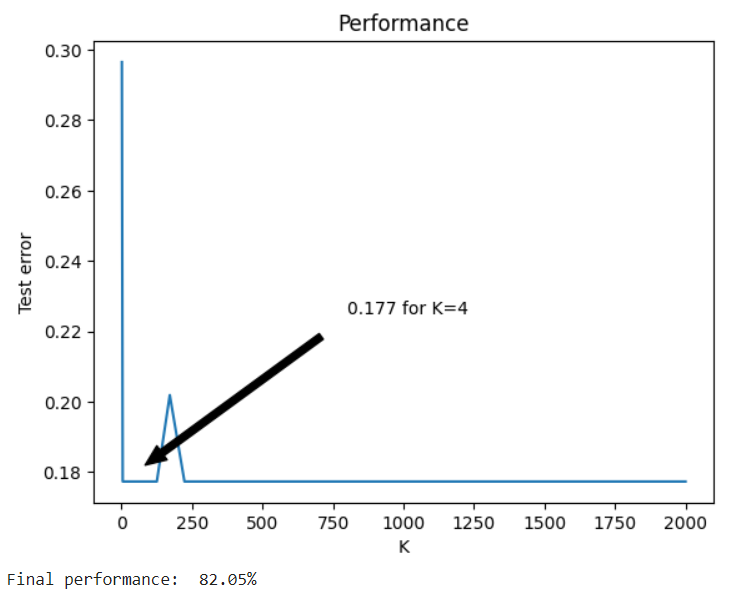
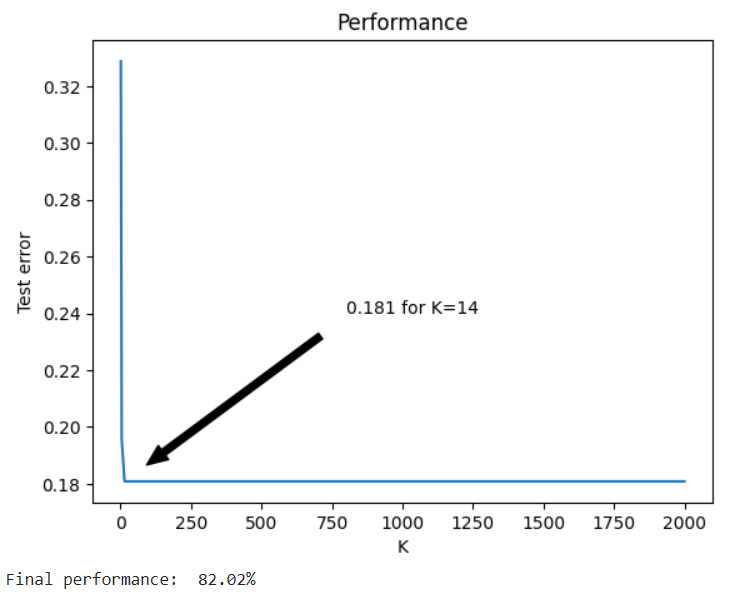
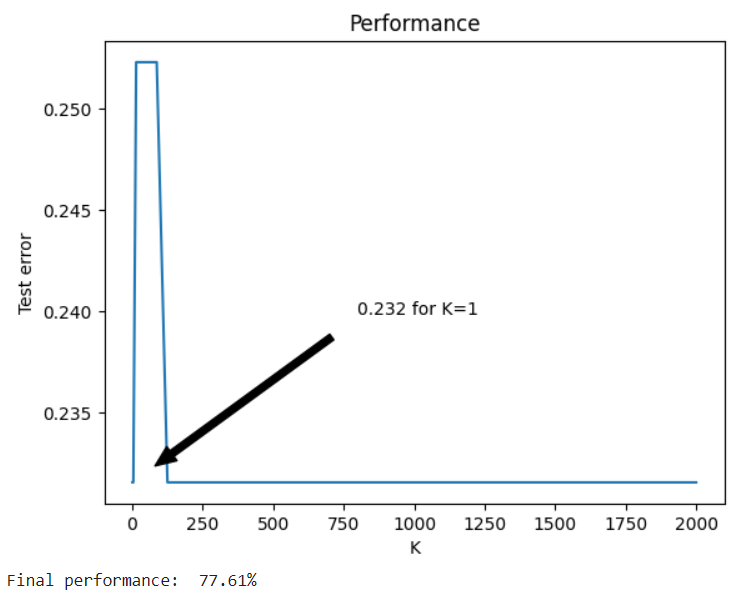
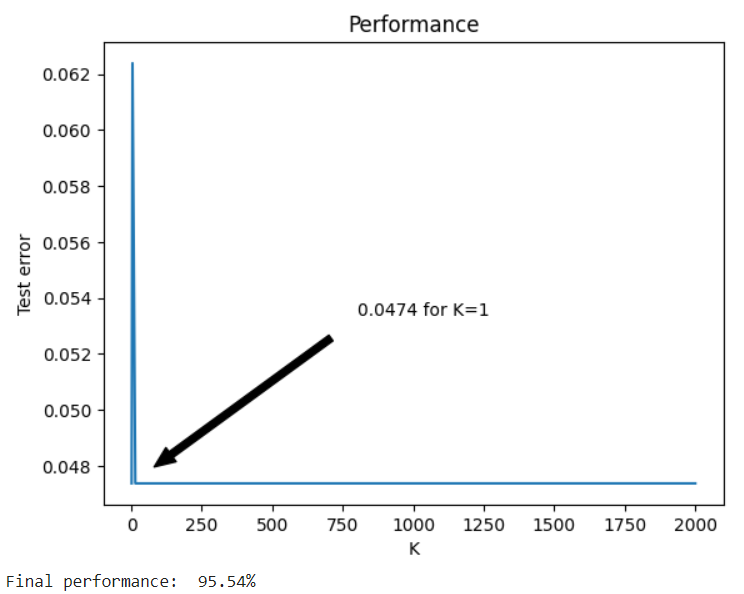
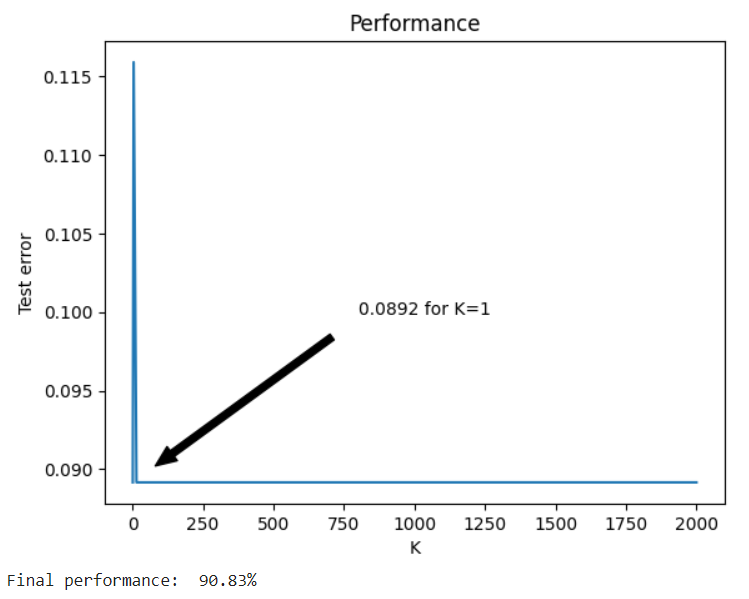
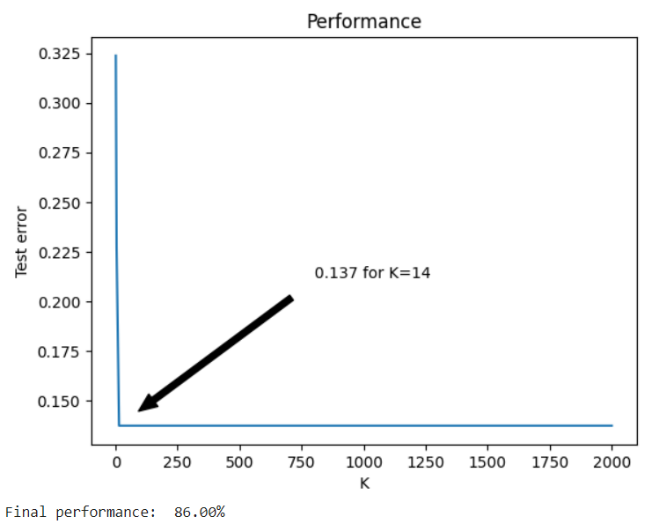
### Dataset reale

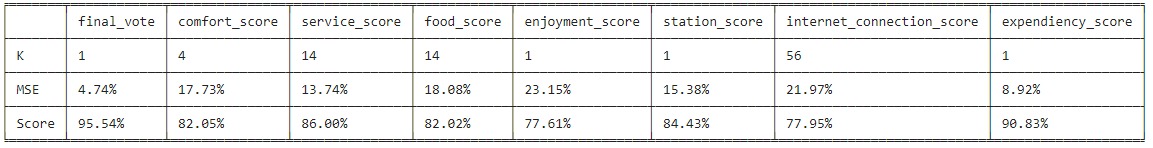
Anche in questo caso sono state testate diverse combinazioni di parametri, ad eccezione delle coppie, le quali avrebbero richiesto un tempo di calcolo eccessivo e impraticabile.

#### ‘final\_vote’ e ‘expendiency\_score’



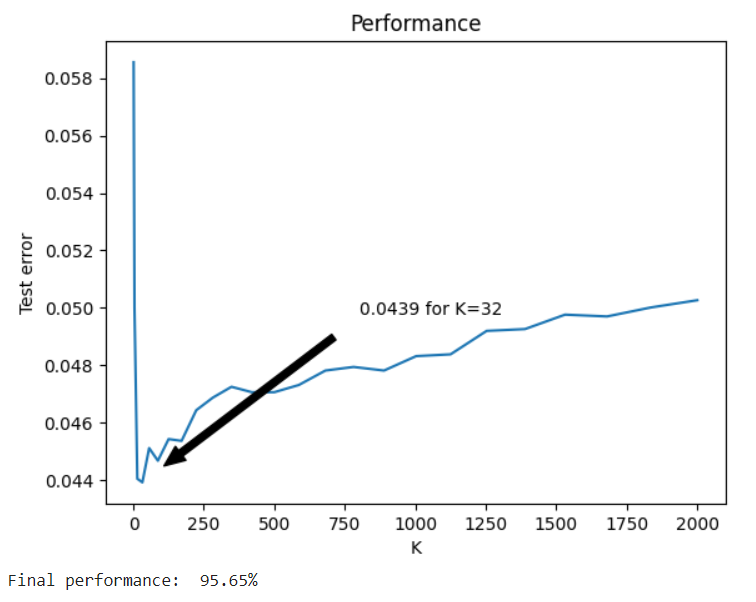
#### Tutti i parametri singolarmente





Il miglior parametro singolo è risultato ‘final\_vote’.

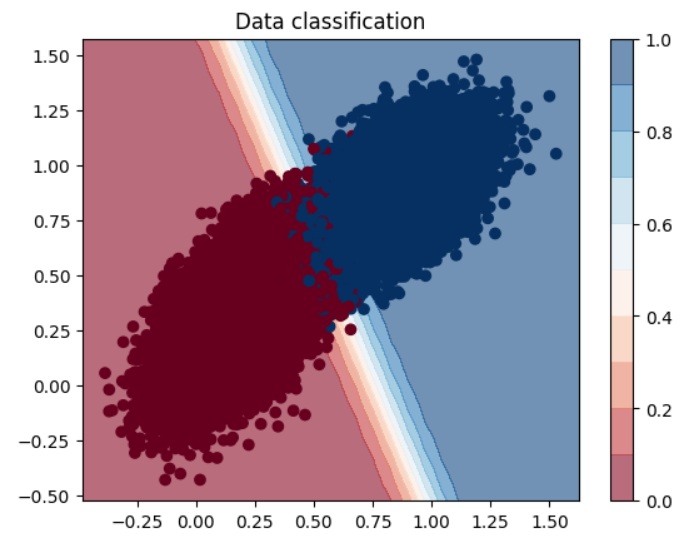
#### Tutti i parametri insieme



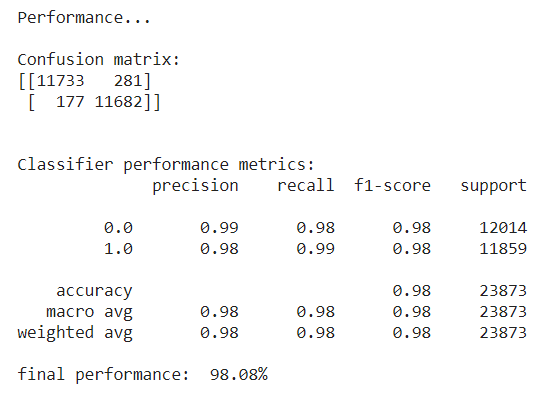
## Logistic Regression

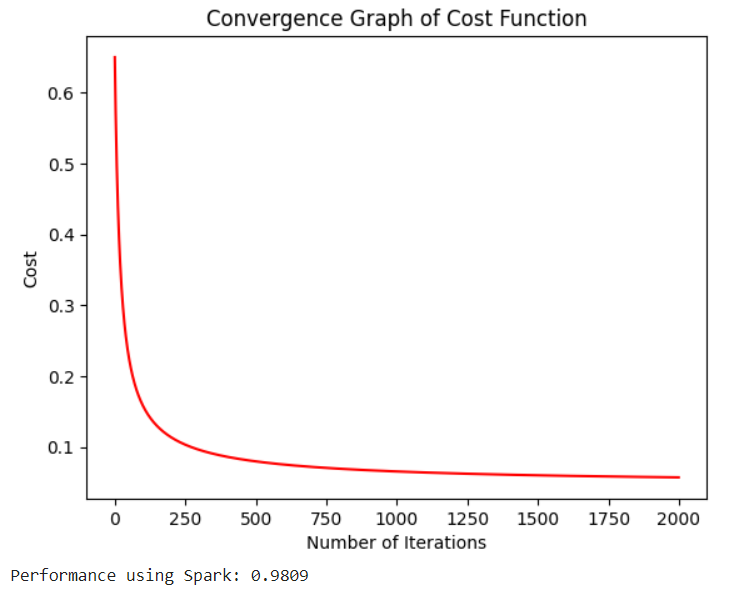
Questo classificatore è caratterizzato dal vettore di parametri che deve essere appreso attraverso la minimizzazione di una certa funzione di costo, e l’algoritmo di apprendimento utilizzato è il ‘Gradient Descent’.

### Dataset sintetico



La figura mostra i risultati ottenuti utilizzando la libreria sklearn, i quali sono molto simili a quelli ottenuti tramite Spark. Riconosciamo 10 livelli di probabilità, secondo i quali vengono classificati i vari sample. Queste sono le performance:



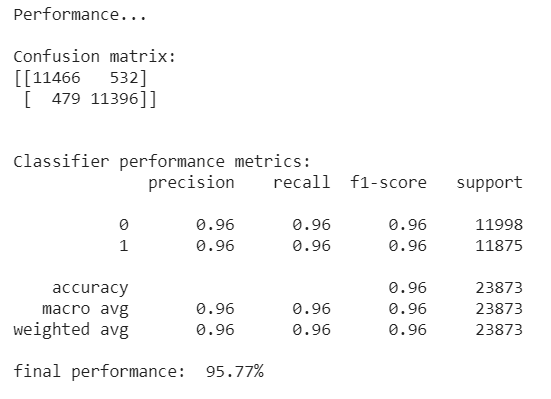


Questo grafico analizza la convergenza della funzione di costo per i parametri ‘learning\_rate’ = 0.9 e ‘momentum’ = 0.001.

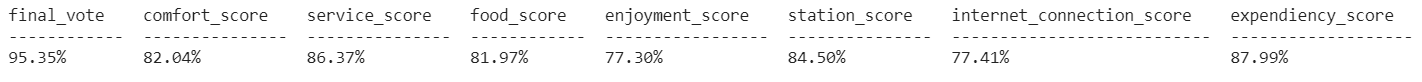
### Dataset reale

Il classificatore Logistic Regression è stato applicato anche al dataset reale utilizzando l’implementazione di libreria. Avendo dimostrato che l’implementazione parallelizzata tramite Spark è corretta e porta gli stessi risultati degli algoritmi presenti in libreria, si è deciso di utilizzare solo la libreria in quanto l’esecuzione tramite Spark avrebbe richiesto una quantità di tempo non permissva.

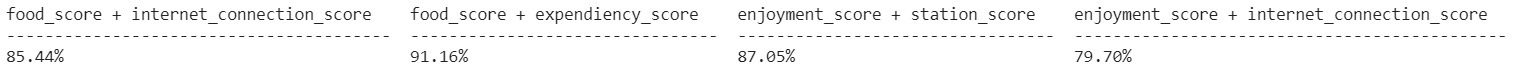
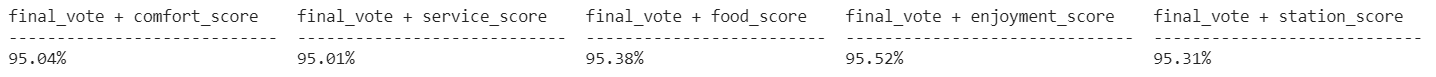
#### ‘final\_vote’ e ‘expendiency\_score’



#### Tutti i parametri singolarmente

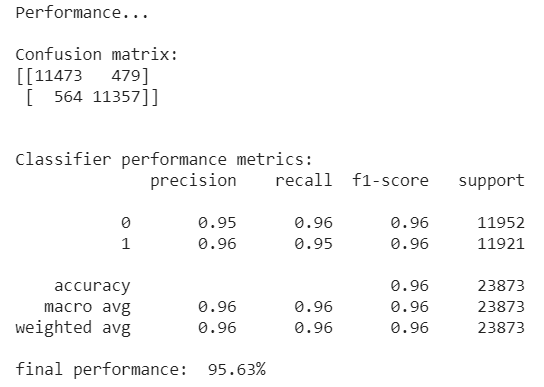


#### Tutte le coppie di parametri



Notiamo che le migliori coppie sono quelle contenenti ‘final\_vote’ come parametro, e forniscono tutte performances molto simili e leggermente migliori rispetto al metodo Naïve Bayes.

#### Tutti i parametri insieme



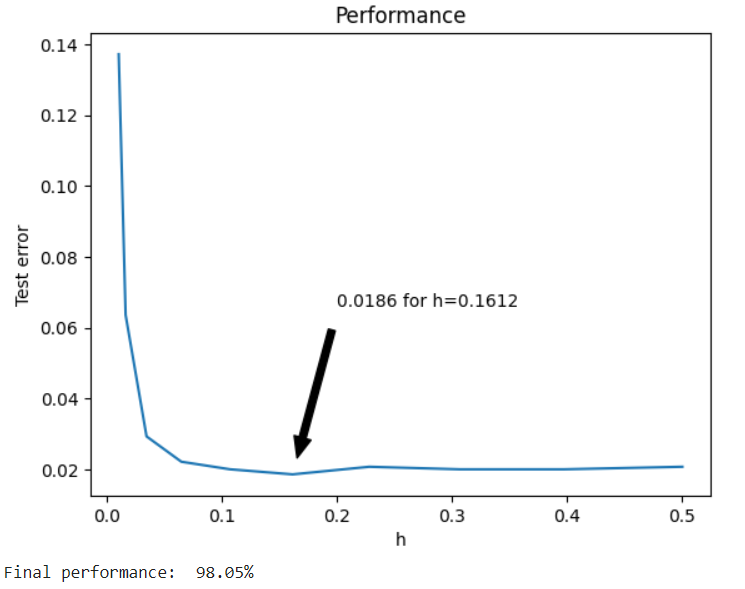
Anche in questo caso otteniamo una performance praticamente uguale.

## Naïve Kernel

Questo metodo prevede la regolazione dell’iperparametro h, ossia la soglia di distanza che identifica l’intorno del punto x da considerare, dove x è un generico sample del dataset. Essendo un’operazione molto onerosa dal punto di vista computazionale, e dovendola eseguire più volte per le diverse combinazioni di parametri, è stato scelto di testare 6 valori di tale iperparametro, scelti in un intervallo variabile a seconda del caso. Ovviamente il dataset è stato nuovamente partizionato in tre, come nel caso del KNN.

### Dataset sintetico

Anche in questo caso si è deciso di effettuare alcune verifiche preliminari con il dataset sintetico. I soliti due parametri ‘final\_vote’ e ‘expendiency\_score’ sono usati come predittori, il numero di sample è 7000, mentre sono stati testati (solo per questo caso) 10 valori per h, nel range [0.01,0.5]:

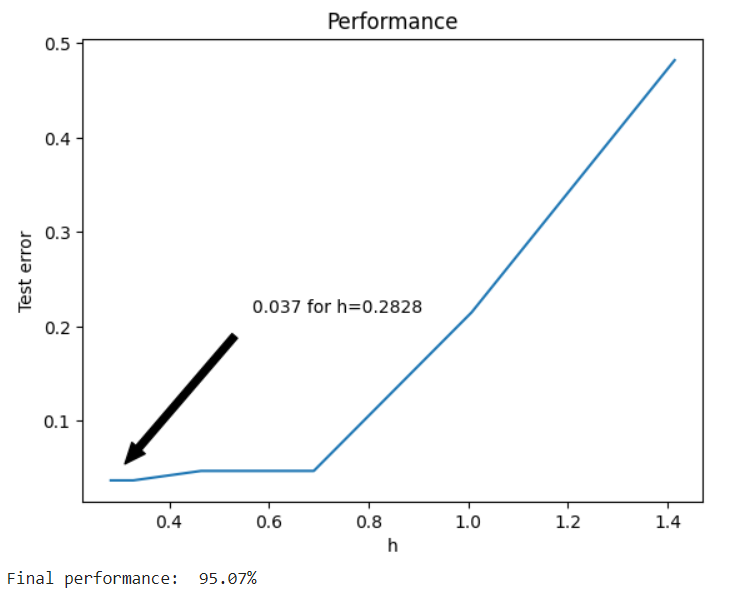


Questo grafico mostra l’MSE al variare del parametro h. I risultati sono simili a quelli visti per gli altri metodi.

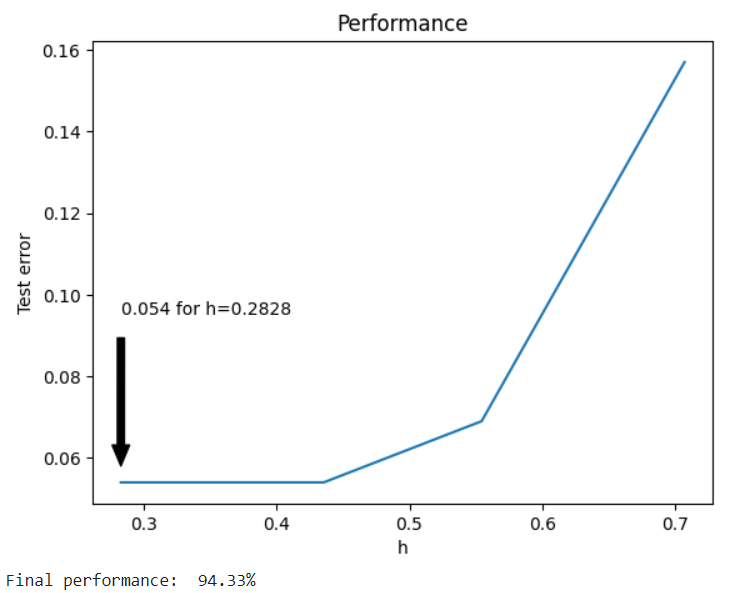
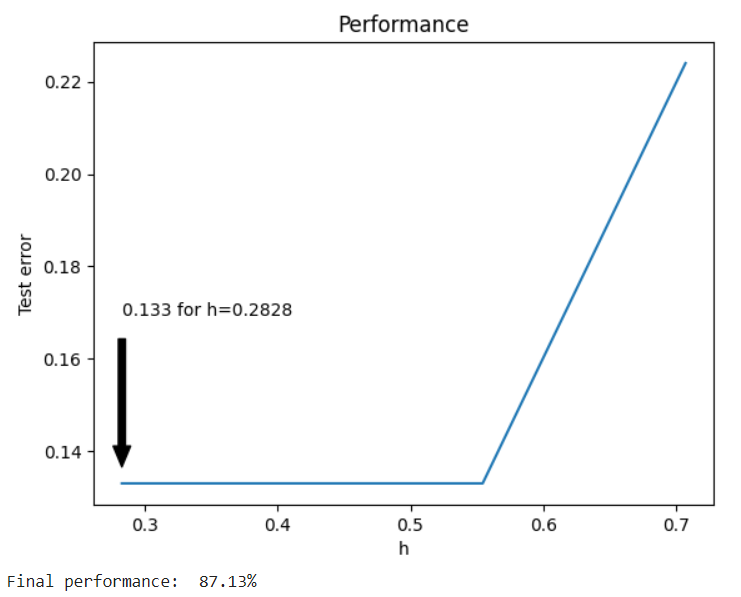
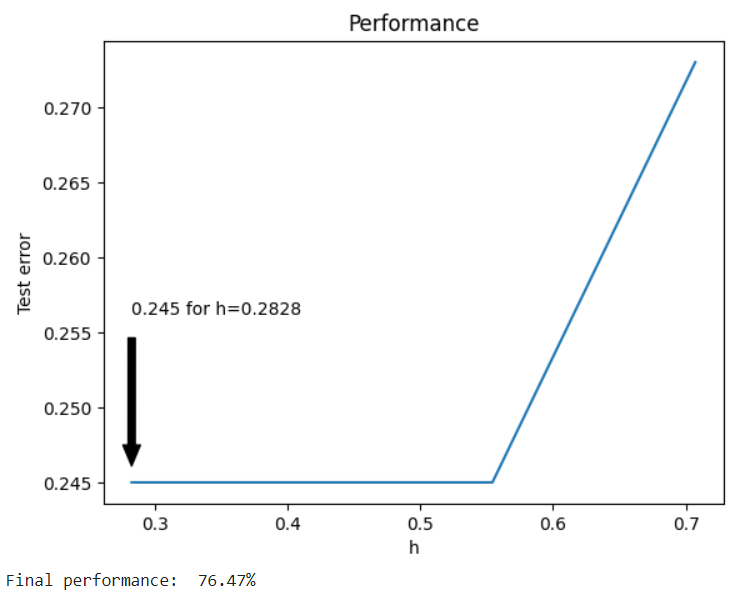
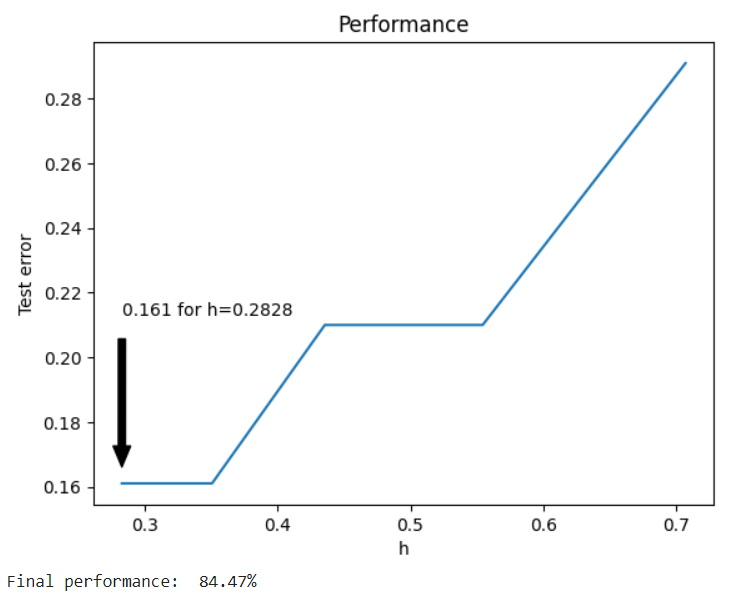
### Dataset reale

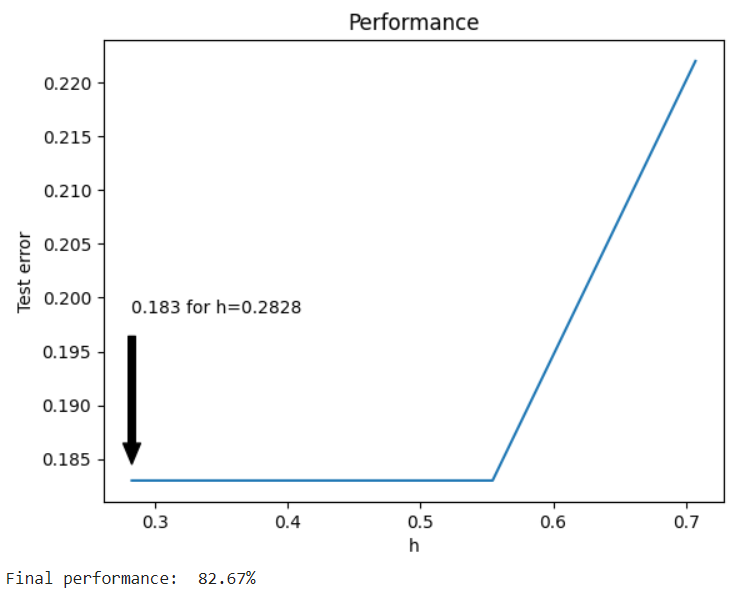
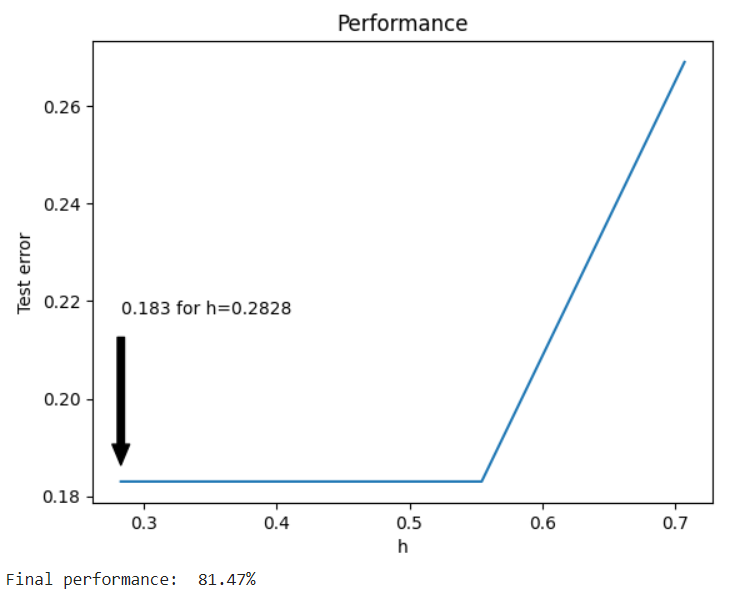
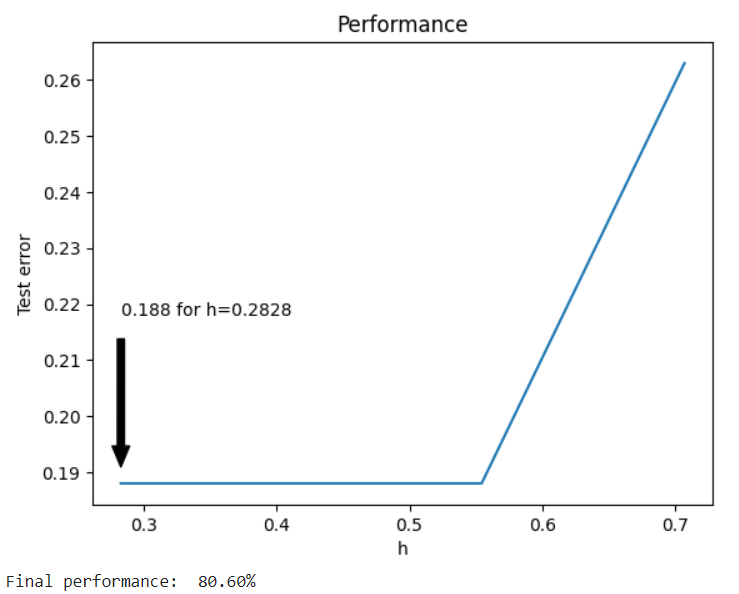
Si è optato qui per un dataset ridotto (5000 sample), per le ragioni di complessità computazionale di cui sopra.

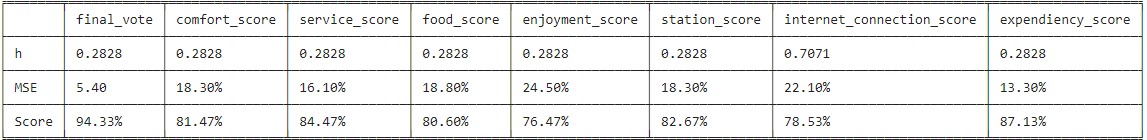
#### ‘final\_vote’ e ‘expendiency\_score’



#### Tutti i parametri singolarmente

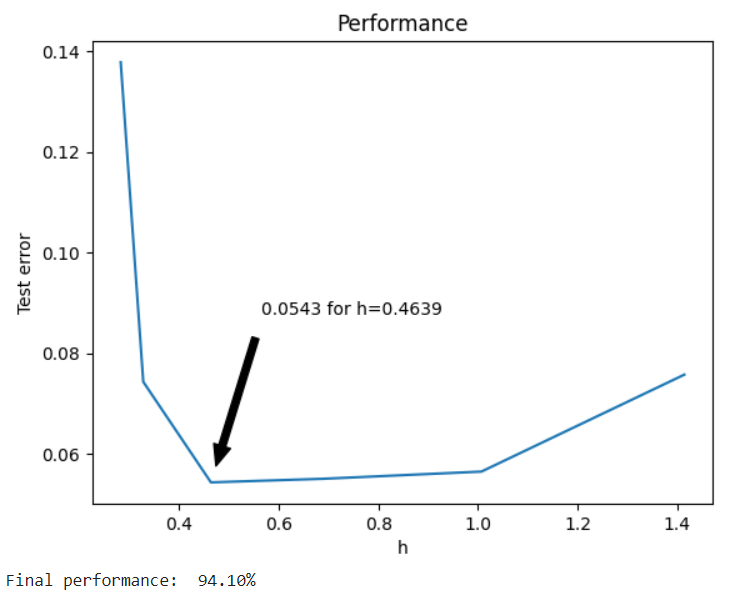






Le performances finali indicano ottimi risultati per ‘final\_vote’, che è il miglior parametro, seguito da ‘expendiency\_score’ e ‘service\_score’.

#### Tutti i parametri insieme



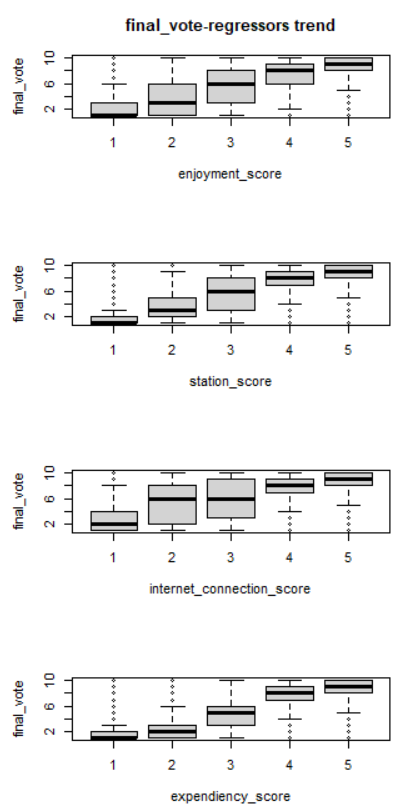
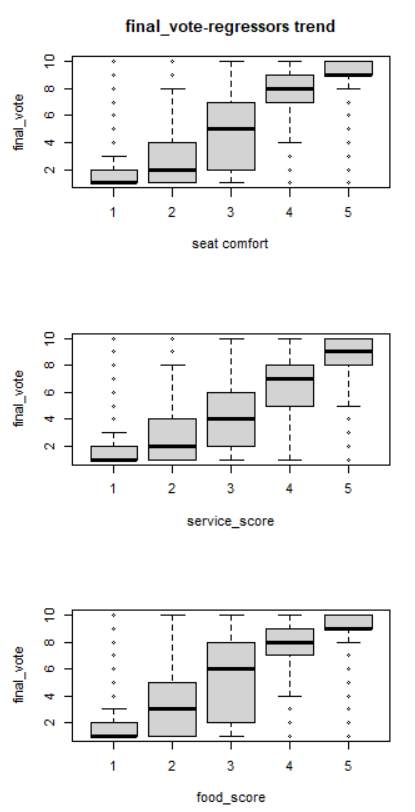
# RStudio

Per questa sezione, è stato utilizzato l’ambiente RStudio con il linguaggio R, al fine di applicare i metodi di regressione analizzati durante il corso.  
A differenza della sezione di Google Colab, in questa è stata scelta come variabile dipendente ‘final\_vote’, mentre come predittori abbiamo ‘approved’, ‘comfort\_score’, ‘service\_score’, ‘food\_score’, ‘enjoyment\_score’, ‘station\_score’, ‘internet\_connection\_score’, ‘expendiency\_score’.

## Regressione Multipla

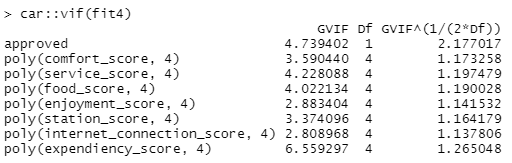
Il tipo di regressione analizzata è la ‘regressione dei minimi quadrati’, che ha come obiettivo quello di trovare la miglior retta interpolatrice dei punti del piano.

Come primo passo, valutiamo la tendenza tra la risposta del sistema da noi cercata (‘final\_vote’) e tutti i predittori scelti.



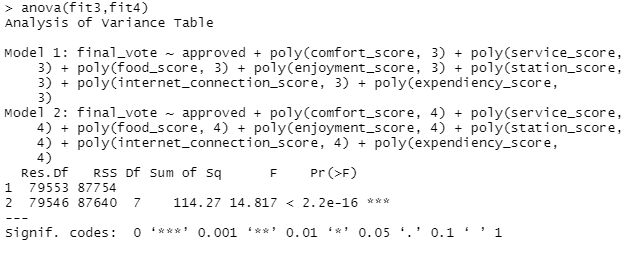
Notiamo che la tendenza tra i regressori e ‘final\_vote’ non è, in generale, lineare; per tale motivo, sono stati eseguiti altri tipi di trasformazioni non-lineari dei predittori presi in esame, in particolare quella quadratica, cubica, di ordine 4 e logaritmica. Per ogni predittore, innanzitutto, valutiamo se esso sia significativo o meno per la stima della variabile dipendente; così otteniamo, per ognuno di essi, la stima dei coefficienti, lo standard error, il t-value e il p-value.

In seguito, viene fatta un’analisi sulla collinearità presente nel modello in questione:

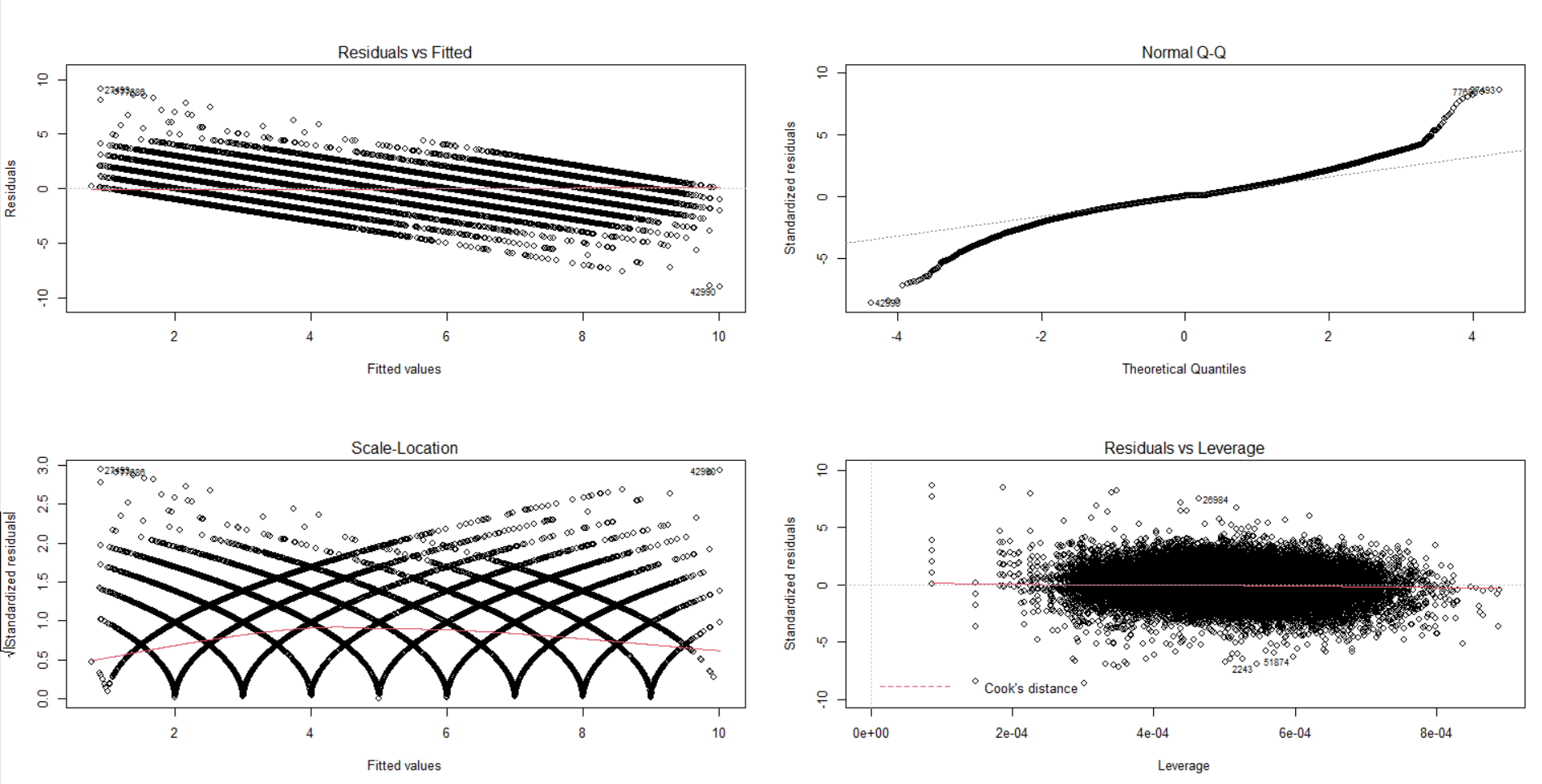


I valori del VIF non superano mai 5, tranne ‘expendiency\_score’, ma comunque nessuno supera 10, la soglia massima per la collinearità.

Confrontiamo i diversi modelli anche in termini di RSS e capiamo quale sia quello migliore per l’interpretazione dei dati. Al termine di tutto il calcolo, si è constatato che il modello polinomiale di ordine 4 ha prodotto i risultati migliori:

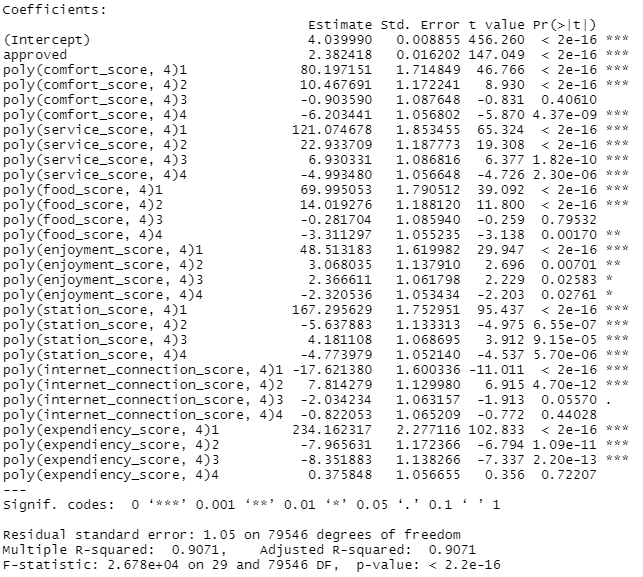


Possiamo vedere questa ottimalità del modello di ordine 4 anche dai seguenti grafici forniti dalla libreria di R:



* In alto a sx: indica i residui in funzione di Y^. Ogni linea rappresenta un valore diverso di ‘final\_vote’ (vanno da 1 a 10, quindi sono discreti);
* In alto a dx: indica se i residui sono approssimabili a una distribuzione normale; in tal caso vediamo che, approssimativamente, la linea calcolata giace in gran parte sulla retta ideale, dunque i residui standardizzati sono riconducibili a una distribuzione normale;
* In basso a sx: indica i valori di Y^ in funzione dei residui standardizzati. Utile per identificare ‘outliers’;
* In basso a dx: indica la ‘leverage’ in funzione dei residui standardizzati.

Infine, possiamo vedere dalla seguente figura i parametri più o meno significativi per la stima di ‘final\_vote’:



## Metodi di ricampionamento

Abbiamo usato K-fold Cross Validation e Bootstrap come metodi di ricampionamento statistico.  
I suddetti metodi permettono di effettuare un *fit* su un *training set* e valutare le performance su un *training set*. L’obiettivo è trovare il modello ottimale, che ottenga buone performance sul training e sul test set.

Come indice delle prestazioni dei modelli utilizzeremo l’*MSE,* o *Mean Square Error* (errore quadratico medio). Esso rappresenta la differenza tra i valori dei dati osservati e i valori dei dati stimati.

In questo caso è necessario discutere del *trade-off* tra modello semplice e complesso:  
Più è semplice il modello, meno riuscirà a spiegare le variazioni dei dati sul training set, e otterrà performance non soddisfacenti sia sul training sia sul test set. La complessità del modello va quindi aumentata, ma non si dovrebbe aumentarla arbitrariamente: al crescere della complessità del modello, esso si adatterà (fitting) ai dati del training set, spiegando apparentemente molto bene la variazione dei dati del training. Il modello però sarà adattato troppo specificamente a tali dati del training set, e, nonostante ottime performance sul training set, avrà performance molto peggiori sul test set. Questo fenomeno è chiamato *overfitting*.

### Validation set

Il validation set approach non è un metodo di ricampionamento statistico, ma è necessario citarlo in questa sezione al fine di spiegare come vengono utilizzati i dati per poi eseguire il fit e valutare le performance di ogni modello: i dati sono divisi in due parti:

* un *training set*, per addestrare il modello
* un *test set*, per valutare le performance del modello

Abbiamo deciso di dividere il dataset in parti uguali, seguendo la divisione 50% training set e 50% test set.  
Sono di seguito riportati i valori di MSE per ciascuno dei modelli considerati:

|  |  |
| --- | --- |
| **Transformation** | **MSE** |
| Linear | 1.111269 |
| Polynomial of 2nd order | 1.095797 |
| Polynomial of 3rd order | 1.094223 |
| Polynomial of 4th order | 1.094223 |
| Logarithmic | 1.241137 |

### K-Fold Cross Validation

Il validation set approach ha però alcuni svantaggi, primo tra tutti che potrebbe catturare e quindi essere influenzato da un eventuale *bias* sui dati. Il K-fold CV supera questo problema, allenando il modello e testandolo su ciascuno dei sottogruppi, detti *folds* in cui vengono divisi i dati.  
I dati vengono appunto divisi in un numero K di folds. K-1 folds vengono usati per il training set e 1 fold per il test set. Questo processo viene ripetuto K volte, una volta per ciascuno dei fold usati come test set.

Le prestazioni del modello su tutti i folds dovrebbero essere molto simili tra loro, e le performance finali del modello vengono calcolate in base all’errore medio in tutte le iterazioni. La seguente tabella cattura tutti i valori di MSE trovati con il K-fold CV:

|  |  |
| --- | --- |
| **Transformation** | **MSE** |
| Linear | 1.120782 |
| Polynomial of 2nd order | 1.105092 |
| Polynomial of 3rd order | 1.103605 |
| Polynomial of 4th order | 1.102377 |

### Bootstrap

L’altro metodo di ricampionamento utilizzato è il bootstrap. Esso è un metodo molto flessibile e potente che può essere utilizzato per quantificare l’incertezza associata ad uno stimatore. La stima generale del bootstrap è la media delle stime ottenute da ciascuna stima del campione bootstrap, e avrà una varianza generalmente inferiore rispetto ad un meccanismo di suddivisione dei dati in train e test set, come per validation set o K-fold VC.

Nel progetto, utilizziamo il bootstrap per esaminare la discrepanza tra la stima dello *standard error* dei coefficienti effettuata dall’approccio e quella effettuata dai modelli di regressione multipla di ogni trasformazione.

La misura di affidabilità del metodo è stata valutata attraverso un numero variabile di valori, che dipende dal grado del polinomio esaminato. Di seguito sono mostrati gli standard error relativi ai coefficienti della trasformazione lineare.

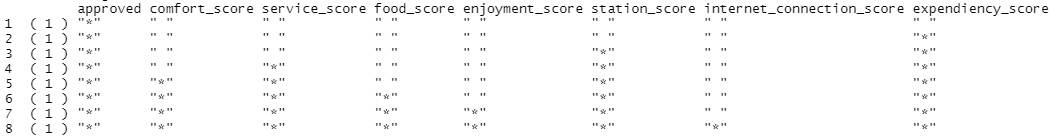
|  |  |
| --- | --- |
| **Coefficients** | **Standard Error Difference** |
| intercepts | 0.0004565354 |
| approved | -0.006183459 |
| comfort\_score | -0.0006145307 |
| service\_score | -0.0009786729 |
| food\_score | -0.0003997568 |
| enjoyment\_score | -0.0005182993 |
| internet\_connection\_score | -0.0012441211 |
| station\_score | -0.0002897655 |
| expendiency\_score | -0.001459983 |

## Subset Selection

Con i metodi seguenti viene scelto un sottoinsieme di predittori da utilizzare nella costruzione del modello.

### Best Subset Selection

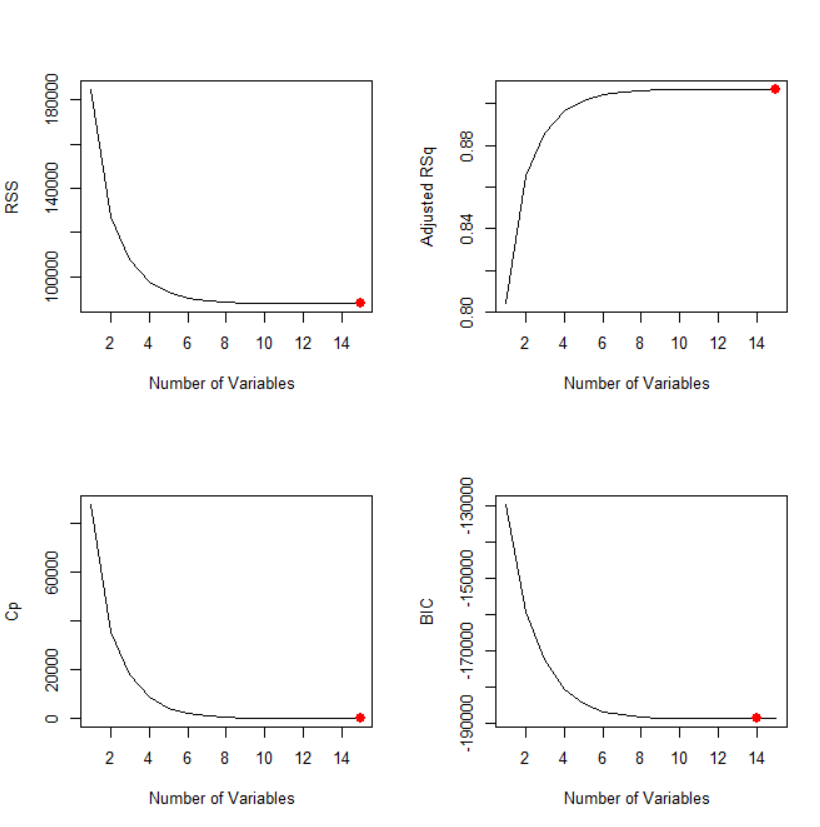
Vediamo nel seguente grafico il miglior set di variabili per ogni possibile dimensione del modello, nel caso di una trasformazione lineare:



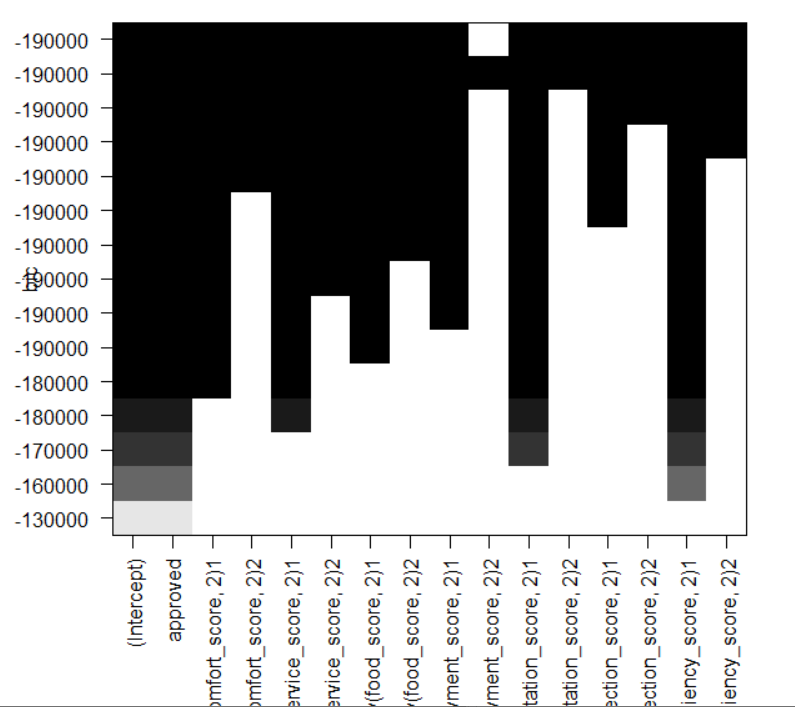
Un asterisco indica che il parametro è inserito nel modello corrispondente.

A noi, tuttavia, interessano di più i casi non lineari, per i motivi di cui sopra. Pertanto, abbiamo testato questi modelli usando come criteri di selezione diversi parametri, come RSS, BIC, Cp di Mallow e Adjusted R2.

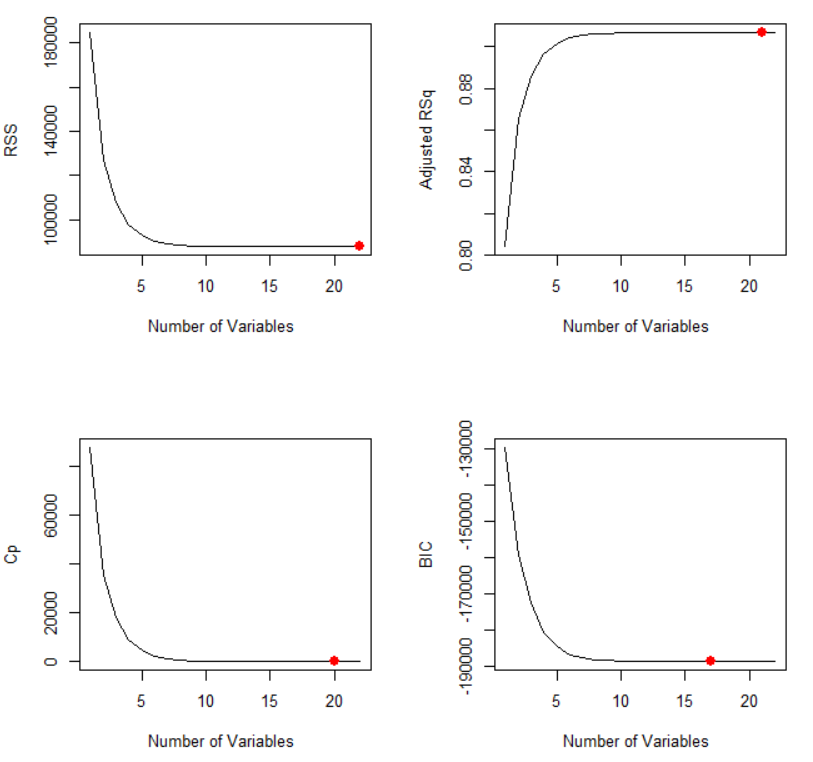
#### Trasformazione quadratica



Vediamo, partendo dall’alto e verso destra, il numero di predittori ottimo per RSS (15), Adjusted R2(15), Cp (15) e BIC (14). Il BIC è quello che impiega meno predittori, pertanto utilizziamo quello per costruire il modello in questo caso. Di seguito vediamo le variabili utilizzate per diversi valori del BIC (colore nero = variabile compresa; colore bianco = variabile esclusa; il sottoinsieme ottimo è quello della prima riga):



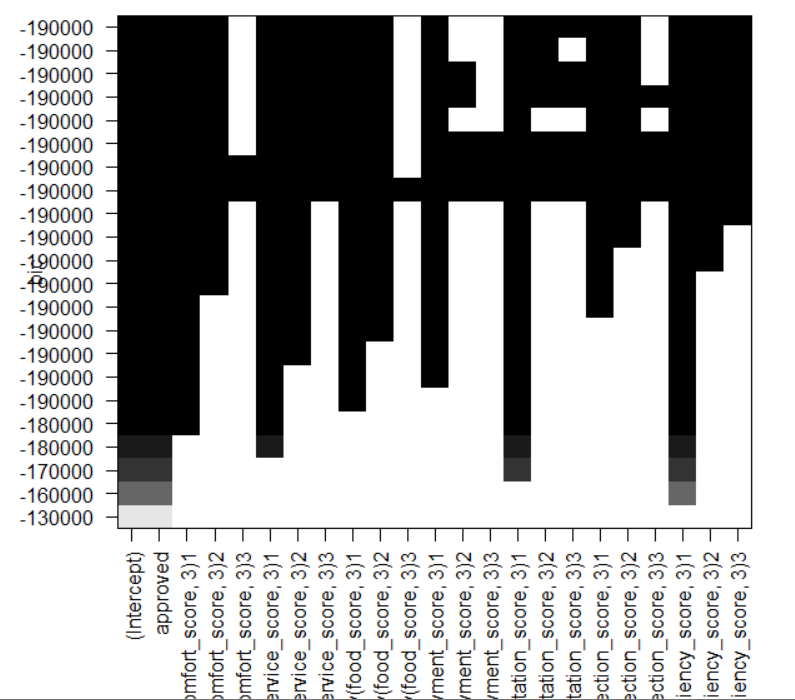
#### Trasformazione cubica



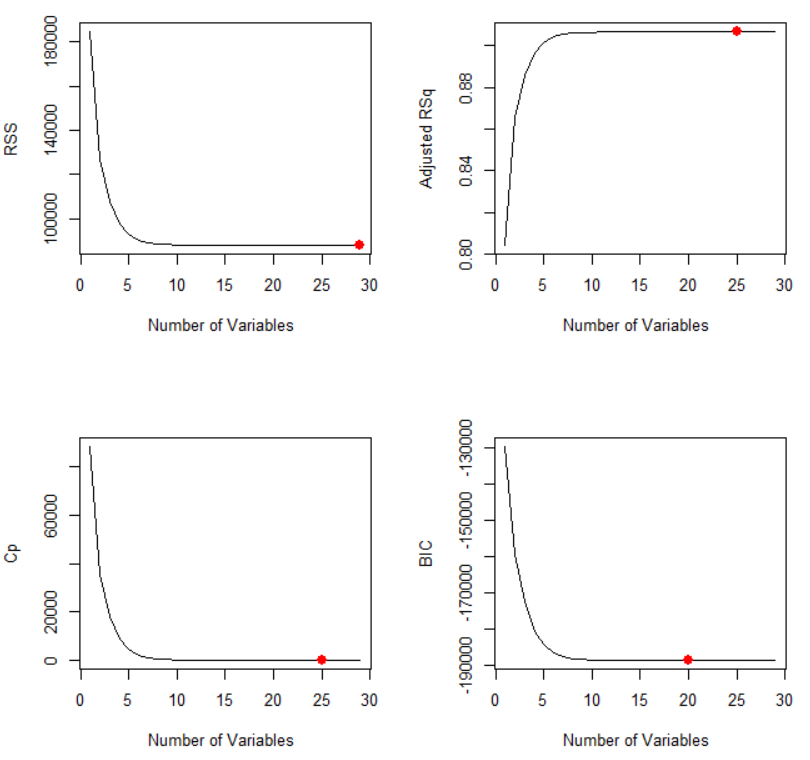
Stavolta abbiamo i seguenti risultati:

* RSS = 22 predittori
* Adjusted R2 = 21 predittori
* Cp = 20 predittori
* BIC = 17 predittori

Il miglior criterio è ancora il BIC:



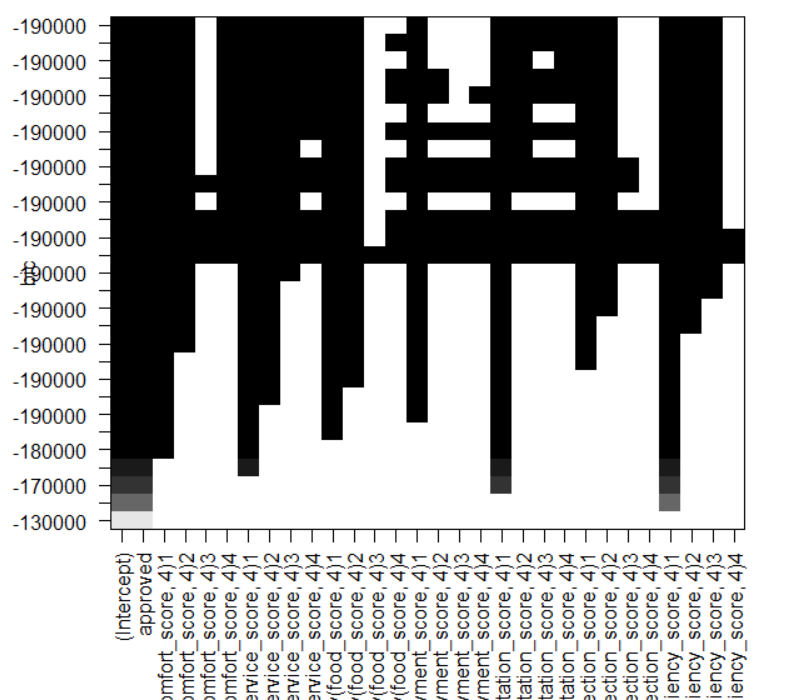
#### Trasformazione di ordine 4



In tal caso:

* RSS = 29 predittori
* Adjusted R2 = 25 predittori
* Cp = 25 predittori
* BIC = 20 predittori

Ancora una volta, il BIC è il criterio ottimo:



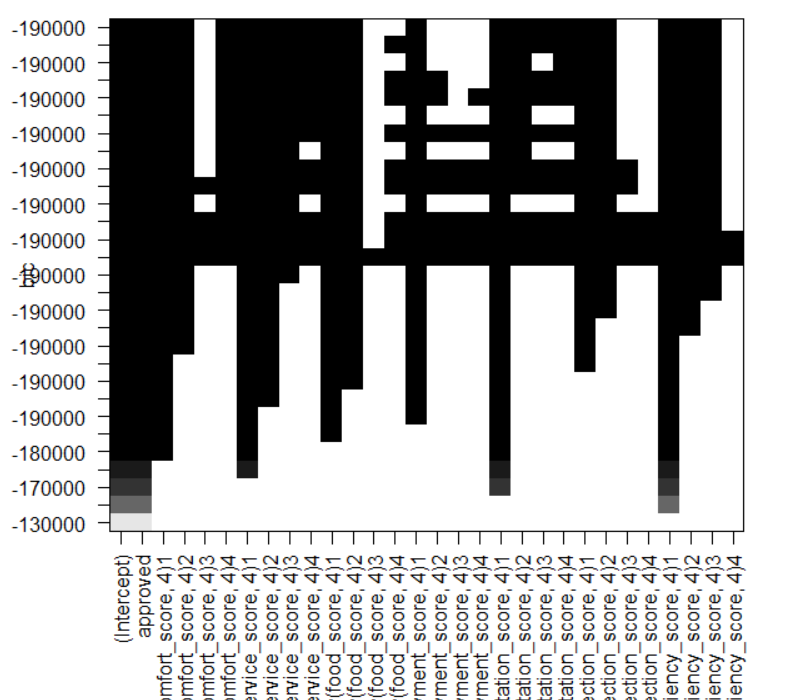
### Stepwise Selection

È un metodo per adattare i modelli di regressione in cui la scelta delle variabili predittive viene effettuata mediante una procedura automatica. Ad ogni passaggio viene considerata una variabile per l’aggiunta (‘forward’) o la rimozione (‘backward’) dall’insieme delle variabili.

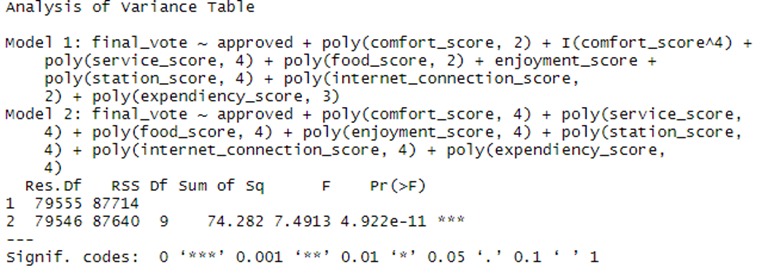
I risultati ottenuti nell’ordine 4 sono stati uguali a quelli sella ‘Best Subset Selection’. Per valutare la prestazione del modello migliore, trovato da queste tre tecniche, è stato utilizzato il ‘Validation Set Approach’, stimando l’MSE sul test set dopo il fitting; osserviamo i seguenti risultati:

* Lineare -> 8 predittori ; MSE: 1.106
* Poly-2 -> 14 predittori ; MSE: 1.090
* Poly-3 -> 22 predittori ; MSE: 1.089
* Poly-4 -> 28 predittori ; MSE: 1.087

L’MSE minimo si ottiene, come vediamo, da un modello a 28 predittori su 29. Detto ciò, si è valutata la trasformazione di ordine 4, e usando come criterio il parametro BIC, in quanto i predittori usati da esso sono minimi (20):



Grazie ad una successiva analisi, vediamo che un numero di predittori superiore a 20 non migliora significativamente le prestazioni, pertanto risulta preferibile scegliere un modello più semplice, al fine di evitare problemi di overfitting in fase di training. Tale modello è stato comparato con quello completo, ed è risultato che lo scarto tra i due RSS è molto basso:



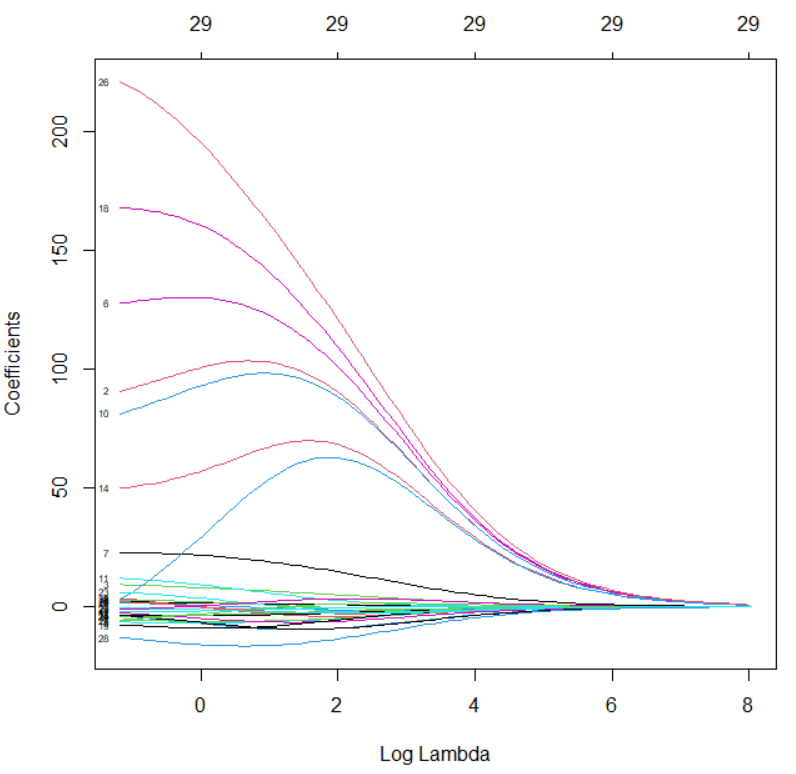
## Metodi di regolarizzazione

Puntano a ridurre la varianza del modello, riducendone i coefficienti, senza una diminuzione significativa del bias. Abbiamo utilizzato le due tecniche Ridge e LASSO. Inoltre, per il valore di λ, abbiamo scelto di utilizzare il metodo di ‘cross-validation’.

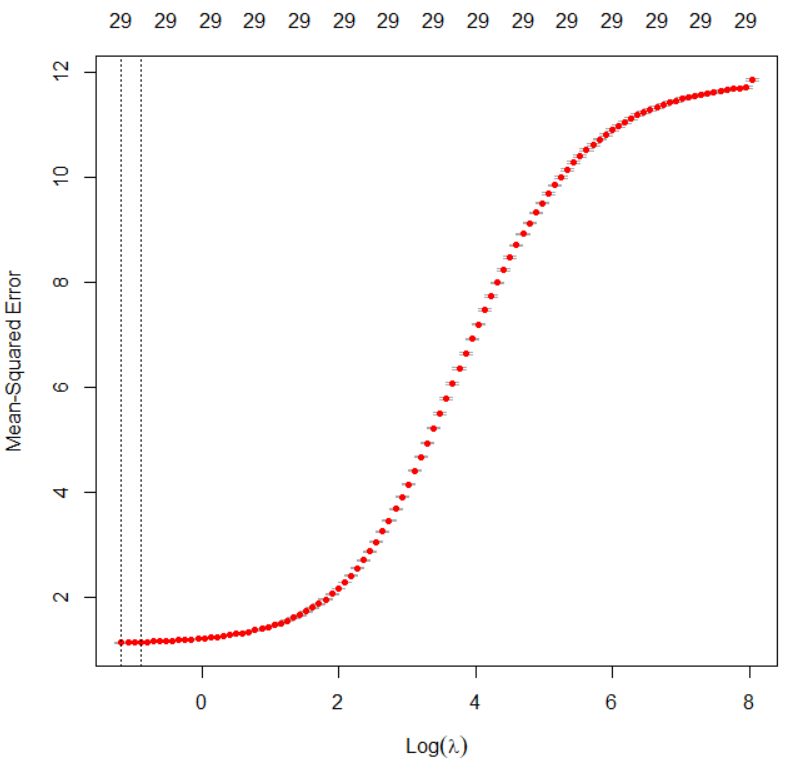
### Ridge

Mostriamo i risultati ottenuti nel caso della trasformazione di ordine 4.

Il seguente grafico mostra il valore dei diversi coefficienti all’aumentare del parametro λ. Come ci si aspetta, essi tendono a 0 per λ molto grandi:



Inoltre, mostriamo il grafico che lega l’MSE a λ, ed anche in questo caso, naturalmente, si osserva un aumento del primo all’aumentare del secondo:

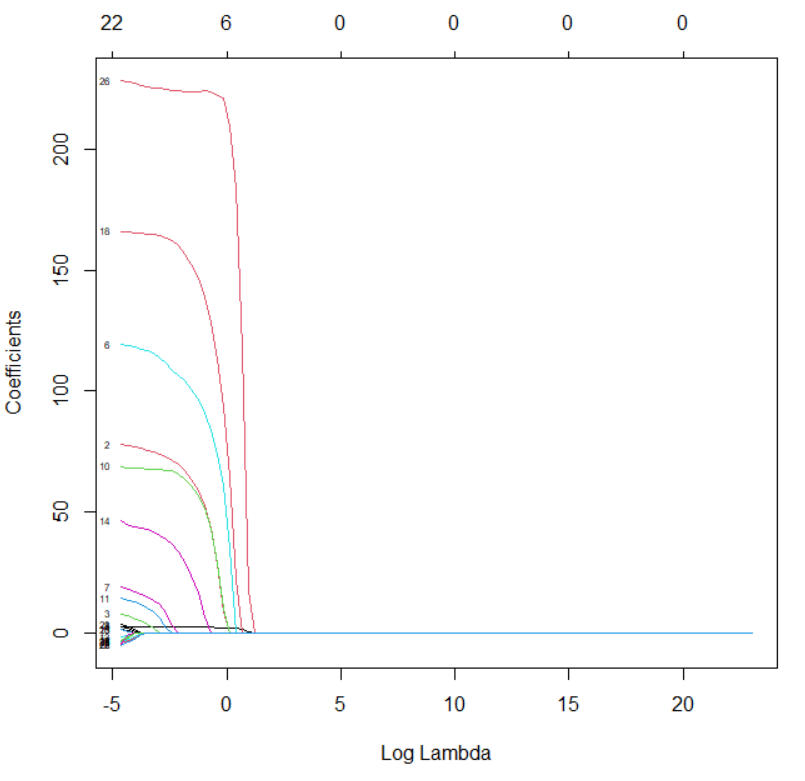


Il metodo in esame, tuttavia, non è particolarmente utile nel nostro caso: la regressione Ridge incontra risultati migliori in dataset ad alta dimensionalità, in cui i predittori sono maggiori delle osservazioni. Questo non è vero nel nostro dataset.

### LASSO

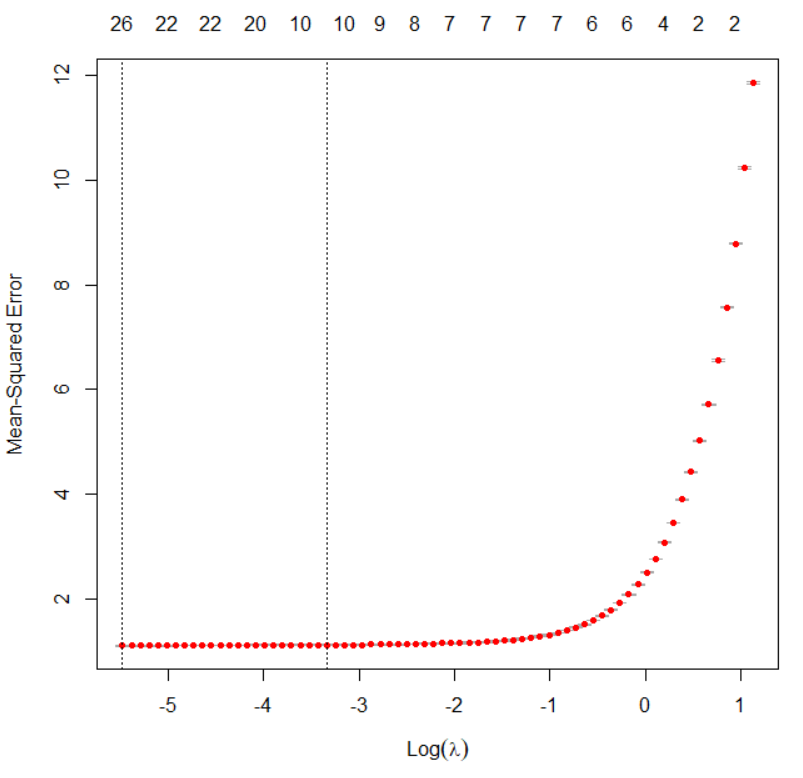
L’analisi effettuata qui è analoga al caso della Ridge, e mostriamo anche qui i risultati della trasformazione di ordine 4.

Il seguente grafico mostra i valori dei coefficienti in relazione a λ:



Come ci si aspetta da questo metodo, i coefficienti vengono portati a 0 ognuno con una diversa funzione, in base a λ.

Nel seguente grafico vediamo l’MSE in relazione a λ:



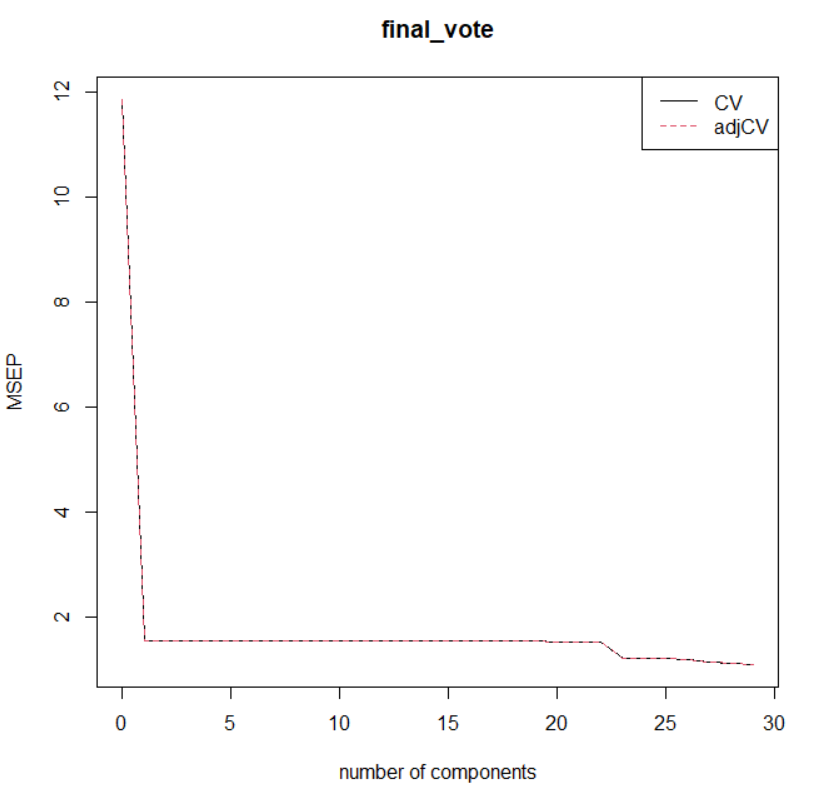
Risulta che il metodo LASSO, applicato a una trasformazione di ordine 4, produce un modello con 7 coefficienti uguali a 0. Questo dimostra che il metodo LASSO, a differenza del Ridge, produce in questo caso un modello molto competitivo e comparabile con i modelli senza regolarizzazione, con il vantaggio di eliminare alcuni coefficienti e quindi di rendere il modello più interpretabile.

## PCR

Questa tecnica, che usa sempre tutti i predittori, risulta vantaggiosa quando poche componenti (ne vengono scelte in numero di M, ognuna da p predittori) riescono a riassumere gran parte della variabilità totale nei p predittori. Il parametro M viene scelto con ‘cross-validation’.

Anche in questo caso, come per quelli precedenti, abbiamo effettuato analisi su vari modelli, in particolare quello lineare, di grado 2, 3 e 4. I risultati di quest’ultimo vengono mostrati a titolo di esempio.

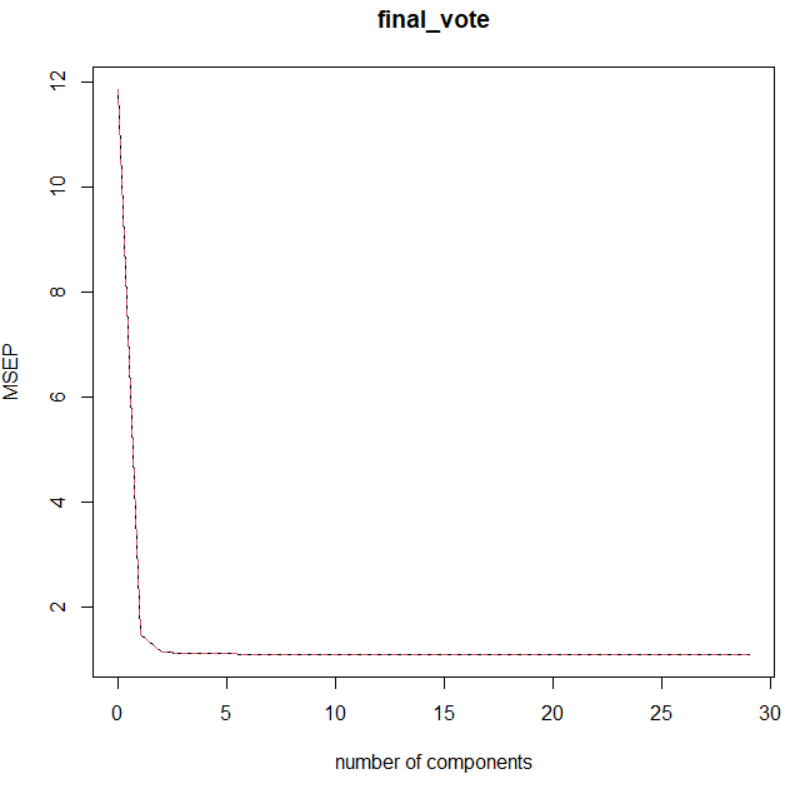
Mostriamo inizialmente l’MSE per un numero diverso di componenti:



L’MSE minimo si ottiene considerando tutti e 29 i predittori. Successivamente, è stata effettuata una regressione con le componenti principali sul training set ed è stato testato il modello ottenuto su un test set. I risultati sul nuovo MSE e RMSE sono stati molto simili a quelli della regressione semplice, in quanto non c’è alcun beneficio nel ridurre il numero di componenti da considerare. Utilizzando questo modello, la percentuale di varianza spiegata su ‘final\_vote’ risulta essere il 90.71%; questa tecnica non è risultata efficace, essendo simile nei risultati alla regressione semplice.

## PLS

Questo metodo è stato applicato per tutte le trasformazioni viste finora. Sono state realizzate delle rappresentazioni che tenessero conto delle variazioni dei singoli parametri presi in esame. Tali risultati vengono mostrati nel seguito, anche questa volta solo quelli ottenuti per l’ordine 4. Nella seguente tabella vediamo l’MSE in funzione del numero di componenti:



Il numero di componenti che minimizza l’MSE è 12 su 29. Per le altre trasformazioni abbiamo:

* Lineare -> 8 su 8
* Poly-2 -> 14 su 15
* Poly-3 -> 15 su 22

Eseguendo il fitting e un successivo test, otteniamo i seguenti risultati:

* Lineare -> 7 comp. -> MSE: 1.111
* Poly-2 -> 9 comp. -> MSE: 1.095
* Poly-3 -> 11 comp. -> MSE: 1.094
* Poly-4 -> 10 comp. -> MSE: 1.093

Il vantaggio della PLS è quello di riuscire a spiegare una maggior varianza della Y sfruttando un minor numero di componenti rispetto alla PCR.