UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione,
Ingegneria Elettrica e Matematica Applicata

SDA Project

Student:
Michele Barbella
0622701341

Professors:
Fabio Postiglione
Vincenzo Matta

Academic Year 2019/2020

Indice

Capitolo 1

Introduzione

Per la realizzazione del progetto di Statistical Data Analysis è stata richiesta l'applicazione delle diverse tecniche di analisi dei dati studiate durante il corso. Sono stati esplorati e confrontati i diversi metodi, motivando la scelta sulla base di considerazioni teoriche ed evidenze sperimentali.
Il progetto si compone di due parti: la prima affronta i diversi approcci sulla regressione, mentre la seconda si concentra sulle tecniche di classificazione. La realizzazione dei due task ha comportato l'utilizzo di due differenti dataset opportunamente scelti durante la prima fase del progetto. La scelta di utilizzare due differenti dataset è dovuta a molteplici fattori che saranno spiegati nelle sezioni successive. In ciascuna delle due parti, vengono mostrati grafici diagnostici e tabelle riassuntive per valutare la bontà delle scelte attuate.
La fase iniziale del progetto è stata caratterizzata dallo studio di diversi dataset. Considerando le indicazioni di progetto, che vedono l'applicazione sia di metodologie di regressione che di classificazione, in un primo momento mi sono concentrato sul ricercare un dataset che permettesse di affrontare entrambe le tecniche. Dopo diverse analisi, però, sia per questioni inerenti ad una buona applicazione delle varie tecniche, sia per motivi legati al tempo a disposizione, ho optato per l'utilizzo di due differenti dataset.

1.1 Dataset regressione

Il contenuto del dataset riguarda informazioni su 506 film. I campi di questo dataset sono: Marketing expense, Production expense, Multiplex coverage, Budget, Movie length, Lead Actor Rating, Lead Actress rating, Director rating, Producer rating, Critic rating, Trailer views, 3D available, Time taken, Twitter hastags, Genre, Avg age actors, Num multiplex, Collection. Il dataset è stato scaricato dal sito `https://www.kaggle.com/`, in cui sono contenuti un gran numero di dataset a servizio di studenti e ricercatori per l'analisi empirica dei dati. In questa collezione `https://www.kaggle.com/datasets/balakrishcodes/others?select=Movie_regression.xls` sono presenti 15 dataset, alcuni adatti alla regressione e altri alla classificazione. L'idea originale è stata quella di usare i due dataset relativi ai film per applicare le tecniche di regressione e di classificazione; infatti il dataset relativo alla classificazione differisce dall'altro solo per una colonna aggiuntiva relativa alla vittoria dell'oscar. Ma poichè dalle analisi effettuate i risultati non erano soddisfacenti, ho deciso di usarne uno diverso per la classificazione.

1.1.1 Descrizione degli attributi

- *Marketing expense:* tutte le spese sostenute dalla produzione per pubblicizzare e vendere il film. Pubblicità, campagne, eventi promozionali, sponsorizzazione delle celebrità e ricerche di mercato sono tutti inclusi nel costo di marketing.
- *Production expense:* le spese per produrre il film in dollari.
- *Multiplex coverage:* la copertura dei multiplex (singolo edificio che proietta più film contemporaneamente), un numero compreso tra 0 e 1.
- *Budget:* il bilancio del film in dollari.
- *Movie length:* la lunghezza del film in minuti.
- *Lead Actor Rating:* la valutazione dell'attore principale in decimi.
- *Lead Actress rating:* la valutazione dell'attrice principale in decimi.
- *Director rating:* la valutazione del regista in decimi.
- *Producer rating:* la valutazione del produttore in decimi.
- *Critic rating:* la valutazione della critica in decimi.
- *Trailer views:* il numero di volte che è stato visto il trailer.
- *3D available:* la disponibilità della visione in 3D del film.
- *Time taken:* il tempo impiegato per girare il film in minuti.
- *Twitter hastags:* in numero di volte che è stato utilizzato l'hastag del film su Twitter.
- *Genre:* il genere del film.
- *Avg age actors:* l'età media del cast di attori.
- *Num multiplex:* il numero di multiplex in cui è stato distribuito il film.
- *Collection:* un campo di cui non ho trovato informazioni e pertanto non è stato utilizzato nell'analisi.

1.2 Dataset Classificazione

Questo dataset contenente 1372 istanze con 5 attributi, è stato scaricato all'indirizzo `https://www.kaggle.com/datasets/shantanuss/banknote-authentication-uci`.
Il contenuto del dataset riguarda immagini prese da campioni di banconote autentiche e contraffatte. Per la digitalizzazione è stata utilizzata una telecamera industriale solitamente impiegata per l'ispezione delle stampe. Per estrarre le caratteristiche dalle immagini è stato utilizzato lo strumento della trasformata Wavelet, un potente strumento matematico usato in molte applicazioni, come ad esempio nella definizione degli standard JPEG-2000 e MPEG-4.

1.2.1 Descrizione degli attributi

- *variance:* varianza dell'immagine calcolata con la trasformata Wavelet (continua).
- *skewness:* asimmetria dell'immagine calcolata con la trasformata Wavelet (continua).
- *curtosis:* curtosi (proprietà per la quale una curva di distribuzione presenta un addensamento intorno al valore centrale) dell'immagine calcolata con la trasformata Wavelet (continua).
- *entropy:* entropia, una misura statistica della casualità che può essere utilizzata per caratterizzare la texture dell'immagine (continua).
- *class:* classe, valore binario che identifica se la banconota è autentica o contraffatta.

Capitolo 2

Progetto R

In questa sezione vengono affrontati ed esplorati tutti i metodi di regressione, applicando le conoscenze acquisite. Per lo sviluppo di questa parte, è stato utilizzato il software di analisi dei dati RStudio ed, in particolare, una serie di librerie che forniscono le funzioni necessarie allo svolgimento delle operazioni, con lo scopo di stimare un'eventuale relazione esistente tra la variabile dipendente e le variabili indipendenti. Con il susseguirsi dei vari metodi, sono state messe in evidenza le informazioni principali che suggerivano le decisioni sulle ulteriori applicazioni da mettere in atto.
L'obiettivo fissato è stato quello di stimare la variazione dell'attributo Budget, la nostra variabile dipendente, in relazione ai differenti predittori del dataset.

2.1 Preprocessing

La parte iniziale del progetto è stata dedicata alla pre-elaborazione dei dati, fondamentale per una buona analisi. Infatti, i metodi di raccolta dei dati sono spesso controllati in modo impreciso, dando vita a valori fuori range, valori mancanti, ecc. Eseguire una tecnica di regressione o classificazione su dataset che non sono stati attentamente esaminati, può produrre risultati fuorvianti. Pertanto, il controllo sulla rappresentazione e sulla qualità degli stessi è fondamentale prima di eseguire qualsiasi operazione, influendo sul modo in cui vengono interpretati i risultati finali.
Fatta questa premessa, attraverso RStudio, un ambiente di sviluppo utilizzato per l'elaborazione dei dati e l'analisi statistica, sono state effettuate diverse azioni per migliorare la qualità dei dati a disposizione.

2.1.1 Modifiche applicate

Dopo aver convertito il dataset originale in formato csv sono state effettuate diverse operazioni per migliorare la qualità dei dati a disposizione, come si può vedere nel file *1_preprocessing.R*.

Duplicati e valori mancanti

Inizialmente ho verificato che non fossero presenti elementi duplicati, in seguito ho deciso di rimuovere quegli attributi che a mio avviso non erano decisivi al fine dell'analisi quali: *Multiplex coverage, Movie_length, 3D_available, Genre, Num_multiplex, Collection.* Successivamente ho controllato se vi fossero valori mancanti. Dalla ricerca sono risultati 12 elementi mancanti relativi all'attributo *Time_taken*, pertanto ho provveduto a rimuore le righe relative a questi valori.
A questo punto il dataset su cui andremo a fare le operazioni è composto da 494 istanze.

Identificazione e rimozione Outliers

Gli outliers sono delle osservazioni anomale, ovvero che differiscono significativamente dalle altre. Possono essere dovuti a vari fattori tra cui un errore sistematico che si è verificato nella raccolta dei dati e possono influenzare notevolmente le stime e le predizioni di un modello.
Per identificarli si è scelto l'approccio univariato, vale a dire che per ogni feature del dataset è stato costruito un boxplot e si considerano outliers tutte le osservazioni che giacciono fuori da 1.5 * IQR, dove IQR è il range interquartile, ovvero la differenza tra il primo e il terzo quartile.
Nella seguente figura vediamo i grafici degli outliers per ciascun attributo.

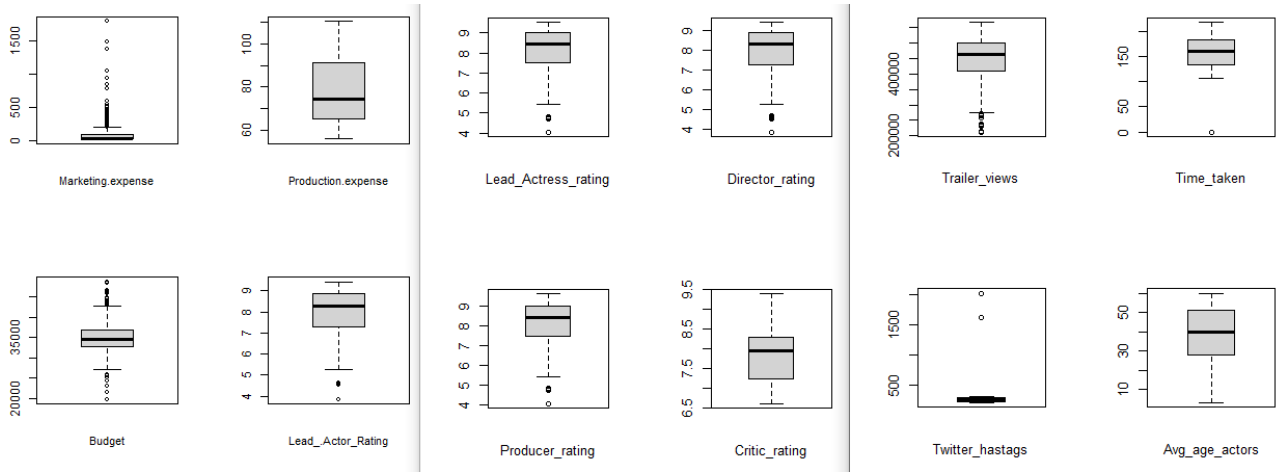


Figura 2.1: Grafico degli outliers

Osserviamo che tutte le variabili indipendenti tranne *Production expense, Critic rating e Avg age actors* contengono outliers.
Per capire se rimuovere o meno gli outliers, sono stati usati, oltre ai boxplot, anche gli istogrammi dei grafici costruiti raggruppando i dati in intervalli di classi e poi plottando il numero di dati che cadono in ogni intervallo, quindi possono essere usati per capire se una variabile è distribuita come una normale.

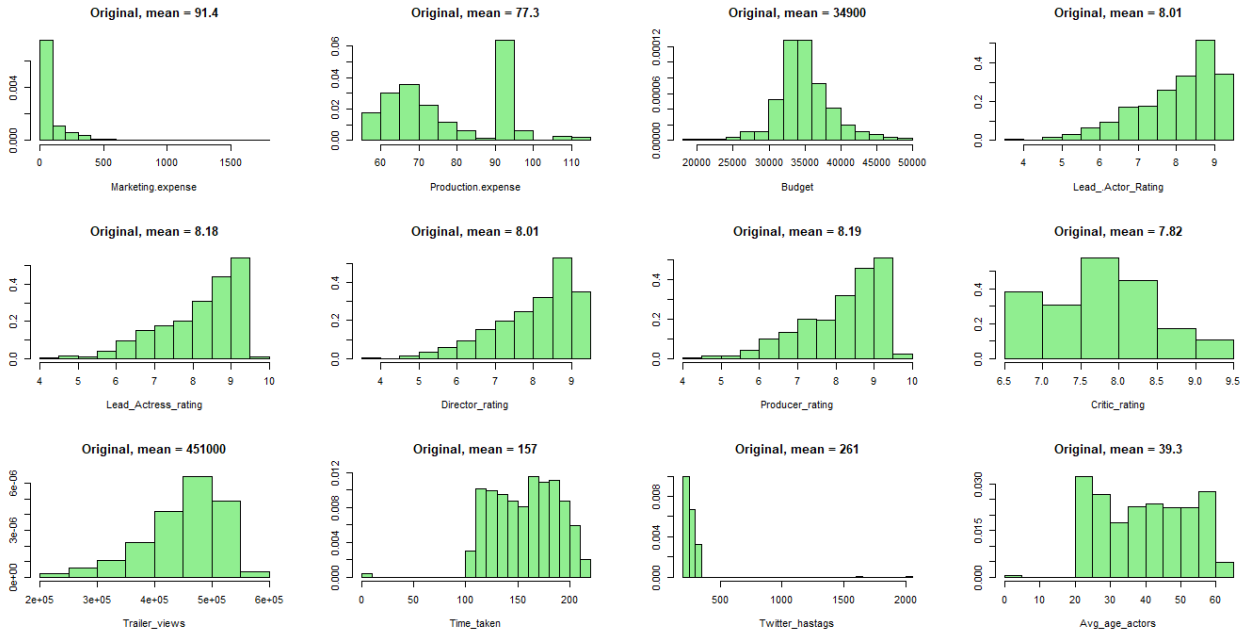


Figura 2.2: Grafico degli istogrammi

Da un'analisi degli istogrammi si può notare che le distribuzioni degli attributi non sono normali ma risultano sbilanciate. Questo è un problema perché può avere un impatto negativo sulla regressione lineare; di conseguenza, si è scelto di apportare delle modifiche alle distribuzioni per renderle più simmetriche possibile, scegliendo di rimuovere gli outliers sopra il 98-esimo percentile da *Marketing expense, Budget, Lead Actor Rating, Lead Actress rating, Director rating, Producer rating, Trailer views, Time taken e Twitter hastags.*

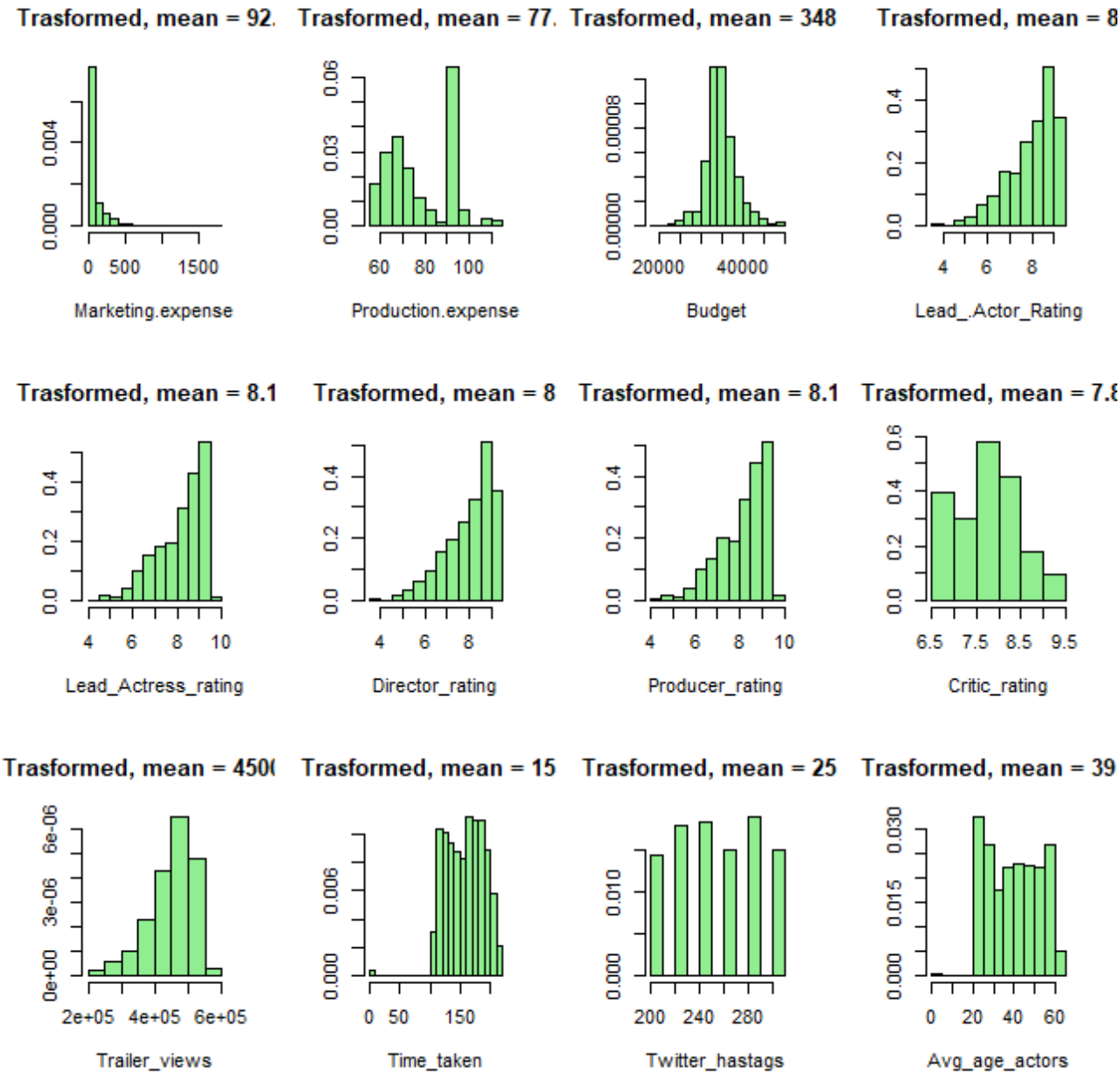


Figura 2.3: Grafico degli istogrammi dopo la rimozione di alcuni outliers

In seguito alla rimozione di alcuni outliers le istanze del nostro dataset si sono ridotte a 484 elementi, quindi sono stati eliminati dal dataset 10 righe.

Nella seguente immagine in dettaglio l'operazione di eliminazione degli outliers per gli attributi di *Marketing expense* e *Twitter hastags*. In particolare a destra vediamo i dati originali e a sinistra i dati dopo la trasformazione.

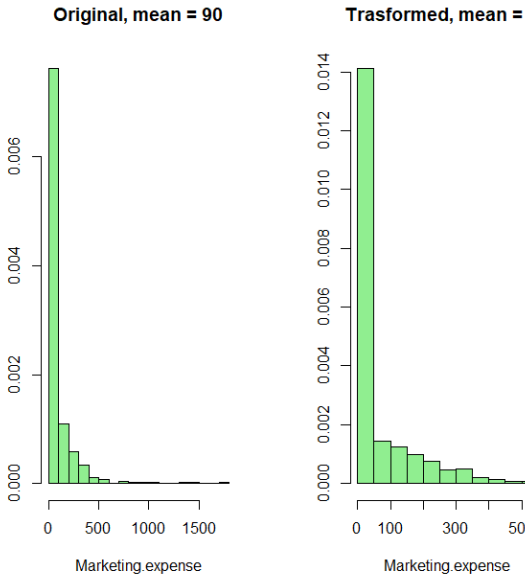


Figura 2.4: Marketing expense

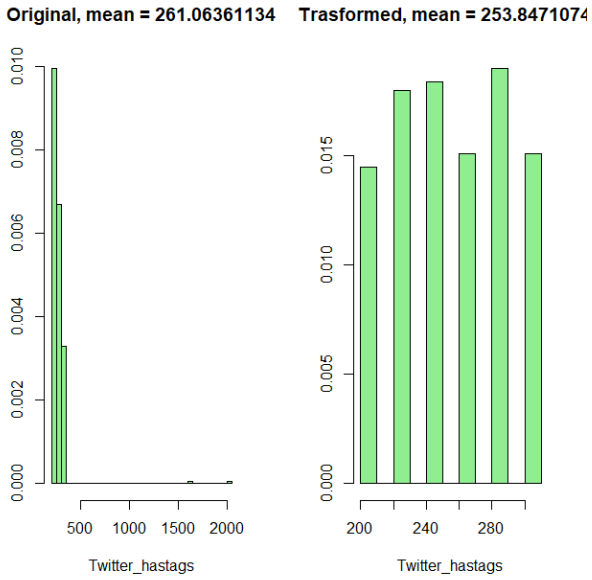


Figura 2.5: Twitter hastags

2.1.2 Matrice di correlazione

Il passo successivo è stato valutare l'indice di correlazione sia tra i predittori presi a coppie sia tra il singolo predittore e la risposta. L'indice di correlazione esprime un'eventuale relazione di linearità tra le variabili, quindi può essere usato per capire quanto sia lineare la relazione tra risposta e predittore (aspetto fondamentale nella regressione lineare) ma anche per valutare se le variabili sono dipendenti tra di loro (quindi possiamo scegliere solo una delle due correlate). È un valore compreso tra -1 e 1:

- se in modulo è uguale a 0, la correlazione è assente.
- se in modulo è compreso tra 0 e 0.3, allora la correlazione è debole.
- se in modulo è compreso tra 0.3 e 0.7, allora la correlazione è moderata.
- se in modulo è compreso tra 0.7 e 1, allora la correlazione è forte.

Di seguito la matrice di correlazione che è stata prodotta col comando **corrplot**:

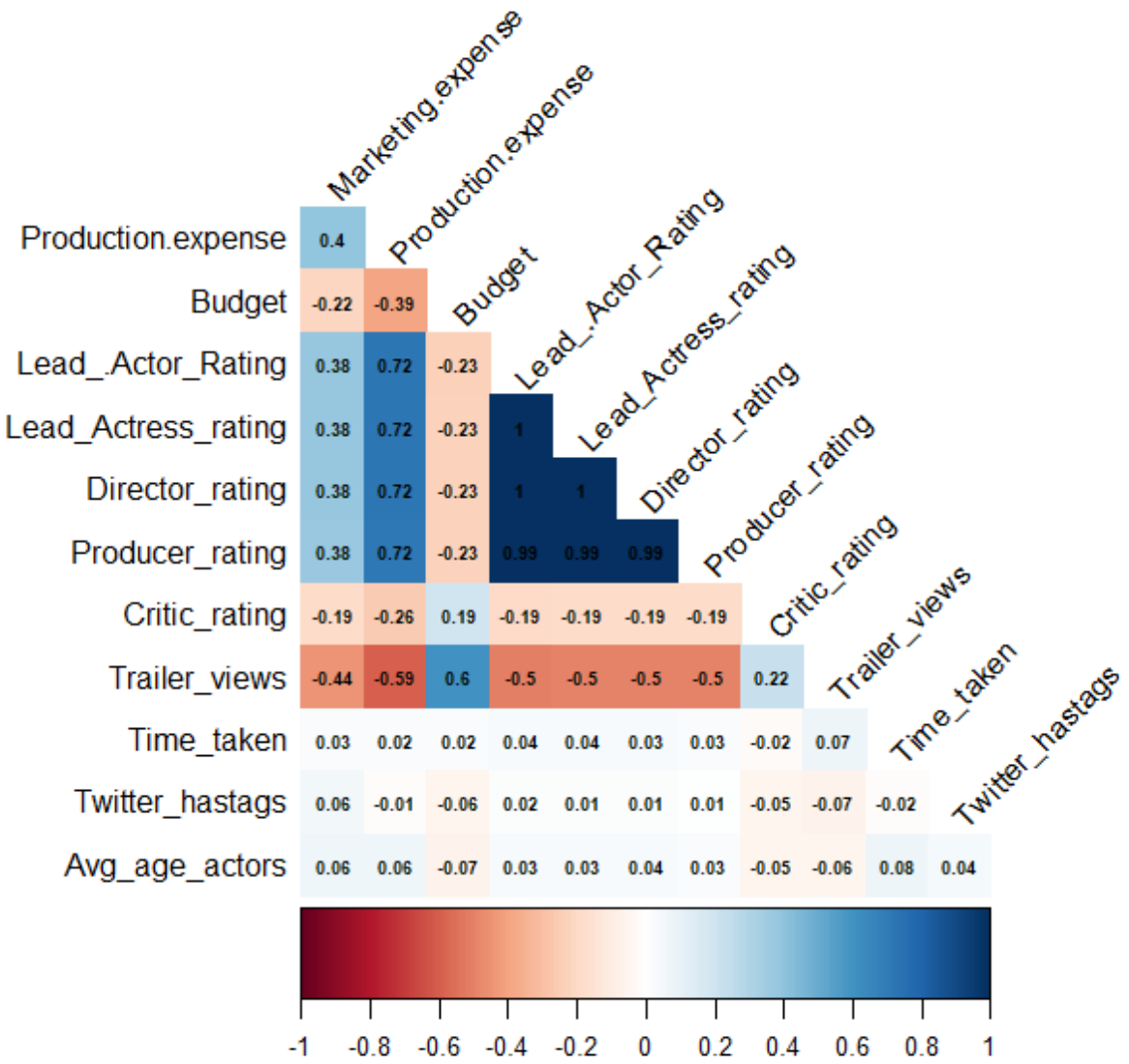
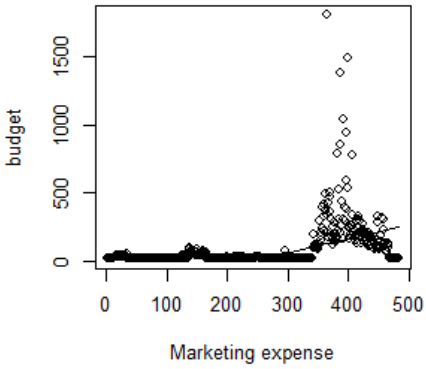
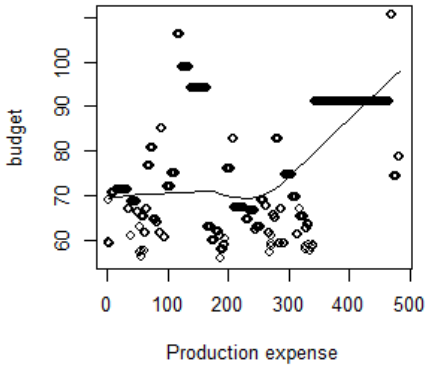


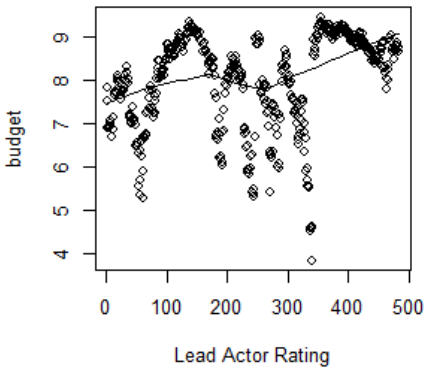
Figura 2.6: Matrice di correlazione

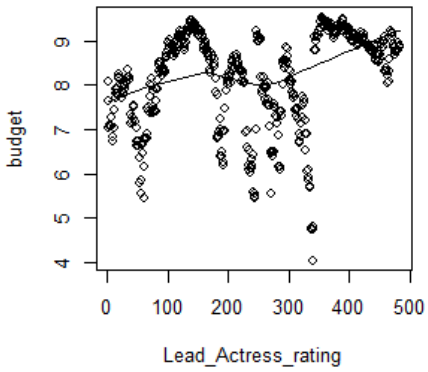
2.1.3 Linearità dei dati

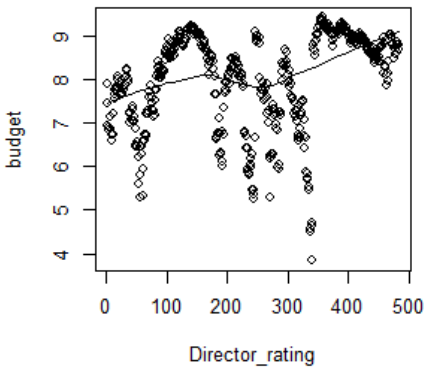
Una delle assunzioni del modello di regressione lineare è che ci sia una relazione lineare tra la risposta e ogni singolo predittore, espressa tramite un coefficiente da stimare opportunamente. Pertanto, si è deciso di controllare se tale assunzione fosse ammissibile anche per il dataset in questione, prima ancora di costruire il modello di regressione, in modo da avere un'idea su quale tipo di modello adottare, prendendo in considerazione eventualmente anche quello polinomiale. A tal proposito sono stati costruiti degli scatterplot tra la risposta e ogni singolo predittore, tramite la funzione **scatter.smooth**, che va a creare uno scatterplot con una smooth line, ovvero una linea smussata per seguire meglio l'andamento di dati.
Di seguito i grafici relativi a tutti i predittori:

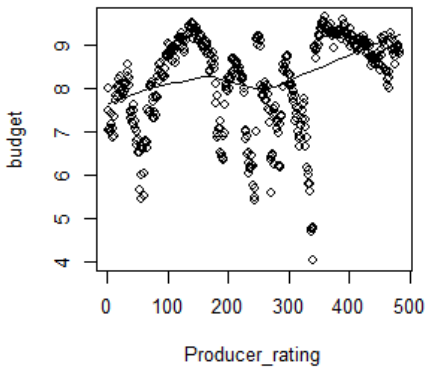

budget
0
500
1000
1500
0
100
200
300
400
500
Marketing expense

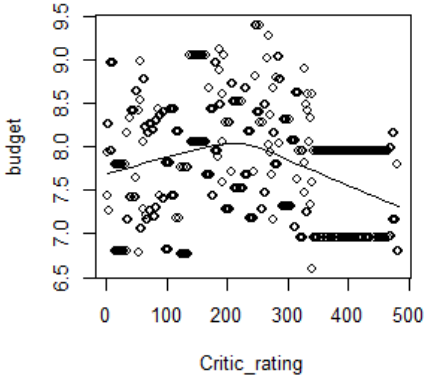

budget
60
70
80
90
100
0
100
200
300
400
500
Production expense

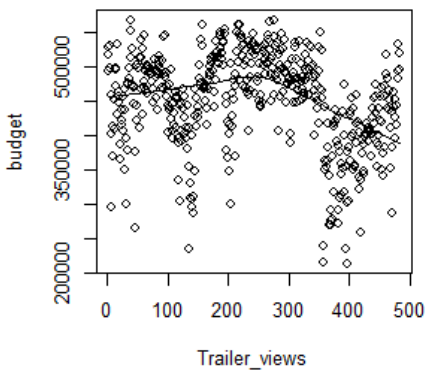

budget
4
5
6
7
8
9
0
100
200
300
400
500
Lead Actor Rating


budget
4
5
6
7
8
9
0
100
200
300
400
500
Lead_Actress_rating


budget
4
5
6
7
8
9
0
100
200
300
400
500
Director_rating


budget
4
5
6
7
8
9
0
100
200
300
400
500
Producer_rating


budget
6.5
7.0
7.5
8.0
8.5
9.0
9.5
0
100
200
300
400
500
Critic_rating


budget
200000
350000
500000
0
100
200
300
400
500
Trailer_views

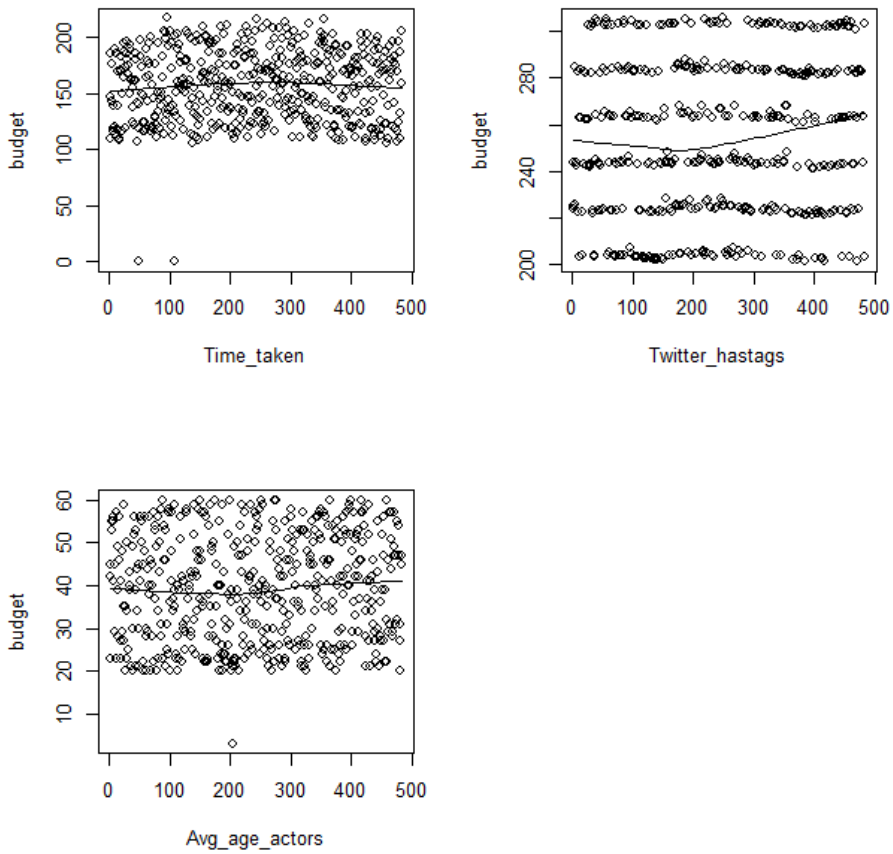


Figura 2.7: Scatterplot tra risposta ed ogni singolo predittore

Si può notare che:

- Time taken e Avg age actors hanno una relazione quasi perfettamente lineare, avvantaggiato anche dalla non presenza di outliers come visto in precedenza.
- Marketing expense ha una relazione lineare per la maggior parte dei valori per poi presentare una piccola curva nella parte finale.
- Lead actor rating, Lead actress rating, Director rating, Producer rating sono tutti caratterizzati dalla stessa medesima curva.
- Tutti gli altri hanno relazioni polinomiali.

Quanto visto ci suggerisce che un modello con una trasformazione polinomiale per quasi tutti i predittori o un modello di regressione completamente polinomiale abbia prestazioni migliori rispetto a uno lineare.

2.2 Regressione multipla

2.2.1 Regressione lineare

La regressione è quella tecnica statistica utilizzata per studiare le relazioni che intercorrono tra due o più caratteri statistici. A differenza della regressione semplice, che analizza la relazione tra una variabile Y e un predittore X, la regressione multipla descrive la connessione tra la variabile Y e più predittori: $X_1, X_2, ..., X_N$.
Strettamente legata alla regressione è il concetto di correlazione, infatti si suppone che più variabili X (nella regressione multipla), assumano valori determinati e si cerca la relazione che lega la variabile Y alle prime. In altre parole, si cerca di stabilire un legame funzionale tra le variabili del tipo

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \tag{2.1}$$

Il tipo di regressione analizzata è la regressione ai minimi quadrati, che ha come obiettivo quello di trovare la miglior retta interpolatrice dei punti del piano (retta di regressione dei minimi quadrati).
Come visto precedentemente, la maggior parte dei predittori non ha una relazione lineare con la risposta; la conseguenza di tale risultato ha spinto alla realizzazione di diverse trasformazioni non lineari dei predittori: trasformazione quadratica, cubica e di ordine 4. Per ognuna, sono stati prodotti i risultati tramite la funzione **summary(model)**, che fornisce per ogni regressore la stima dei coefficienti, lo standard error, il t-value e il p-value.; l'output prodotto per ogni modello riporta anche il "*Residual Standard Error*"(RSE), "*Multiple R-squared*" , "*Adjusted R-squared*" e "*F-statistic*" . Tali risultati sono stati valutati e confrontati tra di loro, per capire quale modello interpretasse nel miglior modo possibile i dati.
Si è partiti dal modello di regressione lineare contenente tutti gli 11 predittori. In questo modello, i predittori più significativi sono Production expense e Trailer views perché, come è possibile vedere dall'immagine seguente, presentano un p-value molto piccolo (valore soglia):

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.713e+04  3.138e+03   5.460  7.7e-08 ***
Marketing.expense   1.371e+00  9.048e-01   1.515 0.130510
Production.expense -5.291e+01  1.584e+01  -3.340 0.000906 ***
Lead_.Actor_Rating -3.268e+03  2.299e+03  -1.421 0.155917
Lead_Actress_rating -3.570e+01  2.441e+03  -0.015 0.988339
Director_rating     3.755e+03  2.387e+03   1.573 0.116398
Producer_rating     2.245e+02  1.274e+03   0.176 0.860238
Critic_rating       3.268e+02  2.197e+02   1.487 0.137664
Trailer_views       3.287e-02  2.613e-03  12.580  < 2e-16 ***
Time_taken         -1.830e+00  4.414e+00  -0.415 0.678563
Twitter_hastags    -1.889e+00  4.162e+00  -0.454 0.650206
Avg_age_actors     -9.310e+00  1.104e+01  -0.843 0.399580
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3003 on 472 degrees of freedom
Multiple R-squared:  0.3941,    Adjusted R-squared:   0.38
F-statistic: 27.91 on 11 and 472 DF,  p-value: < 2.2e-16
```

Figura 2.8: Summary del fit lineare

Successivamente, è stata valutata la multicollinearità, ovvero quando 2 o più predittori sono strettamente collegati tra di loro; per verificare tale proprietà, è stato calcolato il VIF (Variance Inflaction Factor) ed è risultato che le variabili Lead Actor Rating, Lead Actress rating, Director rating e Producer rating eccedono di gran lunga il valore limite 10, come si evince dalla figura 2.9.
Dalla teoria sappiamo che valori elevati del VIF indicano presenza di colinearità.

```
vif(fit)
Marketing.expense  Production.expense  Lead_.Actor_Rating Lead_Actress_rating     Director_rating     Producer_rating
         1.319535            2.545263           317.920642          358.244718          346.691809           97.074204
    Critic_rating       Trailer_views          Time_taken     Twitter_hastags      Avg_age_actors
         1.094682            1.738411            1.031922            1.020655            1.025137
```

Figura 2.9: VIF lineare

Di conseguenza, è stato creato un nuovo modello eliminando 3 delle 4 variabili sopra citate (Lead Actor Rating, Lead Actress rating e Producer rating); tuttavia, confrontandolo con il modello lineare originale, non si ottengono prestazioni migliori.
Questo confronto, così come tutti gli altri effettuati successivamente, è stato realizzato con la funzione **anova(Model 1, Model 2)**.

```
Model 1: Budget ~ Marketing.expense + Production.expense + Lead_.Actor_Rating +
    Lead_Actress_rating + Director_rating + Producer_rating +
    Critic_rating + Trailer_views + Time_taken + Twitter_hastags +
    Avg_age_actors
Model 2: Budget ~ (Marketing.expense + Production.expense + Lead_.Actor_Rating +
    Lead_Actress_rating + Director_rating + Producer_rating +
    Critic_rating + Trailer_views + Time_taken + Twitter_hastags +
    Avg_age_actors) - Lead_.Actor_Rating - Lead_Actress_rating -
    Producer_rating
  Res.Df        RSS Df Sum of Sq     F Pr(>F)
1    472 4256908181
2    475 4280826950 -3 -23918769 0.884 0.4492
```

Figura 2.10: Confronto tra due modelli lineari

Dal valore dell' RSS si deduce che è migliore il modello con tutti i predittori.

Per valutare la bontà del modello creato, in particolare in relazione alle assunzioni della regressione lineare, analizziamo i seguenti 4 grafici diagnostici:

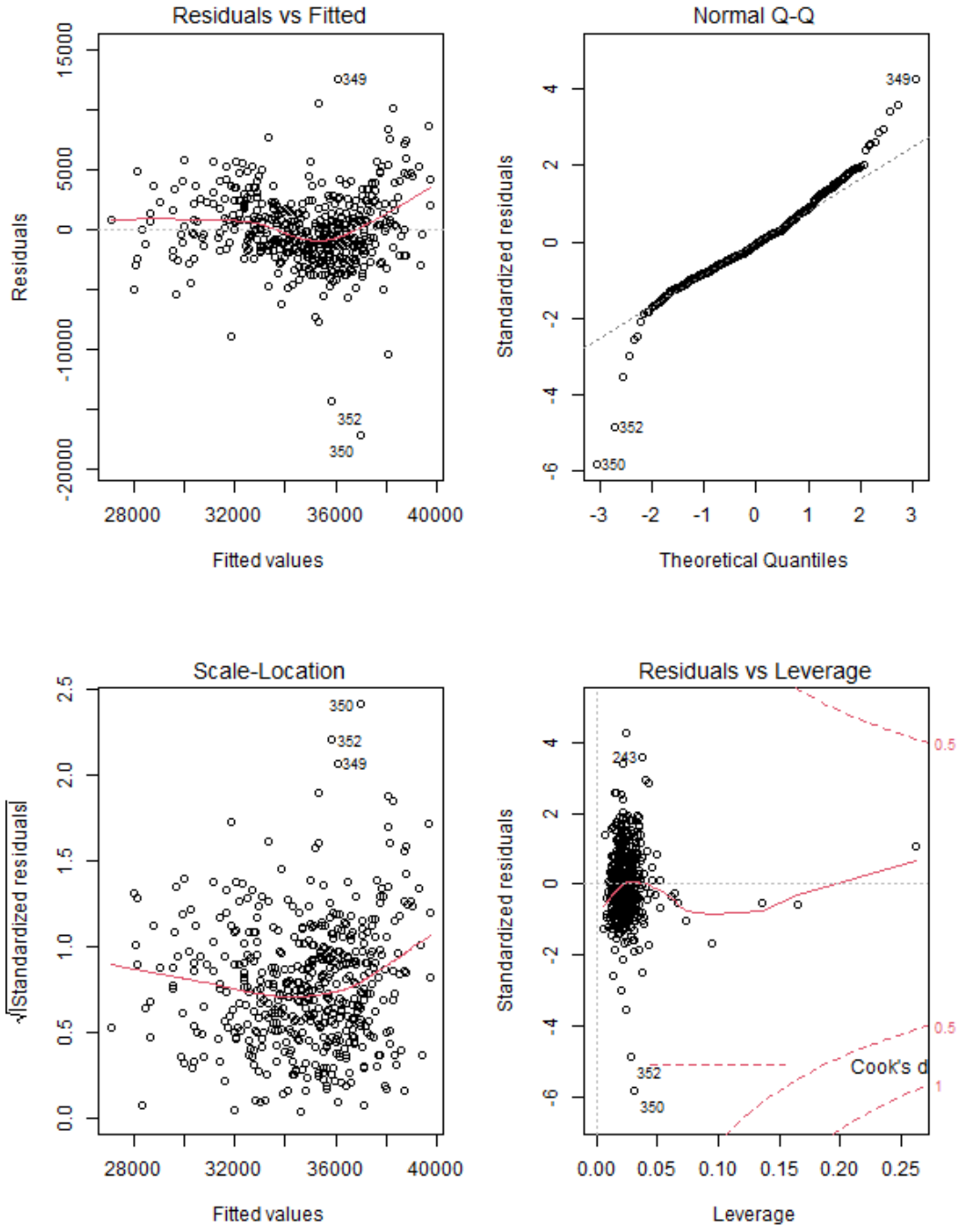


Figura 2.11: Intervallo di confidenza delle variabili

- Il plot in alto a sinistra mostra gli errori residui contro i loro valori stimati. I residui devono essere distribuiti in modo casuale attorno alla linea rossa orizzontale che rappresenta un errore residuo pari a zero, cioè non dovrebbe esserci una netta tendenza nella distribuzione dei punti. Non si evidenzia una tendenza particolare ma le osservazioni 349, 352 e 350 sono abbastanza lontane dai valori della regressione, notiamo che i valori più lontani sono collocati rispettivamente uno sopra i valori teorici della regressione, e due sotto.

- Il plot in alto a destra è un plot Q-Q standard, ovvero confronta i valori dei residui standardizzati (quantile reale) verso la linea che individua la loro distribuzione normale (quantile teorico), ovvero il grafico rappresenta una figura per cui se i punti si distribuiscono sulla linea, la distribuzione dei residui risulta normale e quindi la regressione rappresenta un modello adeguato. Se ciò non accade, vuol dire che la varianza non è costante come assume il modello lineare e quindi che gli intervalli di confidenza ed i p-value calcolati non sono esatti.

- Il plot in basso a sinistra mostra la radice quadrata dei residui standardizzati in funzione dei valori stimati. E' un grafico molto utile per individuare gli outliers che sono rappresentati da residui non distribuiti equamente sopra e sotto la linea orizzontale. Anche in questo grafico notiamo che le osservazioni 349, 352 e 350 sono abbastanza lontane dalla linea orizzontale.

- Infine il plot in basso a destra serve per identificare outliers che hanno grande influenza sul fit del modello, in particolare sono quei punti che hanno un residuo alto ed una leverage alta. I punti spostati a destra ed in alto sono quelli che hanno peso maggiore sulla regressione. Sovrapposte al plot ci sono linee di contorno per la distanza di Cook, che è un'altra misura dell'importanza di ciascuna osservazione sulla regressione. Valori di distanze di Cook bassi per un punto indicano che la rimozione della rispettiva osservazione ha poco effetto sui risultati della regressione, ovvero l'osservazione in particolare non ha valori devianti dalla tendenza. Invece valori di distanze di Cook superiori a 1 sono sospetti ed indicano la presenza di un possibile outlier o di un modello povero.

Pertanto, come si evince dalla figura sopra riportata, non si ha una relazione lineare, in quanto la linea presenta delle curvature e inoltre i dati non hanno la stessa varianza perché i residui sono troppo sparsi nella zona centrale. Inoltre, i residui non sono distribuiti come una normale perché sono presenti delle code che hanno valori più grandi rispetto a quelli desiderati. Infine, notiamo la presenza di alcuni outliers ad alta leverage.

2.2.2 Regressione polinomiale

Per ovviare ai problemi descritti nel paragrafo precedente, dal momento che sembra non esserci una relazione lineare tra la risposta ed i predittori, è stato scelto di costruire un modello di regressione polinomiale di vario ordine per ottenere un miglior fit dei dati. Dalle analisi fatte, il modello di regressione polinomiale migliore è risultato quello con trasformazione di ordine 4 dei regressori.
Nella seguente immagine sono riportati, a titolo di esempio, i risultati di quest'ultimo:

```
Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                     34825.68     126.14 276.097  < 2e-16 ***
poly(Marketing.expense, 4)1     11102.21    4431.54   2.505 0.012597 *
poly(Marketing.expense, 4)2     -5035.07    4014.69  -1.254 0.210451
poly(Marketing.expense, 4)3     10031.47    3362.99   2.983 0.003014 **
poly(Marketing.expense, 4)4     -1741.72    3258.60  -0.535 0.593266
poly(Production.expense, 4)1   -15636.78    5413.67  -2.888 0.004064 **
poly(Production.expense, 4)2    11683.98    3339.05   3.499 0.000514 ***
poly(Production.expense, 4)3    -6277.02    3856.99  -1.627 0.104361
poly(Production.expense, 4)4    -3727.55    3020.30  -1.234 0.217802
poly(Lead_.Actor_Rating, 4)1   -81001.90   58653.73  -1.381 0.167976
poly(Lead_.Actor_Rating, 4)2   -11031.05   44353.31  -0.249 0.803702
poly(Lead_.Actor_Rating, 4)3    -7681.08   42923.20  -0.179 0.858060
poly(Lead_.Actor_Rating, 4)4   -10513.38   24685.13  -0.426 0.670390
poly(Lead_Actress_rating, 4)1 -13641.40   61596.59  -0.221 0.824834
poly(Lead_Actress_rating, 4)2  35379.03   45119.25   0.784 0.433391
poly(Lead_Actress_rating, 4)3   2379.91   39070.98   0.061 0.951457
poly(Lead_Actress_rating, 4)4  10229.79   25566.53   0.400 0.689260
poly(Director_rating, 4)1      60235.75   61492.26   0.980 0.327840
poly(Director_rating, 4)2      -5956.48   41555.80  -0.143 0.886090
poly(Director_rating, 4)3         23.19   35728.22   0.001 0.999482
poly(Director_rating, 4)4      -9141.47   23952.11  -0.382 0.702901
poly(Producer_rating, 4)1      41734.70   35746.28   1.168 0.243632
poly(Producer_rating, 4)2     -28439.16   35484.15  -0.801 0.423298
poly(Producer_rating, 4)3      -5018.72   36664.43  -0.137 0.891186
poly(Producer_rating, 4)4       -562.18   18921.96  -0.030 0.976311
poly(Critic_rating, 4)1         3737.77    3141.72   1.190 0.234799
poly(Critic_rating, 4)2         4233.07    2943.54   1.438 0.151122
poly(Critic_rating, 4)3         5530.02    2936.00   1.884 0.060291 .
poly(Critic_rating, 4)4         3543.36    3087.61   1.148 0.251756
poly(Trailer_views, 4)1        44912.83    4116.86  10.909  < 2e-16 ***
poly(Trailer_views, 4)2         8857.49    3223.41   2.748 0.006246 **
poly(Trailer_views, 4)3        15555.69    2968.23   5.241 2.49e-07 ***
poly(Trailer_views, 4)4        -1712.53    2903.08  -0.590 0.555560
poly(Time_taken, 4)1           -2386.90    2920.33  -0.817 0.414178
poly(Time_taken, 4)2            1064.28    2843.72   0.374 0.708394
poly(Time_taken, 4)3            3393.26    2863.05   1.185 0.236583
poly(Time_taken, 4)4            4125.18    2953.90   1.397 0.163263
poly(Twitter_hastags, 4)1       -590.98    2896.59  -0.204 0.838429
poly(Twitter_hastags, 4)2      -5967.50    2915.72  -2.047 0.041286 *
poly(Twitter_hastags, 4)3       -384.44    2963.71  -0.130 0.896852
poly(Twitter_hastags, 4)4       -963.19    2892.76  -0.333 0.739318
poly(Avg_age_actors, 4)1        -941.21    2900.81  -0.324 0.745741
poly(Avg_age_actors, 4)2         162.91    2862.38   0.057 0.954638
poly(Avg_age_actors, 4)3        3235.85    2857.95   1.132 0.258157
poly(Avg_age_actors, 4)4        4315.83    2893.09   1.492 0.136478
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2775 on 439 degrees of freedom
Multiple R-squared:  0.5188,    Adjusted R-squared:  0.4706
F-statistic: 10.76 on 44 and 439 DF,  p-value: < 2.2e-16
```

Figura 2.12: Summary del fit polinomiale di 4°ordine.

Rispetto al modello lineare, si nota un incremento nell'R2 (da 0.3941 a 0.5188) e una riduzione dell'RSE (da 3003 a 2775), ciò è indice di prestazioni leggermente migliori.

2.3 Ricampionamento statistico

I metodi di ricampionamento statistici utilizzati per testare i modelli realizzati sono: Validation Set, K-fold Cross Validation e Bootstrap.
Queste tecniche vengono utilizzate per effettuare il fit di un modello su una parte di dati riservati per il training e un'altra parte di dati per il test del modello ottenuto. L'obiettivo è sempre quello di trovare il modello ottimale, che si comporti in maniera soddisfacente sia sul set di allenamento sia sul set di test. Un'indicazione indispensabile sulla risposta del modello viene fornita dal MSE (Mean Square Error), che rappresenta la discrepanza tra i valori dei dati osservati ed i valori dei dati stimati. Per tutti i metodi citati, andremo a esaminare l'MSE stimato sul test set da tutti modelli, confrontando i risultati ottenuti. Da notare che, man mano che la complessità di un modello aumenta, l'MSE sul training set diminuisce; questo avviene perché la complessità del modello aiuta ad apprendere, in maggior misura, tutte le variazioni che contraddistinguono i dati di addestramento. Tuttavia, questo porta ad un sovradattamento del modello sui dati di prova e ad un malfunzionamento sui campioni di test. Questo fenomeno è noto come overfitting.

2.3.1 Validation Set

L'approccio del set di validazione è un semplice metodo di divisione dei dati per l'addestramento ed il test. I dati sono divisi in due parti: la prima parte viene utilizzata per addestrare il modello, la seconda viene utilizzata per testarlo. Nel caso in esame, il criterio usato è quello di assegnare un numero di campioni pari a 242 per entrambe la parti.
I valori ottenuti sono riprodotti nella tabella seguente:

Validation Test	
Trasformazione	MSE
Lineare	9986868
Polinomiale di 2° ordine	10118900
Polinomiale di 3° ordine	11052963
Polinomiale di 4° ordine	83488661

Tabella 2.1: Tabella valori MSE per Validation Set.

2.3.2 K-Fold Cross Validation

Il metodo K-Fold Cross Validation elimina molti svantaggi del metodo del set di validazione. Principalmente assicura che il bias non penetri nelle prestazioni del modello. Lo fa allenandosi e testando su ciascuno dei sottogruppi in cui vengono divisi i dati. Il numero di sottogruppi (folds) scelto per l'analisi è 10, ognuno caratterizzato da una serie casuale di punti dati. Questo numero indica anche la quantità di iterazioni che vengono eseguite per l'addestramento ed il test dei sottogruppi. Le prestazioni complessive del modello vengono calcolate in base all'errore medio in tutte le iterazioni.
I valori ottenuti sono riprodotti nella tabella seguente:

K-Fold Cross Validation	
Trasformazione	MSE
Lineare	9187535
Polinomiale di 2° ordine	9427550
Polinomiale di 3° ordine	8709870
Polinomiale di 4° ordine	10049127

Tabella 2.2: Tabella valori MSE per K-Fold Cross Validation.

I risultati sull'MSE ottenuti sono simili ma leggermente superiori a quelli ottenuti tramite l'approccio Validation Set. Ciò è dovuto al fatto che in questo caso i modelli sono stati testati su più set di test differenti, questi ultimi risultati sono da considerarsi più attendibili.
Si può notare come il modello polinomiale di 4° ordine, che aveva prestazioni migliori nel fit dei dati, in questo caso risulta essere il peggiore dato il maggiore valore di MSE rispetto agli altri modelli.

2.3.3 Bootstrap

Un terzo metodo di ricampionamento dei dati è quello Bootstrap, ovvero un metodo statistico che può essere utilizzato per stimare l'incertezza associata ad uno stimatore. Si stima l'errore standard degli stimatori e si usa un dataset limitato; è utile per costruire il modello e stimare i coefficienti. Questa tecnica viene usata anche per ottenere nuovi dataset a partire da quello che abbiamo, ripetendo il campionamento a partire dalle osservazioni.
Viene usata la funzione **boot()** per calcolare la stima dello standard error dei coefficienti e valutarne le differenze. In base al grado del polinomio si stimano gli standard error e si confrontano; essi devono avere piccole differenze (c - se) tra di loro per mostrare che il metodo è affidabile. Con la funzione **boot.fn()** si ottengono stime dei parametri intercetta e coefficiente angolare.
A titolo di esempio, vengono mostrate le differenze tra gli standard error predetti con bootstrap e quelli calcolati con i minimi quadrati, relativi ai coefficienti della trasformazione lineare:

Bootstrap	
Coefficienti	Standard Error difference
Marketing expense	3137.744
Production expense	0.9048451
Budget	15.84347
Lead Actor Rating	2299.317
Lead Actress rating	2441.141
Director rating	2387.089
Producer rating	1274.279
Critic rating	219.7297
Trailer views	0.0026128
Time taken	4.413881
Twitter hastags	4.162027
Avg age actors	11.0422

Tabella 2.3: Tabella delle differenze tra gli standard error predetti con bootstrap e quelli calcolati con i minimi quadrati.

Una piccola differenza nella stima dello Standard Error è sinonimo di una buona interpretabilità dei dati da parte del modello scelto.

2.4 Subset Selection

La Subset Selection è il processo di selezione di un sottoinsieme di predittori da utilizzare nella costruzione del modello. In questo paragrafo vengono considerati alcuni dei metodi per la selezione del modello migliore: Best Subset Selection, Forward Stepwise Selection e Backward Stepwise Selection.
Tutti questi metodi semplificano il modello al fine di migliorare l'accuratezza dei dati. Questi sono stati applicati per tutte le diverse trasformazioni, già precedentemente descritte; in seguito sono stati commentati i risultati ottenuti con grafici di supporto.

2.4.1 Best Subset Selection

La funzione **regsubset()** in R, presente nel pacchetto leaps, può essere utilizzata per identificare diversi modelli migliori di dimensioni diverse. È necessario specificare il parametro opzionale *nvmax*, che rappresenta il numero di predittori da incorporare nel modello. La funzione **summary()** riporta il miglior set di variabili per ogni dimensione del modello.
Di seguito, viene mostrato il Summary relativo ai coefficienti della trasformazione lineare:

```
         Marketing.expense Production.expense Lead_.Actor_Rating Lead_Actress_rating Director_rating Producer_rating
1  ( 1 ) " "               " "                " "               " "                " "             " "
2  ( 1 ) " "               " "                " "               " "                "*"             " "
3  ( 1 ) " "               "*"                " "               " "                "*"             " "
4  ( 1 ) " "               "*"                "*"               " "                "*"             " "
5  ( 1 ) " "               "*"                "*"               " "                "*"             " "
6  ( 1 ) "*"               "*"                "*"               " "                "*"             " "
7  ( 1 ) "*"               "*"                "*"               " "                "*"             " "
8  ( 1 ) "*"               "*"                "*"               " "                "*"             " "
         Critic_rating Trailer_views Time_taken Twitter_hastags Avg_age_actors
1  ( 1 ) " "           "*"           " "        " "             " "
2  ( 1 ) " "           "*"           " "        " "             " "
3  ( 1 ) " "           "*"           " "        " "             " "
4  ( 1 ) " "           "*"           " "        " "             " "
5  ( 1 ) "*"           "*"           " "        " "             " "
6  ( 1 ) "*"           "*"           " "        " "             " "
7  ( 1 ) "*"           "*"           " "        " "             "*"
8  ( 1 ) "*"           "*"           " "        "*"             "*"
```

Figura 2.13: Summary trasformazione lineare.

Dalla figura appena mostrata ne deriva che un asterisco specifica che una data variabile è inclusa nel modello corrispondente. Per confrontare le prestazioni complessive dei modelli e scegliere quello migliore sono necessarie alcune metriche e strategie statistiche. Nello specifico, è necessario stimare l'errore di previsione di ciascun modello e selezionare quello con l'errore minimo. I criteri di selezione del modello sono: R2 , Cp e BIC. La funzione **summary()** restituisce alcune metriche (Adjusted R2, Cp, BIC), permettendoci di identificare il modello migliore complessivo; per migliore si intende un modello che massimizza R2 e minimizza l'errore di previsione (RSS, Cp, BIC).
Non esiste un'unica soluzione corretta per la selezione dei modelli, ciascuno di questi criteri porterà a modelli leggermente diversi. Infatti, nel caso in esame, per ogni metrica presa in considerazione, abbiamo ottenuto modelli diversi per le trasformazioni non lineari.

Di seguito verranno mostrati graficamente i modelli migliori rispetto alle metriche sopra citate:

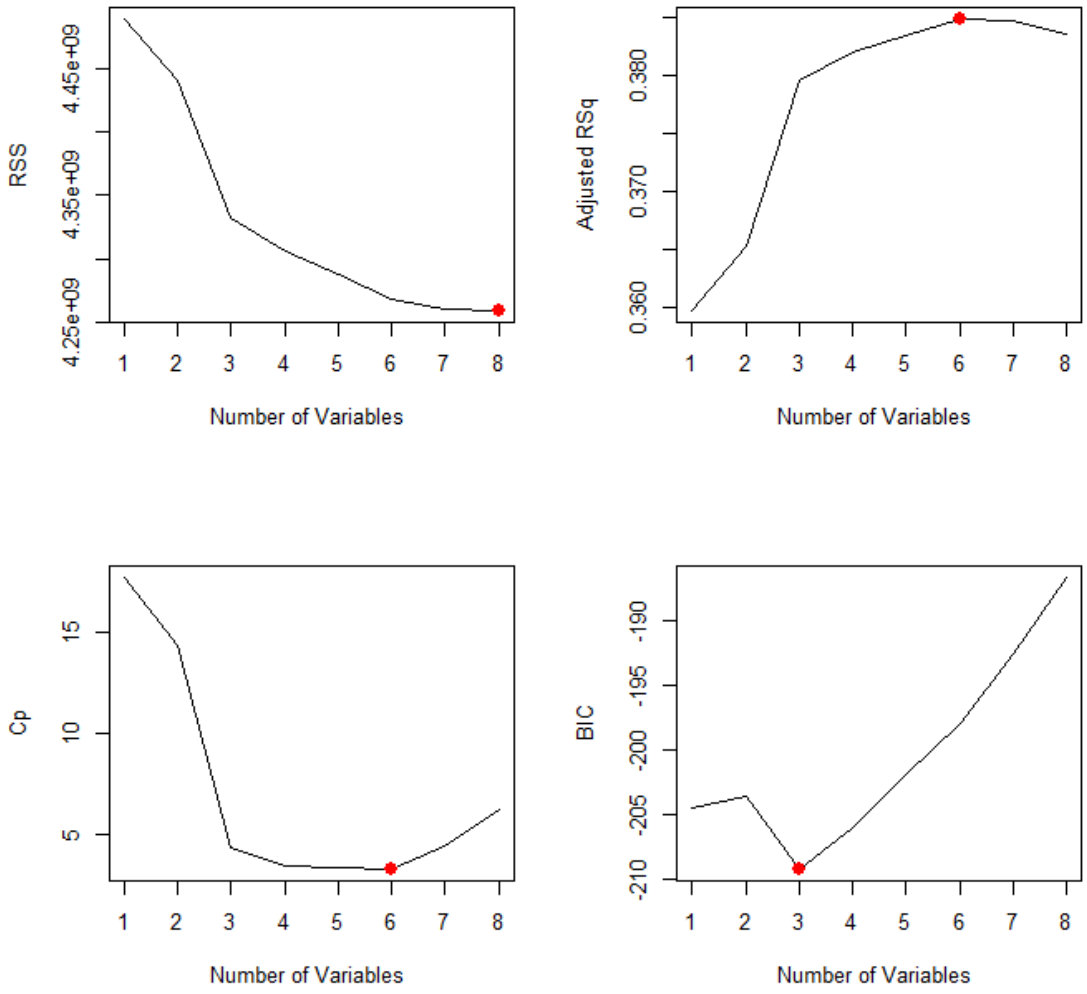


Figura 2.14: Plot trasformazione lineare.

- Grafico in alto a sinistra: mostra l'RSS all'aumentare dei predittori all'interno del modello (minimo a 8 predittori).
- Grafico in alto a destra: mostra l'Adjusted R2 all'aumentare dei predittori all'interno del modello (massimo a 6 predittori).
- Grafico in basso a sinistra: mostra il Cp all'aumentare dei predittori all'interno del modello (minimo a 6 predittori).
- Grafico in basso a destra: mostra il BIC all'aumentare dei predittori all'interno del modello (minimo a 3 predittori).

Come è possibile vedere dai grafici abbiamo modelli diversi a seconda della metrica utilizzata.

Oltre all'output del summary mostrato in precedenza, per capire quali predittori sono stati eliminati da un determinato modello, si può utilizzare anche la seguente rappresentazione:

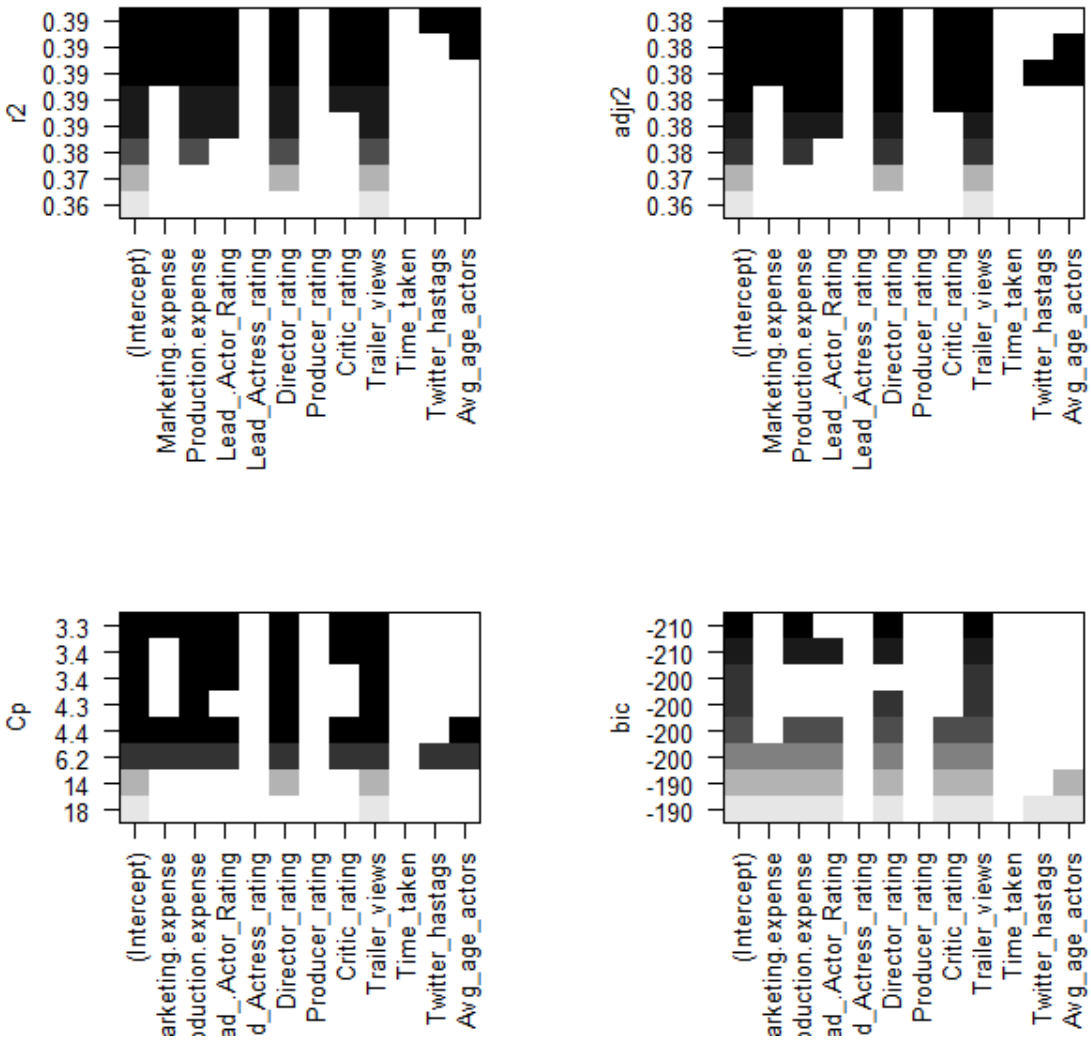


Figura 2.15: Plot alternativo per trasformazione lineare.

Il miglior modello relativo ad una data metrica si ottiene considerando la prima riga del grafico partendo da sopra, in cui ogni rettangolo nero indica che il predittore corrispondente va inserito nel modello, mentre ogni spazio bianco indica l'esclusione di quel determinato predittore dal modello.
Di seguito viene riportata una tabella contenente i risultati ottenuti per tutti gli altri modelli presi in considerazione

Modello	RSS	Adjusted R2	Cp	BIC
Polinomiale 2°	22	15	4	7
Polinomiale 3°	33	17	6	13
Polinomiale 4°	44	26	6	16

Tabella 2.4: Tabella dei risultati ottenuti per le trasformazioni polinomiali.

2.4.2 Stepwise Selection

La Stepwise Selection (regressione graduale) 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 sottrazione (backward), dall'insieme di tutte le variabili.

2.4.3 Forward Selection

Tale approccio prevede di iniziare con una sola variabile nel modello, di testare l'aggiunta di ogni variabile utilizzando un criterio di adattamento del modello scelto, aggiungendo la variabile. Se l'inclusione di quest'ultima fornisce un miglioramento significativo dell'adattamento, la variabile viene aggiunta e il processo viene iterato fin quando nessuna variabile aggiunta ottimizzerà il modello.

2.4.4 Backward Selection

Questa tecnica impone di iniziare con tutte le variabili del modello, di testare la cancellazione di ogni variabile utilizzando un criterio, ed eventualmente eliminarla. Se la perdita di quest'ultima determina un deterioramento insignificante del modello, la variabile viene eliminata e il processo viene iterato fin quando ulteriori variabili rimosse non produrranno perdite significative.

2.4.5 Considerazioni finali

Tutta questa premessa sul meccanismo che sta dietro la generazione di modelli mediante questi due approcci ci serve per mettere in evidenza un risultato fondamentale. Infatti, la risposta prodotta da queste due tecniche, in termini di valori parametrici, è la stessa di quella ottenuta attraverso la Best Subset Selection. Questo fa capire come non ci sia una discordanza, nonostante l'algoritmo utilizzato per la ricerca del modello migliore sia diverso.
Per valutare la prestazione del modello migliore, trovato da queste tre tecniche, è stato utilizzato il Validation Set Approach. E' stato realizzato il fit sul training set e poi stimato l'MSE sul test set. Il modello migliore che fornisce il minimo MSE è quello con trasformazione di ordine 3 dei predittori.

Modello	Predittori considerati	Min MSE
Lineare	11	9076750
Polinomiale di 2° ordine	22	9156773
Polinomiale di 3° ordine	33	8717899
Polinomiale di 4° ordine	44	8812859

Tabella 2.5: Tabella di confronto tra modelli, predittori e valore MSE.

Come si evince dalla tabella, l'MSE minimo si ha con un modello di 33 predittori.

2.5 Regressione con Regolarizzazione

In un modello di Statistical Learning, la regolarizzazione riduce significativamente la varianza del modello, senza un sostanziale aumento del suo bias. Attraverso l'introduzione di un parametro λ, si punta a diminuire i coefficienti del modello in modo da ridurre la varianza e quindi l'accuratezza stessa.
Le due tecniche di regolarizzazione affrontate in questo progetto sono Ridge e LASSO. Anche in questo caso il modo di lavorare è stato lo stesso adoperato per la Subset Selection: analizzare i risultati prodotti dai modelli di diverse trasformazioni e confrontarli per valutare quale abbia un peso migliore. Inoltre, data l'influenza di λ, si è preferito usare la regola del cross-validation per stimarla in maniera più affidabile, invece di usare ben precisi valori arbitrari. In questo modo, è stato possibile calcolare il miglior valore di λ, in modo da ottenere risultati più vantaggiosi.

2.5.1 Ridge

Con regressione Ridge ci si riferisce ad un modello di regressione lineare i cui coefficienti non sono stimati dal metodo dei minimi quadrati (OLS), ma da un altro stimatore, chiamato stimatore Ridge, che possiede bias, ma ha una varianza inferiore rispetto allo stimatore OLS.
Lo stimatore Ridge riduce i coefficienti di regressione, in modo che le variabili, con un contributo minore al risultato, abbiano i loro coefficienti vicini allo zero. Invece di forzarli a essere esattamente zero, vengono penalizzati con un termine chiamato norma L2, costringendoli così a essere piccoli in modo continuo. In questo modo, diminuiamo la complessità del modello senza eliminare nessuna variabile.
L'ammontare della penalità può essere messo a punto usando una costante chiamata λ:

$$\widehat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{b}} \sum_{i=1}^{n} (y_i - x_i b)^2 + \lambda \sum_{k=1}^{K} b_k^2, \tag{2.2}$$

Quando $\lambda = 0$, il termine di penalità non ha alcun effetto e la regressione Ridge produrrà i coefficienti minimi quadrati classici. Quando λ aumenta all'infinito, l'impatto della penalità aumenta e i coefficienti di regressione si avvicinano allo zero. La regressione Ridge è particolarmente vantaggiosa rispetto al metodo dei minimi quadrati in una situazione in cui si hanno molti dati multivariati con il numero di predittori maggiore del numero di osservazioni.

Analisi e osservazioni

Per mostrare a titolo qualitativo i risultati ottenuti, l'analisi seguente verrà commentata solo sul caso della trasformazione non lineare di quarto grado. Per poter utilizzare una regressione con regolarizzazione in RStudio, è necessario utilizzare la funzione **glmnet()** presente nell'omonimo package. Dapprima, è stata creata una griglia di 100 valori da 10^-2 a 10^2 disponibili per λ. In seguito, sono stati testati i modelli con un valore di λ arbitrario all'interno di tale griglia e, successivamente, con il miglior λ restituito dalla cross-validation.
Si nota che, come atteso, all'aumentare di λ i coefficienti sono diminuiti.
Il valore dell'MSE al crescere del fattore λ viene rappresentato nel grafico seguente:

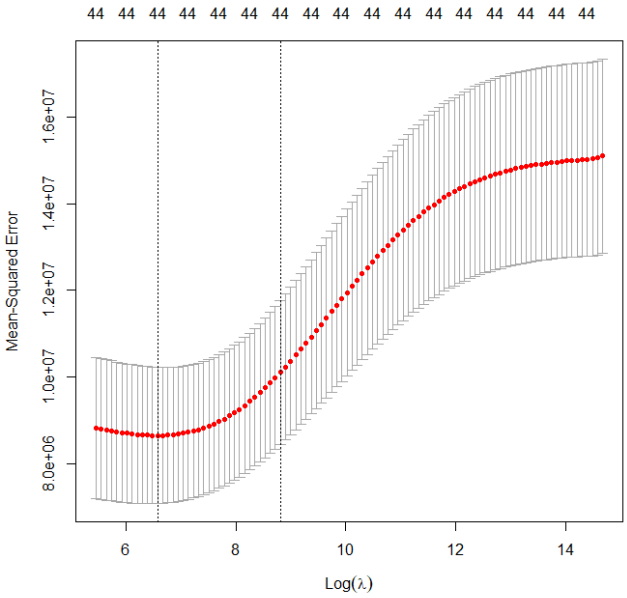


Figura 2.16: Variazione dell'MSE sul test set in relazione a λ.

Questo grafico mostra, per ogni valore dell'MSE, l'intervallo di confidenza calcolato usando la cross-validation. Le linee verticali rappresentano λ_{min} e λ_{1SE}; la prima rappresenta il valore che minimizza il cross-validation error, la seconda rappresenta il valore corrispondente all'aggiunta di uno Standard Error al precedente valore minimo dell'errore.
Un'altra osservazione da fare riguarda la variazione dei coefficienti del modello in analisi, in base al crescere di λ all'interno della griglia citata prima:

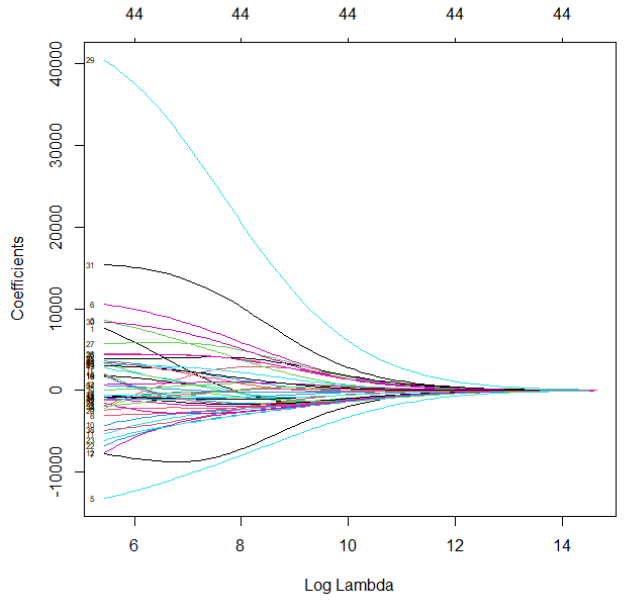


Figura 2.17: Variazione dei coefficienti con differenti valori di λ.

Dal grafico si può notare come, man mano che λ cresce, il termine di regolarizzazione ha un effetto maggiore perché sempre più coefficienti assumono valori prossimi allo 0.

Di seguito viene riportata una tabella contenente i migliori valori di λ per ogni modello:

Trasformazione	Miglior lambda
Lineare	5.461045
Polinomiale di 2° ordine	5.647112
Polinomiale di 3° ordine	7.042618
Polinomiale di 4° ordine	6.57745

Tabella 2.6: Tabella valori λ migliori per ciascun modello.

2.5.2 LASSO

LASSO (Least Absolute Shrinkage and Selection Operator), riduce i coefficienti di regressione verso lo zero, penalizzando il modello di regressione con un termine di penalità chiamato norma L1, che è la somma dei coefficienti assoluti. L'equazione di una regressione LASSO è molto simile a quella della regressione Ridge:

$$\widehat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{b}} \sum_{i=1}^{n} (y_i - x_i b)^2 + \lambda \sum_{k=1}^{K} |b_k|, \tag{2.3}$$

La penalità ha l'effetto di forzare alcune delle stime dei coefficienti a essere esattamente uguali a zero. Ciò significa che il LASSO può anche essere visto come un'alternativa ai metodi di feature selection per eseguire la selezione delle variabili al fine di ridurre la complessità del modello. Quando λ è piccolo, il risultato è molto vicino alla stima dei minimi quadrati. All'aumentare di λ, si verifica una contrazione in modo da poter eliminare le variabili che sono a zero. Un ovvio vantaggio della regressione LASSO rispetto alla Ridge, è che produce modelli più semplici e più interpretabili che incorporano solo un insieme ridotto di predittori.

Analisi e osservazioni

I dati forniti dai modelli in esame sono risultati particolarmente significativi per arrivare alla deduzione di quanto possa incidere un'analisi con regolarizzazione LASSO. E' stata utilizzata la stessa griglia di valori per λ utilizzata nella regressione Ridge, così come è stata realizzata la cross-validation per stimare il λ migliore.
Il valore dell'MSE al crescere del fattore λ viene rappresentato nel grafico seguente:

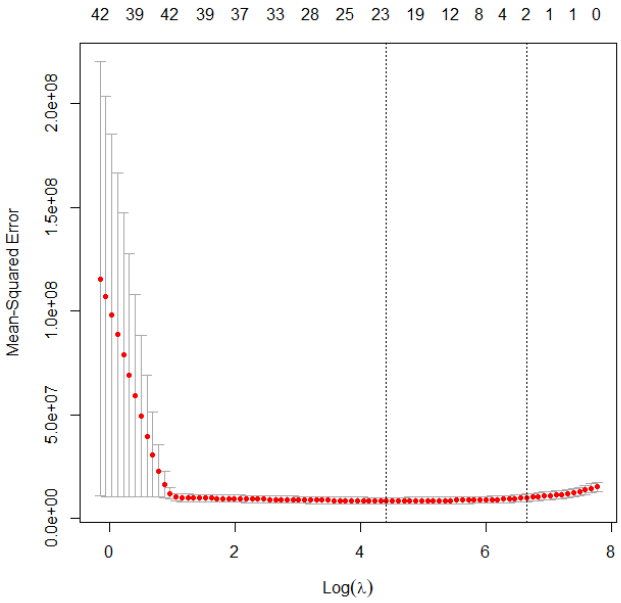


Figura 2.18: Variazione dell'MSE sul test set in relazione a λ.

In quest'altra rappresentazione, invece, vengono messe in risalto le curve che disegnano i coefficienti in funzione della misura di λ, presente sulle ascisse del grafico:

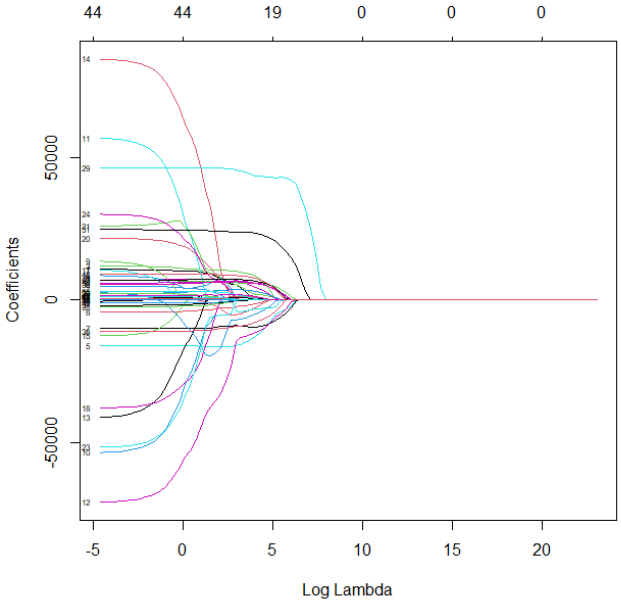


Figura 2.19: Variazione dei coefficienti con differenti valori di λ.

Dal grafico si può notare come, man mano che λ cresce, il termine di regolarizzazione ha un effetto maggiore e quindi vedremo sempre meno variabili per questo modello perché sempre più coefficienti vengono portati a 0.

Di seguito viene riportata una tabella contenente i migliori valori di λ per ogni modello e i coefficienti che vengono portati a zero:

Trasformazione	Miglior lambda	Coefficienti = 0
Lineare	4.507449	5
Polinomiale di 2° ordine	4.228348	8
Polinomiale di 3° ordine	4.972618	16
Polinomiale di 4° ordine	4.414415	20

Tabella 2.7: Tabella dei λ e dei coefficienti pari a 0.

Infine, viene mostrata una tabella riassuntiva contenente i test MSE stimati con il Validation Set Approach per ogni modello con e senza la regolarizzazione.

Trasformazione	MSE senza regolarizzazione	MSE Ridge	MSE LASSO
Lineare	8714612	8692656	8774099
Polinomiale di 2° ordine	8897057	8719061	8721803
Polinomiale di 3° ordine	8226480	8214007	8246507
Polinomiale di 4° ordine	8997441	8398308	8398355

Tabella 2.8: Tabella di confronto degli MSE di Ridge e LASSO.

Un ultimo aspetto da mettere in risalto riguarda la differenza di valori conseguiti con i processi Ridge e LASSO. Si può notare che LASSO produce risultati leggermente più alti rispetto a Ridge. Quest'ultimo produce modelli molto competitivi e comparabili con quelli senza regolarizzazione, con il sostanziale beneficio di ridurre i coefficienti a zero e pertanto produrre modelli molto più interpretabili. Inoltre, nel caso polinomiale di 4° ordine si può notare un grande miglioramento.

2.6 Riduzione della dimensionalità

Tutti i metodi discussi finora controllano la varianza in due modi diversi, o usando un sottoinsieme di variabili, oppure portando i loro coefficienti a 0. Tutti questi metodi sono però definiti usando i predittori originali. La Dimension Reduction, invece, è un insieme di approcci che trasformano i predittori ottenendo un insieme più piccolo di quello iniziale e poi addestrano il modello usando i minimi quadrati e questi predittori trasformati, ottenendo quindi un problema di dimensione minore.

2.6.1 PCR

PCR (Principal Component regression) è una tecnica di regressione basata sull'analisi delle componenti principali (PCA); consiste nel costruire M componenti principali da p predittori e utilizzarli come nuovi predittori in un modello di regressione lineare. Poiché le componenti principali cercano di preservare la variabilità dei predittori, l'assunzione di base, non necessariamente vera, è che questo preservi al massimo anche la relazione con Y. Tuttavia, la PCR non è una tecnica di selezione delle variabili, poiché usa sempre tutti i predittori originali nella costruzione delle combinazioni lineari. La tecnica risulta vantaggiosa quando poche componenti riescono a riassumere una parte sostanziale della variabilità totale nei p predittori. Il parametro M di variabili da usare nella PCR è tipicamente scelto con cross-validation.

Analisi e osservazioni

Lo sviluppo dell'analisi sulla tecnica PCR ha previsto gli stessi passi affrontati per i metodi precedentemente esaminati, sono state infatti sperimentate tutte le trasformazioni considerate, partendo da quella lineare e giungendo a quella polinomiale di 4° grado. Ancora una volta, si è tenuto traccia dei risultati prodotti di volta in volta, dedicando la giusta attenzione al confronto tra di essi. Sempre in supporto, sono stati generati dei grafici che mostrassero l'andamento dei parametri essenziali alla comprensione dell'attendibilità del modello.
A titolo di esempio, verranno presentate delle tabelle riassuntive e dei grafici per il modello relativo alla trasformazione di quarto grado. Per utilizzare questa tecnica nell'ambiente di sviluppo RStudio, è necessario utilizzare la funzione **pcr()** presente nel package pls. Effettuando il **summary()** dell'oggetto restituito dalla funzione precedentemente citata, vengono mostrati due risultati principali, vale a dire "Root MSE" e la percentuale di varianza spiegata utilizzando n componenti.
Per impostazione predefinita, la funzione **pcr()** calcola il Root MSE, ma è possibile specificare di mostrare l'MSE come possiamo vedere nel grafico seguente:

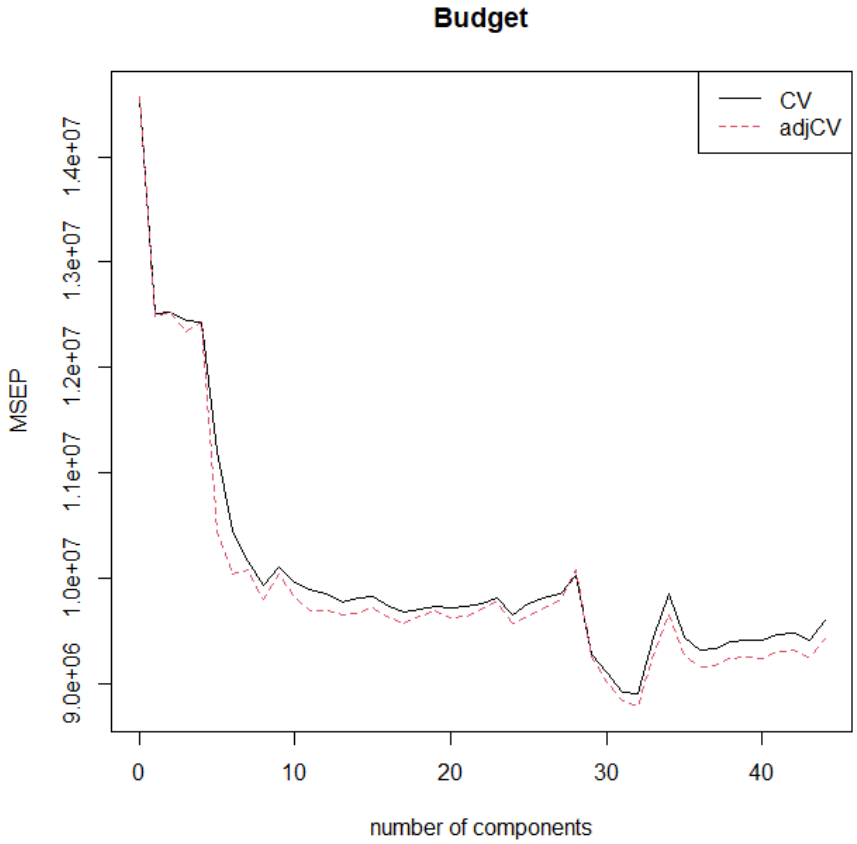


Figura 2.20: Grafico dell'MSE per la tecnica PCR.

Come si può notare, l'MSE minimo si può ottenere considerando 32 predittori, mentre la dimensionalità originaria era 44.
Di seguito viene riportata una tabella contenente un resoconto dei risultati ottenuti con tutti i modelli considerati applicando il Validation Set Approach per la stima dell'MSE sul test set:

Trasformazione	MSE senza riduzione	MSE con riduzione	Num componenti
Lineare	8714612	8718149	7
Polinomiale di 2° ordine	8897057	8989853	16
Polinomiale di 3° ordine	8226480	10465315	10
Polinomiale di 4° ordine	8997441	8927266	31

Tabella 2.9: Tabella MSE PCR.

Sono stati ricalcolati RMSE e l'MSE, ma i risultati sono stati più o meno analoghi a quelli calcolati con la regressione semplice, poiché non c'è alcun beneficio nel ridurre il numero di componenti da considerare.
Inoltre, utilizzando una trasformazione del quarto ordine dei predittori, la percentuale della varianza spiegata sulla variabile Budget risulta essere il 51.88% considerando tutti i predittori.
Per tutte le trasformazioni realizzate per la costruzione di un modello predittivo, usando la PCR i risultati sono analoghi alla regressione semplice, dunque la tecnica di riduzione della dimensionalità non è risultata efficace.

2.6.2 PLS

A differenza della PCR, che non considera la relazione esistente tra $X_1, ..., X_p$ e Y nella costruzione delle variabili $Z_1, ..., Z_M$, il metodo PLS (Partial Least Squares) cerca di considerare questo aspetto. Dunque, la PCR può essere vista come una tecnica di *statistical learning unsupervised*, mentre il metodo PLS è una tecnica *supervised*.

Analisi e osservazioni

Le osservazioni a cui si è giunti durante il procedimento PLS, permettono di fare alcune considerazioni fondamentali. Prima di tutto, è opportuno sottolineare come l'applicazione della tecnica PLS è stata realizzata per tutte le trasformazioni in esame. A rafforzare le conseguenze espresse dai risultati, sono state realizzate delle rappresentazioni che tenessero conto delle variazioni dei singoli parametri presi in esame. Queste rappresentazioni non sono altro che tabelle e grafici riassuntivi che verranno presentati di volta in volta.
A titolo di esempio, verranno raffigurati solamente risultati caratterizzanti la trasformazione del polinomio di grado quattro. In RStudio, per poter realizzare questa tecnica di regressione, è necessario utilizzare la funzione **plsr()** presente nel package pls. Il **summary()** prodotto è lo stesso del PCR, ovvero, è presente il cross-validation del Root MSE e la varianza spiegata al variare del numero di componenti. Anche in questo caso c'è un validation plot per capire il numero di componenti che minimizza l'MSE e il RMSE.

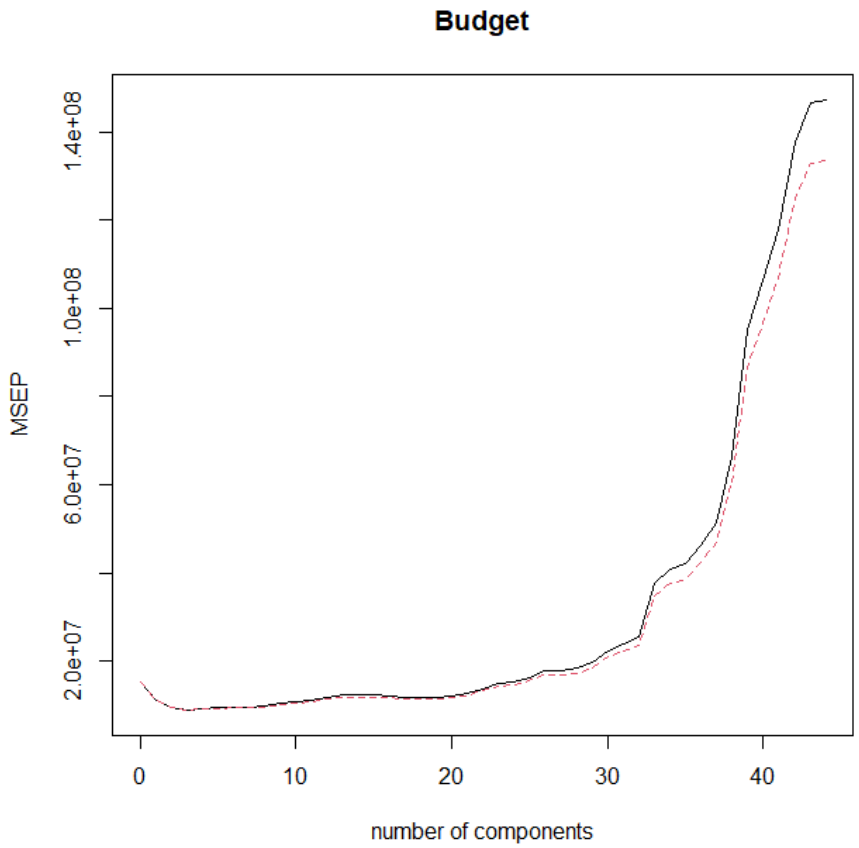


Figura 2.21: Grafico dell'MSE per la tecnica PLS.

Come si può notare, l'MSE minimo si può ottenere considerando 3 predittori, mentre la dimensionalità originaria era 44.

Di seguito viene riportata una tabella contenente un resoconto dei risultati ottenuti con tutti i modelli considerati applicando il Validation Set Approach per la stima dell'MSE sul test set:

Trasformazione	MSE senza riduzione	MSE con riduzione	Num componenti
Lineare	8714612	8739354	3
Polinomiale di 2° ordine	8897057	8886340	3
Polinomiale di 3° ordine	8226480	8886340	3
Polinomiale di 4° ordine	8997441	9018172	3

Tabella 2.10: Tabella MSE PLS.

Successivamente viene riportata una tabella contenente, per ogni modello considerato, la proporzione della varianza spiegata dal modello col massimo numero di componenti trovato tramite il CV approach sia per la tecnica PCR che PLS:

Trasformazione	PEV PCR	Componenti PCR	PEV PLS	Componenti PLS
Lineare	38.69	7	38.94	3
Polinomiale di 2° ordine	41.70	16	40.35	3
Polinomiale di 3° ordine	36.44	10	47.30	3
Polinomiale di 4° ordine	49.71	31	48.17	3

Tabella 2.11: Tabella di confronto della varianza spiegata (Prediction Error Variance).

Come si può vedere, il vantaggio della tecnica PLS, rispetto alla PCR, è quello di riuscire a spiegare una maggiore varianza della Y con un minor numero di componenti, come mostra la tabella 2.11.
Invece di guardare soltanto i regressori e minimizzare la proiezione dei soli regressori in uno spazio più piccolo, adesso si coinvolge anche la variabile Y. Quindi si cerca di ottimizzare la variabilità di X e della Y stessa, infatti i risultati cui si è giunti sono i seguenti:

- Il numero di predittori considerati dalla PLS è inferiore rispetto a quelli considerati dalla PCR.
- La massima varianza spiegata dai predittori si raggiunge attraverso un minor numero di variabili.
- Per quanto riguarda l'MSE, i suoi valori sono in linea con quelli trovati con Ridge e LASSO.

Capitolo 3

Progetto Python

In questa sezione verranno analizzati i risultati ottenuti tramite l'applicazione di diversi metodi di classificazione. Per lo sviluppo di questa parte, è stato utilizzato l'ambiente Google Colab.
L'obiettivo preposto è stato quello di predire e stimare il parametro (o classe) *class* (rappresentata da uno 0 per le banconote false e da un 1 per quelle autentiche) utilizzando come parametri di classificazione tutti quelli risultanti dalla fase di preprocessing dei dati, ovvero: *variance, skewness, curtosis, entropy.* Per la classificazione, sono stati utilizzati i metodi analizzati durante il corso, quali Naïve Bayes, KNN (K-Nearest Neighbors), Naïve Kernel e Logistic Regression. Per gli ultimi tre, è stato utilizzato anche il framework Apache Spark il quale, in generale, permette la parallelizzazione dei compiti. Ciò ha richiesto un'implementazione manuale dei suddetti algoritmi, le quali funzioni sono state appunto parallelizzate attraverso Spark.

3.1 Analisi dei dati

La parte iniziale del progetto è stata dedicata, anche per questo dataset, alla pre-elaborazione dei dati, fondamentale per una buona analisi.
Come abbiamo già detto prima il dataset in questione contiene 1372 istanze con 5 attributi. La classe binaria è composta da 762 elementi etichettati dal valore 0 e quindi 610 valori opposti.

3.1.1 Modifiche applicate

Bilanciamento dataset

Dal momento che il dataset risulta non bilanciato, la prima operazione è stata proprio quella di eguagliare la frequenza dei valori binari. Dopo questa operazione il numero di istanze totali è pari a 1220.

Normalizzazione

A seguito di questo intervento è stato necessario applicare una normalizzazione dei dati per la presenza di valori disposti su diverse scale per le varie features. Si è deciso dunque di normalizzare e portare tutti i valori su una scala da 0 a 1. Nella pagina seguente è possibile vedere il procedimento di normalizzazione dei campioni effettuato tramite codice python in Colab.

```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

data=pd.DataFrame(scaler.fit_transform(balanced),
                  columns=balanced.columns, index=balanced.index)
```

Figura 3.1: Codice utilizzato per normalizzare il dataset.

	variance	skewness	curtosis	entropy	class
0	-3.3553	0.35591	2.6473	-0.37846	1
1	3.1887	-3.41430	2.7742	-0.20260	0
2	-1.1005	-7.25080	6.0139	0.36895	1
3	1.4806	7.63770	-2.7876	-1.03410	0
4	-2.3361	11.96040	3.0835	-5.44350	0

	variance	skewness	curtosis	entropy	class
0	0.265871	0.528687	0.341758	0.742859	1.0
1	0.737786	0.387611	0.347225	0.758850	0.0
2	0.428474	0.244055	0.486786	0.810820	1.0
3	0.614607	0.801161	0.107631	0.683243	0.0
4	0.339369	0.962911	0.360549	0.282304	0.0

Figura 3.2: Prima (sinistra) e dopo (destra) la normalizzazione del dataset.

3.1.2 Parametri statistici dei dati

Per consentire la generazione di un dataset sintetico, la cui analisi fornisce un supporto all'analisi effettuata sul dataset reale, sono stati estrapolati i parametri statistici dei dati. In particolare, sono state calcolate la media e la varianza delle features, e la covarianza tra coppie di features. Questi parametri sono stati valutati sia per i dati appartenenti alla classe 0 (banconote contraffatte) sia per i dati appartenenti alla classe 1 (banconote autentiche). Segue una figura che rappresenta la visualizzazione della media e della varianza delle features (*curtosis* e *variance*) sia per banconote false che per quelle valide. Da un primo sguardo, notiamo che la *curtosis* è sostanzialmente più alta in media nelle banconote autentiche e che, al contrario, quelle originali hanno in media un valore di *variance* più basso delle false.

```
For class 0 (Non valid), the curtosis mean is 0.2625 with variance 0.0193
For class 1 (Valid), the curtosis mean is 0.3203 with variance 0.0514

========================================================================

For class 0 (Non valid), the variance mean is 0.6758 with variance 0.0212
For class 1 (Valid), the variance mean is 0.3731 with variance 0.0184
```

Per verificare eventuali caratteri di indipendenza tra le features presenti nel dataset, si è deciso di calcolare la matrice delle covarianze ed osservare il comportamento di ognuna rispetto a tutte le altre. Segue, quindi, una figura che rappresenta proprio la matrice appena descritta.
Da un primo sguardo possiamo già notare come, indipendentemente dal segno, molti valori sono vicini allo zero, ricordando che se due features sono statisticamente indipendenti il loro valore di covarianza sarà proprio nullo.

	variance	skewness	curtosis	entropy
variance	0.042710	0.012282	-0.014945	0.011082
skewness	0.012282	0.048276	-0.033210	-0.021305
curtosis	-0.014945	-0.033210	0.036167	0.011071
entropy	0.011082	-0.021305	0.011071	0.035973

Figura 3.3: Matrice di covarianza.

3.1.3 Generazione Dataset sintetico

Per l'analisi dei dati, è stato inizialmente utilizzato un approccio basato su un dataset sintetico, al fine di valutare (anche visivamente) la distribuzione dei dati. La caratteristica del dataset sintetico è appunto quella di essere un dataset con le stesse proprietà statistiche dei dati reali. In questo caso è stato ricavato da solo due campi del dataset reale. I campi selezionati possono essere modificati cambiando il valore assegnato alle variabili nel codice sorgente.
Per generare tale dataset si è supposto che la distribuzione dei valori seguisse una normale multivariata. Sono stati quindi calcolati i valori di media, varianza e covarianza a partire dal dataset reale come descritto nel paragrafo precedente. Utilizzando il metodo *multivariate_normal*, messo a disposizione dalla libreria *numpy*, vengono generati i campioni secondo tale distribuzione.
Data la natura disomogenea tra dati discreti e continui che caratterizza le features, non è stato possibile rappresentarli in maniera chiara. Pertanto, la realizzazione di un dataset sintetico, a partire dalle stime di sue features, ha consentito di plottare le distribuzioni dei dati, ed in particolare delle due features scelte, sia in 2D che in 3D.
Qui di seguito è mostrato il grafico associato al dataset sintetico generato a partire dalle features *curtosis e variance*:

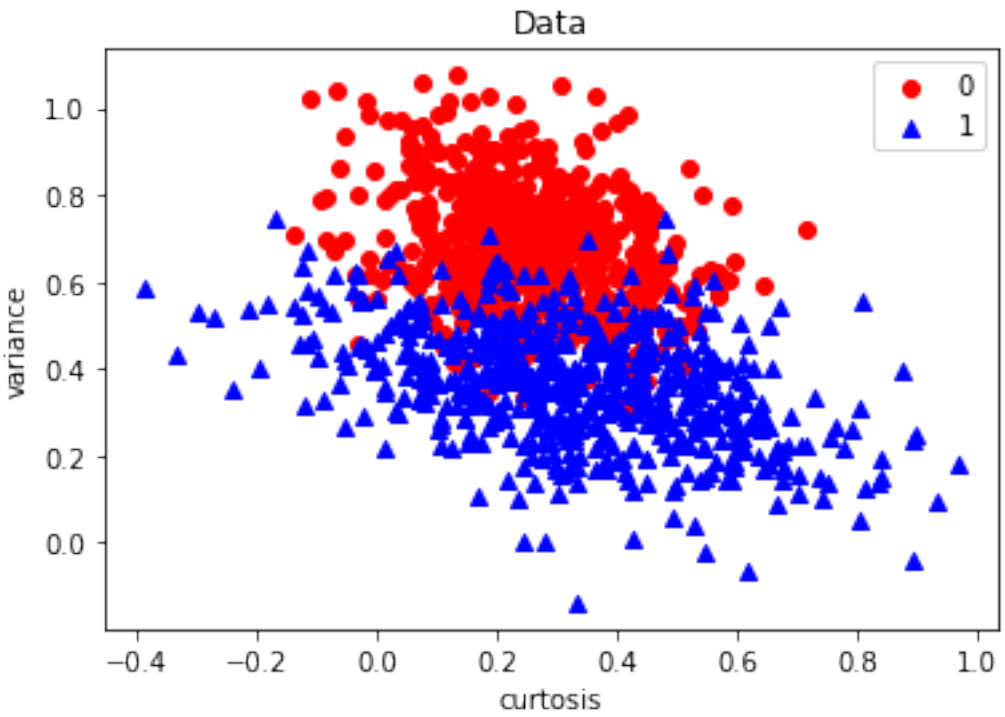


Figura 3.4: Plot Dataset sintetico per 2 features.

3.2 Naïve Bayes

Il classificatore Naïve Bayes utilizza il teorema di Bayes per calcolare la probabilità di osservare una classe data la probabilità a posteriori, cioè le osservazioni delle features. Successivamente applica il criterio MAP (Maximum A Posteriori) o massima probabilità a posteriori, vale a dire che è scelta quella classe che massimizza la probabilità a posteriori. In particolare, il classificatore suppone che le features siano tra loro indipendenti, dunque la distribuzione congiunta delle features condizionata all'osservazione di una classe è calcolata come il prodotto delle distribuzioni marginali delle singole features.
In sostanza, dunque, il classificatore Naïve Bayes è caratterizzato dall'ipotesi di indipendenza.

3.2.1 Indipendenza tra le features

Data l'assunzione di indipendenza tra le features fatta dal classificatore, bisogna verificare che questa possa essere assunta anche nel caso in esame. A tal proposito, è possibile consultare la matrice di covarianza e quella di correlazione tra tutte le features.
Sono state graficate le distribuzioni dei dati, per le due features *curtosis e variance*, sia considerando il dataset sintetico, generato a partire dalle distribuzioni del dataset reale, sia considerando una stima della media e della varianza sui campioni del dataset sintetico stesso. In questo secondo caso si forza l'ipotesi di indipendenza e si assume una distribuzione normale.
Sono stati realizzati i grafici delle distribuzioni sia in 2D che in 3D. Le due rappresentazioni riportano le stesse informazioni, ma in due formati differenti. Il primo grafico rappresenta la distribuzione delle due features sul dataset sintetico, mentre il secondo la distribuzione in cui si è data per vera l'ipotesi di indipendenza.

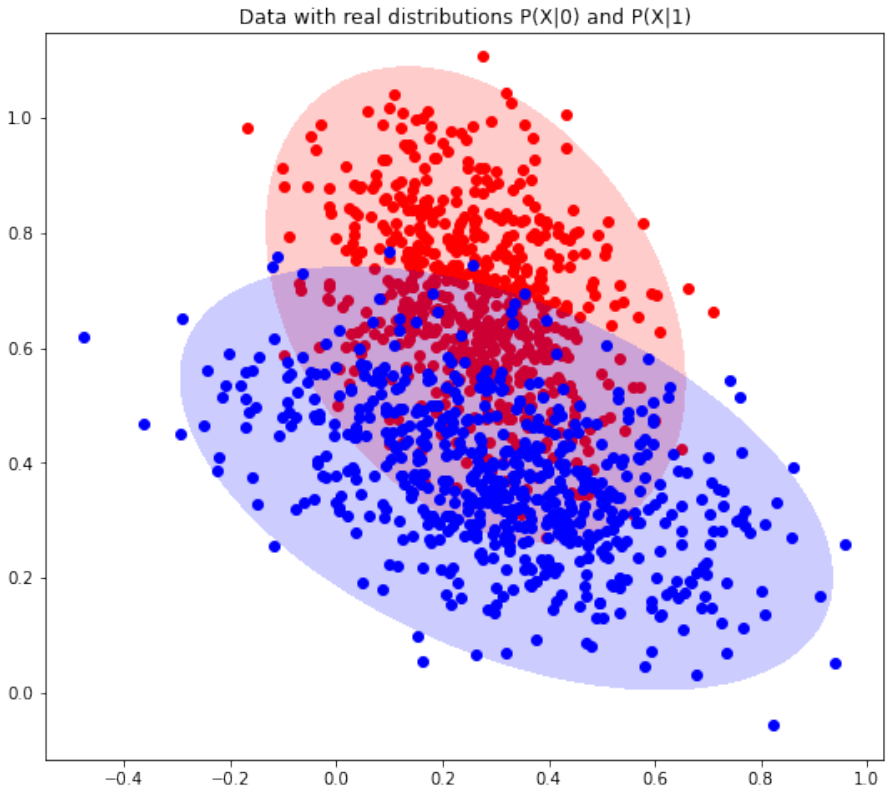


Figura 3.5: Campioni Dataset sintetico con distribuzione stimata del Dataset reale.

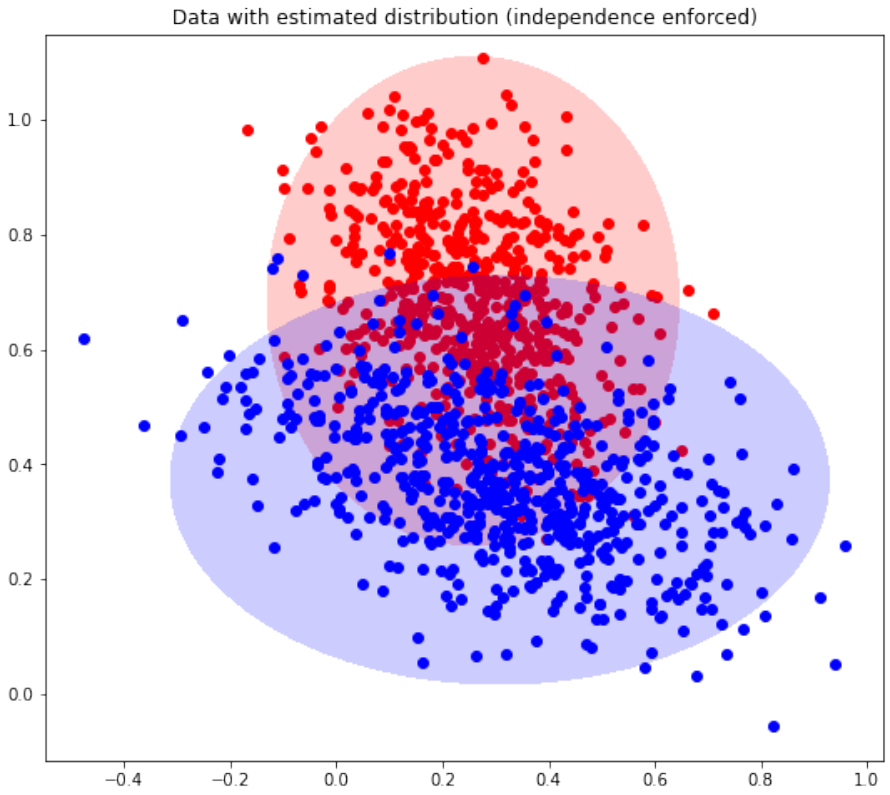


Figura 3.6: Campioni Dataset sintetico con distribuzione stimata del Dataset sintetico.

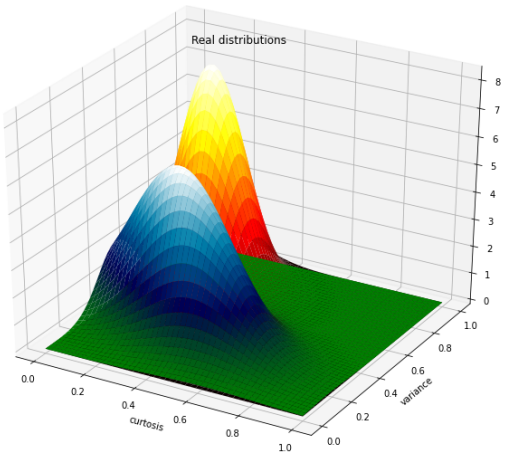


Figura 3.7: Distribuzione 3D reale delle due features selezionate.

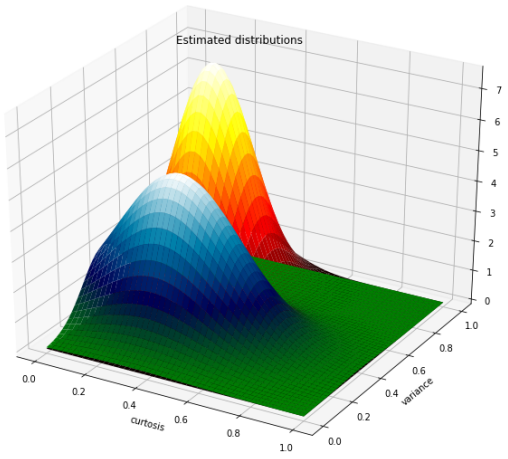


Figura 3.8: Distribuzione 3D stimata dal dataset sintetico

I due plot in 3D appena mostrati sono stati generati dalla funzione *plot_gaussian_3D* presente nella libreria matplotlib che consente appunto di graficare le distribuzioni appena discusse in tre dimensioni. Da questi plot possiamo dunque stimare che, forzando l'ipotesi di indipendenza tra le due features, continuiamo comunque ad avere dati ben separabili.
Osservando i grafici in 2D, invece, le distribuzioni tendono a sovrapporsi e quindi, la correlazione tra le features assume un'importanza rilevante. Le due features *curtosis e variance*, presentano una correlazione pari a circa -0.380265, il che fa presupporre che queste non siano stocasticamente indipendenti.
Come è lecito aspettarsi, il classificatore Naïve Bayes, non può esibire delle prestazioni troppo lontane dall'ottimo nel caso dei dataset sintetici, mentre per quanto riguarda il dataset reale, in cui la distribuzione dei campioni potrebbe non essere normale, si potrebbero avere delle performances lontane dall'ottimo, almeno che queste non risultino indipendenti.

3.2.2 Naïve Bayes su dataset sintetico con due features

```
Confusion matrix:
[[163  27]
 [ 19 157]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       0.90      0.86      0.88       190
         1.0       0.85      0.89      0.87       176

    accuracy                           0.87       366
   macro avg       0.87      0.87      0.87       366
weighted avg       0.88      0.87      0.87       366


Precision Score:
0.8532608695652174


final performance:  87.43%
```

Come si può notare le performances sono intorno all' 87% il che dipende dal fatto che il classificatore non tiene in considerazione la correlazione tra le due features.

3.2.3 Naïve Bayes su dataset sintetico usando tutte le features

```
Confusion matrix:
[[1990  245]
 [ 235 1922]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       0.89      0.89      0.89      2235
         1.0       0.89      0.89      0.89      2157

    accuracy                           0.89      4392
   macro avg       0.89      0.89      0.89      4392
weighted avg       0.89      0.89      0.89      4392


Precision Score:
0.8869404706968159


final performance:  89.07%
```

I risultati sono nettamente migliori rispetto a quelli ottenuti sull'intero dataset reale: questo è dovuto al fatto che il dataset utilizzato per tali analisi contiene un numero di campioni maggiore.

3.2.4 Naïve Bayes su dataset reale con due features

In questo caso viene utilizzata la porzione del dataset composta unicamente dalle features *curtosis e variance*. I risultati sono leggermente inferiori rispetto a quelli riscontrati utilizzando il dataset sintetico, come mostrato nella figura seguente.

```
Confusion matrix:
[[166  31]
 [ 28 141]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       0.86      0.84      0.85       197
         1.0       0.82      0.83      0.83       169

    accuracy                           0.84       366
   macro avg       0.84      0.84      0.84       366
weighted avg       0.84      0.84      0.84       366


Precision Score:
0.8197674418604651


final performance:  83.88%
```

Dal momento che la correlazione tra le due features risulta circa -0.38, le seguenti performances sono coerenti.

3.2.5 Naïve Bayes su dataset reale usando una feature per volta

Al fine di comparare l'impatto di ogni singola feature sulla percentuale di corretta classificazione, il classificatore Naïve Bayes è stato testato sulla porzione del dataset composta da ciascuna singola feature, ottenendo il seguente risultato:

variance	skewness	curtosis	entropy
87.43%	60.11%	60.11%	49.45%

Come si può notare la feature *variance* garantisce una buona performance nel caso questa venga impiegata singolarmente per fare predizioni.

3.2.6 Naïve Bayes su dataset reale usando tutte le possibili coppie di features

Si è ripetuta la stessa analisi, ma questa volta su tutte le possibili coppie di features.

variance + skewness	variance + curtosis	variance + entropy	skewness + curtosis	skewness + entropy	curtosis + entropy
89.62%	81.15%	84.70%	58.74%	62.57%	59.02%

Si può notare come la presenza della feature *variance* comporta un aumento delle performances, in linea con quanto già detto precedentemente.

3.2.7 Naïve Bayes su dataset reale usando tutte le features disponibili

```
Confusion matrix:
[[101  82]
 [ 93  90]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       0.52      0.55      0.54       183
         1.0       0.52      0.49      0.51       183

    accuracy                           0.52       366
   macro avg       0.52      0.52      0.52       366
weighted avg       0.52      0.52      0.52       366


Precision Score:
0.5232558139534884


final performance:  52.19%
```

I risultati ottenuti sono nettamente inferiori rispetto a quelli ottenuti considerando le sole due features *curtosis e variance*. Questo potrebbe essere dovuto al fatto che, l'assunzione di indipendenza tra le features non risulta essere verificata.
Inoltre, l'aumento del numero di features, inevitabilmente, richiederebbe un corrispondente aumento delle istanze, il cui numero però è rimasto invariato.

3.3 KNN

Il classificatore K-Nearest Neighbor parte dall'idea che dati appartenenti ad una stessa classe siano raggruppati tra loro e separati da quelli appartenenti ad altre classi. Pertanto, un nuovo dato x verrà assegnato alla classe più frequente tra i K dati più vicini a x. Quindi, un iperparametro di questo classificatore è appunto il numero di vicini K da utilizzare nell'algoritmo.
Un primo vantaggio del KNN è che si tratta di un approccio totalmente non parametrico, dato che non si fa nessuna assunzione sulla forma delle soglie di decisione. L'algoritmo del KNN "soffre" in maniera alquanto importante quando si hanno molti predittori: al crescere del numero di questi, l'algoritmo, necessita di una corrispondente crescita (non lineare) del numero di osservazioni necessarie per ottenere delle previsioni attendibili (Curse of Dimensionality).

3.3.1 Scelta di K

Il punto fondamentale nell'applicazione del KNN è la scelta del parametro K, ovvero il numero di osservazioni più prossime da includere nel "vicinato" di un dato punto. Quando K è piccolo, il classificatore produce soglie di decisione molto flessibili. Man mano che K cresce, il metodo diventa meno flessibile.
Altre analisi possono essere condotte sulla dimensione di K: un valore di K piccolo rende l'algoritmo più locale, mentre un valore di K più grande comporta la perdita di località. Anche se, al crescere del numero di campioni N, K deve necessariamente crescere. Questo, però, non significa perdita di località; infatti, i punti tendono ad addensarsi nell'intorno della X di test. A seguito di queste considerazioni, è stato previsto un opportuno algoritmo di tuning per il parametro K.

3.3.2 Distanza

La scelta di una metrica di distanza piuttosto che un'altra, può influenzare l'accuratezza della classificazione. Per classificare, l'algoritmo calcola le "distanze" tra le X di test e le X nel training set, e sceglie i K campioni più vicini.
Si è scelto di utilizzare la distanza Euclidean, perchè questa performa in maniera ragionevolmente migliore su dati categorici e numerici.

3.3.3 Divisione del dataset

Tale metodo di classificazione è piuttosto oneroso da un punto di vista computazionale e richiede la regolazione dell'iperparametro K. A tal fine è stata eseguita una partizione del dataset in 3 parti:

- Training set (50%): usato per l'estrazione dei K più vicini ad un nuovo sample;
- Validation set (20%): usato per ottimizzare il parametro K, valutando su di esso l'MSE, e selezionando il valore di K per il quale il valore di errore è più piccolo;
- Test set (30%): per la valutazione delle performances finali in termini di corretta classificazione.

Tale divisione è stata realizzata tramite la funzione *train_test_val_split* che, prendendo in input le percentuali di grandezza dei 3 set, li restituisce opportunamente divisi.

3.3.4 Implementazione realizzata

Sono state implementate due funzioni per l'addestramento e la predizione:
KNN_fit_and_predict_spark: effettua l'addestramento e la predizione utilizzando Spark, prende come parametri di input il training set, le X di test ed il valore K da utilizzare. In questo caso, il training set viene parallelizzato, e su ogni partizione vengono individuati i K vicini più prossimi, per poi aggregare i risultati.
KNN_fit_and_predict_no_spark: effettua l'addestramento e la predizione usando il classificatore KNN messo a disposizione dalle librerie sklearn. I parametri di input sono il training set, le X di test ed il valore K da utilizzare.
I risultati tra le due funzioni precedenti possono essere comparati utilizzando la funzione **KNN_fit_and_predict_comparison**. Nel caso questi non combacino, viene automaticamente generata un'eccezione, che comporta la fine dell'esecuzione, altrimenti ritorna la Y predetta.
KNN_plot viene invece utilizzata per plottare i risultati della classificazione, realizzando una meshgrid, e facendo predizioni per tali valori.
Il grosso del lavoro viene realizzato dalla funzione **KNN**, che, sulla base di come sono settati i parametri di input, effettua o meno la comparazione tra le predizioni fatte utilizzando o non Spark. Le predizioni vengono fatte sul Validation set. Quando si è certi che queste combaciano, il che è certificato dal fatto che nessuna eccezione è stata generata, si possono impiegare le funzioni di libreria sklearn. Utilizzando queste, vengono determinate le performances sul Test set e l'MSE sul Validation set. Quest'ultimo sarà impiegato in fase di tuning del parametro K. Il tutto si conclude con la stampa dei risultati ottenuti dal classificatore con le aree di predizione, delle performances sul test set, ma anche delle altre metriche impiegate, come la matrice di confusione.
KNN_tuning: effettua l'ottimizzazione del parametro K. Tale funzione prende in input un massimo valore per K ed il numero di valori da testare. Per ogni K vengono determinati MSE di Validation e Test performances. Alla fine si sceglie il valore di K che determina il più basso MSE, e di conseguenza le corrispondenti Test performance. In ultimo viene visualizzata la relazione tra Validation MSE e K.
In seguito si riportano i risultati ottenuti andando a realizzare il tuning di K con un massimo valore pari a 300, ed un numero di valori testati pari a 50.

3.3.5 Spark

Per poter utilizzare correttamente il framework Spark è stato necessario implementare nuovamente le funzioni utilizzate dal KNN, quali quella del calcolo della distanza euclidea e della determinazione dei K elementi più vicini. La classe di campione del test set viene assegnata secondo quella più frequente tra i suoi K elementi più vicini, senza pesare ogni classe in base alla distanza rispetto a x. Tali scelte si sono ripetute anche nel fit tramite le funzioni di libreria. Pertanto si è deciso di impiegare lo stesso metodo di determinazione della classe e di calcolo delle distanze, per ottenere risultati identici.
E' stata effettuata la parallelizzazione del dataset, il broadcast del parametro K e dei punti del Test Set. Quando ogni cluster ha terminato il calcolo delle distanze tra i campioni della propria partizione e l'elemento x, e dopo aver restituito i K elementi più vicini, viene effettuata una nuova parallelizzazione dei risultati per il calcolo della classe più frequente. Dato che il problema di classificazione è binario e che le due classi sono espresse come 0 o come 1, è stato sufficiente eseguire la somma delle classi associate ai K elementi. La somma ottenuta viene quindi confrontata con il valore K: se tale somma è maggiore di K/2, x viene assegnato alla classe 1; altrimenti, x viene assegnato alla classe 0.

3.3.6 KNN su dataset sintetico con due features

Features: *curtosis, variance*.

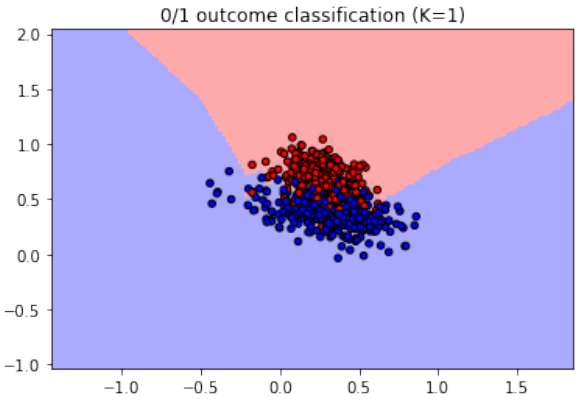


Figura 3.9: Performance 83.61%.

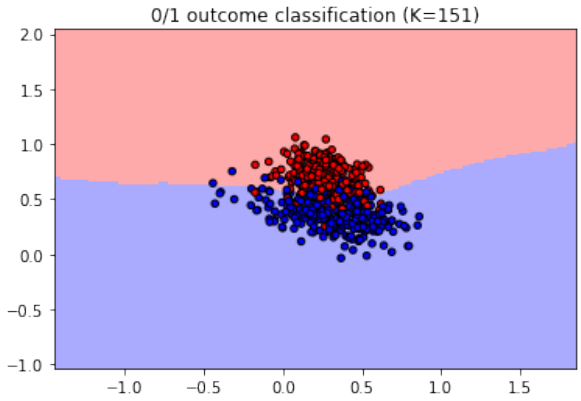


Figura 3.10: Performance 86.89%.

Come si può ben notare per valori piccoli di K, l'algoritmo tende a seguire molto precisamente la divisione tra le due nuvole di punti. All'aumentare del valore di K, invece, si nota una divisione molto più approssimata, tendente ad una divisione lineare delle due classi. Questo perché più aumenta K, più punti si stanno considerando all'interno dell'algoritmo.
In tutti i casi, comunque, si può notare che il modello riesce a separare più o meno correttamente le due regioni. Infatti, oltre ai grafici delle regioni di decisione, lo script grafica anche l'andamento delle prestazioni del modello, in base al valore dell'MSE, indicando quale sia il valore di K che restituisce il risultato migliore. Di seguito viene riportato tale grafico.

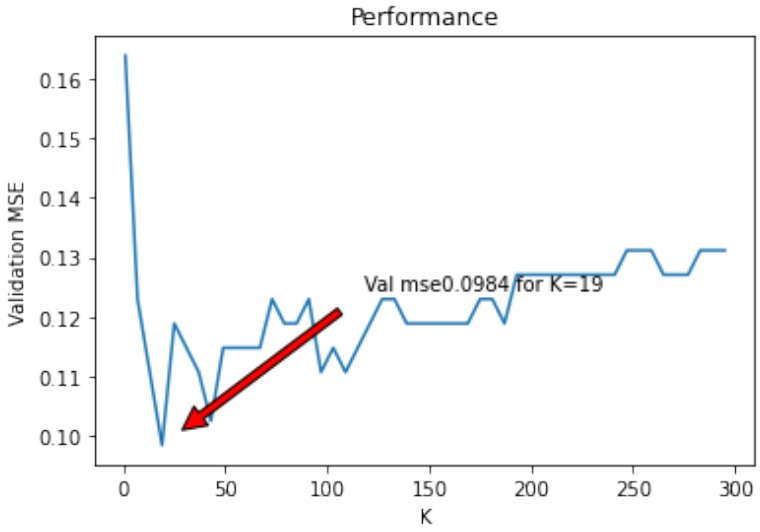


Figura 3.11: Performance migliore: 86.34% per K=19.

É bene notare che le variazioni di MSE tra i diversi valori di K non sono così elevate. Infatti, il test error oscilla intorno al valore 0.12, raggiungendo picchi massimi di 0.131 e raggiungendo il valore minimo per K pari a 19. Tale discorso non è generalizzabile e valido per le altre coppie di features.

3.3.7 KNN su dataset reale con due features

Avendo quindi dimostrato tramite il dataset sintetico che l'implementazione parallelizzata tramite Spark è completamente funzionante e corretta, il classificatore KNN è stato applicato anche al dataset reale utilizzando la sola implementazione di libreria.

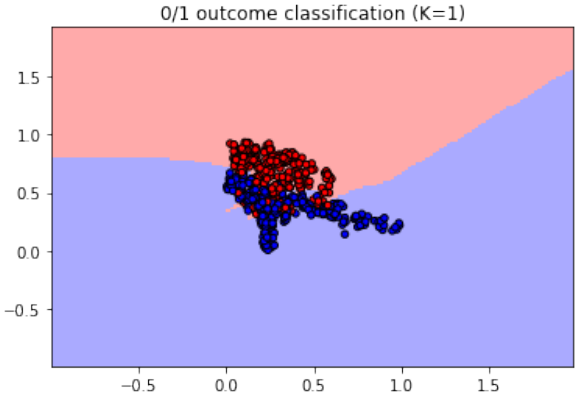


Figura 3.12: Performance 86.61%.

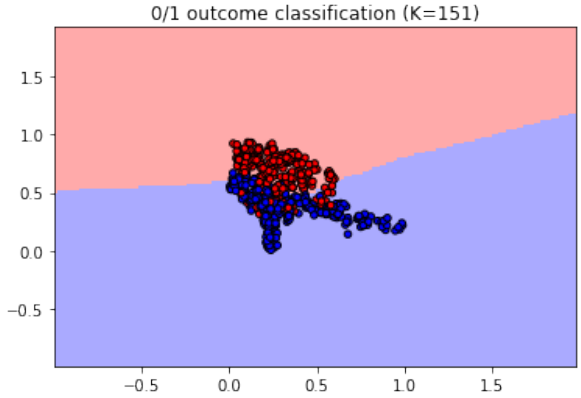


Figura 3.13: Performance 89.07%.

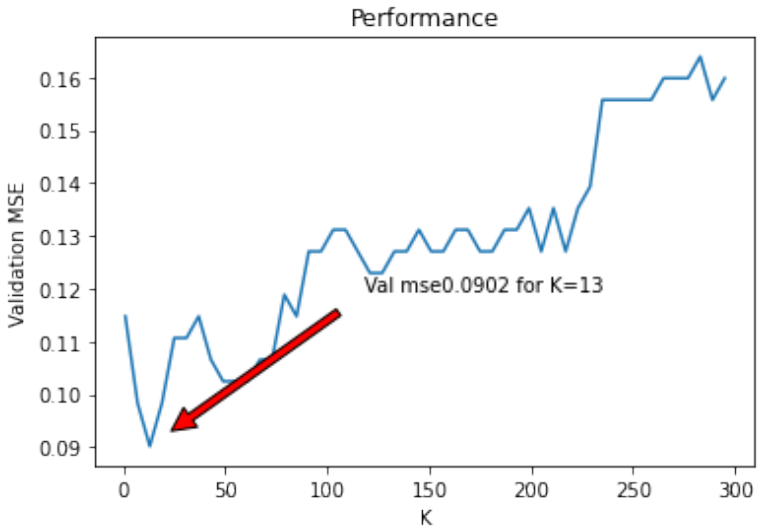


Figura 3.14: Performance migliore: 89.34% per K=13.

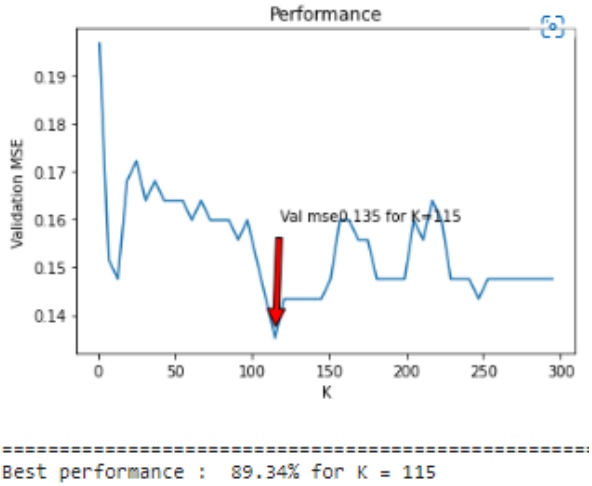
Analizzando i risultati ottenuti, si può notare che con K prossimi ad 1, la divisione tra le due classi sembra essere già abbastanza buona. Infatti si possono riscontrare delle performances che sono vicine ai migliori valori ottenuti: 89.34% per k = 13.
Ancora una volta si può notare che, man mano che K cresce, progressivamente le performances tendono a decrescere fino a decadere al di sotto del 86% quando K è maggiore di 200. Come già detto, abbiamo una perdita di località. Ciò è rafforzato dal fatto che la divisione tra le due classi va via via peggiorando al crescere di K. In questo caso, dalle analisi sull'MSE del Validation set, risulta che il valore di K che garantisce minimo errore è K = 13, leggermente in contrasto con quanto si evince dalle performances sul Test Set, che in questo circostanza risultano pari a 91.26% per K=115.

3.3.8 KNN su dataset reale usando una feature per volta

Allo scopo di verificare quali siano quelle features che maggiormente caratterizzano gli elementi di una classe, è stato deciso di effettuare un fit su ogni singola feature presente nel dataset.

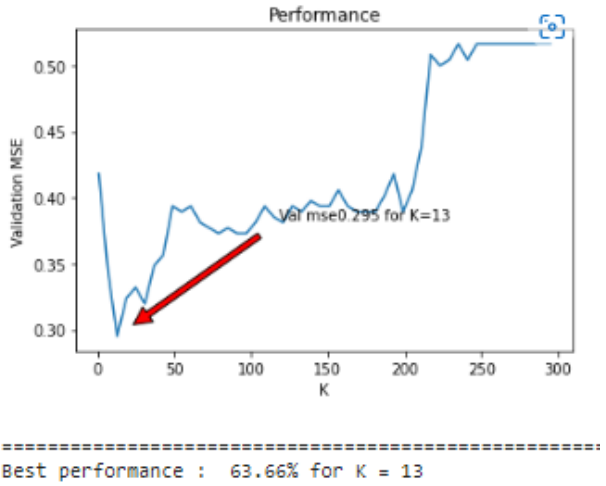
	variance	skewness	curtosis	entropy
K	115	31	13	73
MSE	13.52%	22.54%	29.51%	45.08%
Score	89.34%	76.23%	63.66%	53.83%

Dalla tabella precedente si possono analizzare i risultati ottenuti effettuato l'ottimizzazione di K, facendo addestramento e predizioni con dataset costituiti da singole features. In particolare osserviamo i valori di K che garantiscono il minor MSE sul validation set, ed i valori per le performances. A causa della diversa distribuzione dei dati, i valori di K sono diversi per quasi tutte le features.
E' possibile osservare i grafici di *curtosis e variance*, le variabili tenute in considerazione fino a questo momento, che riportano la relazione tra MSE e K.




```
Best performance :  89.34% for K = 115
```

Figura 3.15: Performance variance.




```
Best performance :  63.66% for K = 13
```

Figura 3.16: Performance curtosis.

3.3.9 KNN su dataset reale usando tutte le features disponibili

```
Performance for K = 1:  100.00%


Confusion matrix:
[[181   0]
 [  0 185]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       181
         1.0       1.00      1.00      1.00       185

    accuracy                           1.00       366
   macro avg       1.00      1.00      1.00       366
weighted avg       1.00      1.00      1.00       366


Precision Score:
1.0


Validation MSE =  0.004098
```

Non risulta possibile riportare i vari grafici poiché il problema non è più bidimensionale.
In questo caso, si evince che le migliori performances sul Test set sono ottenute per K prossime ad 1. Le oscillazioni tra valori di K vicini sono dovute alla variabilità dei dati, l'andamento crescente dell'errore ha una natura diversa: con valori di K troppo grandi, si vanno a considerare troppi elementi. Nel caso in cui il nuovo elemento sia circondato da altri campioni della sua classe vera, questo verrà correttamente classificato. Se invece il nuovo elemento si trova nell'intersezione tra le diverse nuvole di punti, considerando molti elementi, si vanno a considerare anche quelli dell'altra classe e quindi si effettua un errore di valutazione.

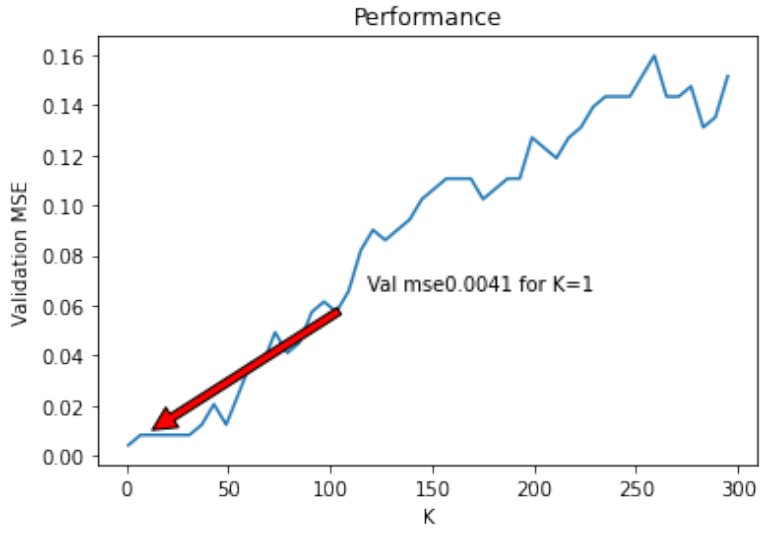


Figura 3.17: Performance migliore: 100.00% per K=1.

3.3.10 KNN su dataset sintetico usando tutte le features insieme

Per motivi computazionali e di tempo, si è scelto di non impiegare l'intero dataset sintetico creato, ma solamente una porzione: in tal senso, sono stati realizzati due test, uno con 1220*2 istanze del dataset sintetico, ed un altro con 1220*3 istanze. Questo per andare ad analizzare le variazioni in performance dei valori di K ottenuti, nel momento in cui viene incrementata la dimensione del dataset.

Numero di campioni N*2

```
Performance for K = 1:  99.04%


Confusion matrix:
[[350   3]
 [  4 375]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       353
         1.0       0.99      0.99      0.99       379

    accuracy                           0.99       732
   macro avg       0.99      0.99      0.99       732
weighted avg       0.99      0.99      0.99       732


Precision Score:
0.9920634920634921


Validation MSE =  0.006148
```

Le migliori performance sul Test set vengono riscontrate quando K è compreso tra 1 e 13, e si attestano intorno al 99%. In ogni caso, per tutti i valori testati, le performances non scendono mai al di sotto del 91%.
In questo caso il valore più basso di MSE viene ottenuto in corrispondenza di K=1, con performance sul Test set pari a circa 99%.

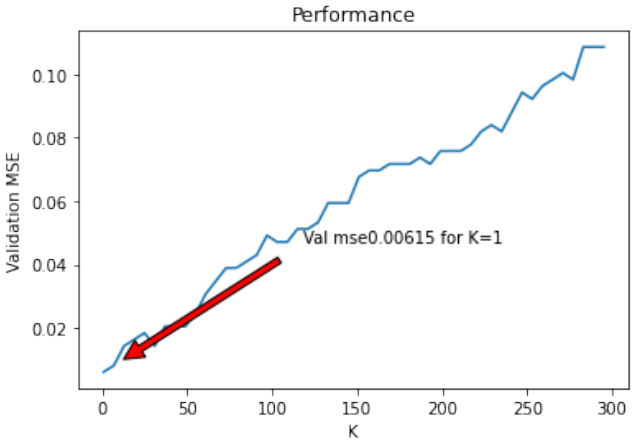


Figura 3.18: Performance migliore: 99.04% per K=1.

Numero di campioni N*3

```
Performance for K = 1:  99.64%


Confusion matrix:
[[517   2]
 [  2 577]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       519
         1.0       1.00      1.00      1.00       579

    accuracy                           1.00      1098
   macro avg       1.00      1.00      1.00      1098
weighted avg       1.00      1.00      1.00      1098


Precision Score:
0.9965457685664939


Validation MSE =  0.01093
```

A differenza di quanto ci saremmo aspettati, un aumento delle dimensioni del dataset, non ha comportato un incremento del valore di K, ma la performance, con più campioni a disposizione, è leggermente migliorata. Questo non vale per il valore dell'MSE sul Validation test che non è più piccolo del caso precedente.
I valori di K che garantiscono il più piccolo MSE di Validazione sono 2: K=1 e K=7 con delle performances di Test rispettivamente del 99.64% e 99.18%.

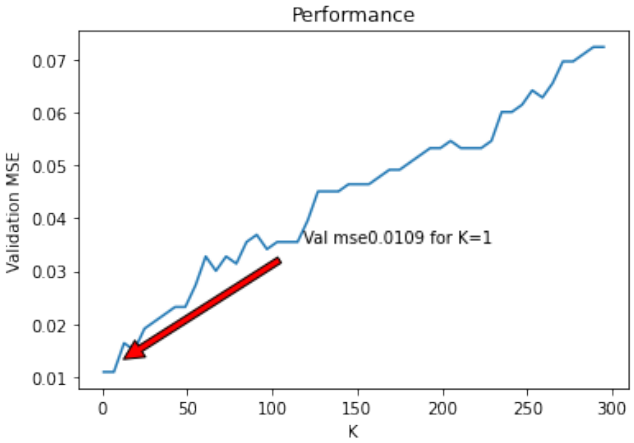


Figura 3.19: Performance migliore: 99.64% per K=1.

Al termine di questa analisi sintetica, è possibile trarre due importanti conclusioni: l'algoritmo KNN è adatto a risolvere questo problema di classificazione restituendo buoni risultati sul dataset sintetico, specialmente su alcune coppie di feature facilmente separabili; l'implementazione manuale dell'algoritmo è corretta e del tutto simile all'implementazione della libreria sklearn, ad esclusione di minori scelte implementative.

3.4 Naive Kernel

Il classificatore Naïve Kernel sfrutta come idea di base la stessa del KNN, ovvero gli elementi di una stessa classe sono raggruppati e divisi dagli altri. La differenza sta nel fatto che non vengono considerati i K elementi più vicini, ma tutti quelli che hanno una distanza minore di una certa soglia, sia tale soglia h. Un nuovo dato verrà assegnato alla classe più frequente tra i dati la cui distanza da esso è minore o uguale a h. Quindi, un iperparametro di questo classificatore è appunto la soglia di distanza h da utilizzare nell'algoritmo.
Anche in questo caso, la definizione di distanza può variare: nel caso in esame, è stata scelta nuovamente la distanza euclidea. Dopo aver ottenuto tali distanze, vengono considerate le classi dei campioni vicini e calcolata la classe più frequente. Tale metodo di classificazione è piuttosto oneroso da un punto di vista computazionale e richiede la regolazione dell'iperparametro h.

3.4.1 Scelta parametro h

Anche in questo caso risulta cruciale la scelta del parametro h. Questo consente di capire quali campioni devono rientrare nel vicinato della X di test che stiamo considerando. In maniera opposta a quanto accadeva per il KNN, in questo caso un incremento del numero di campioni nel dataset, deve necessariamente coincidere con una riduzione del parametro h. Infatti, considerando un certo intervallo, al crescere di N, i campioni tenderanno ad addensarsi in esso. Al contrario, se il dataset risulta povero, il valore di soglia deve essere maggiore, dato che i campioni, a quel punto, non risultano densi in un intervallo. Pertanto, si è optato per una funzione avente l'obiettivo di andare ad ottimizzare, e quindi, a scegliere il parametro h ottimo (*NaiveKernel_tuning()*).

3.4.2 Divisione del dataset

Tale metodo di classificazione è piuttosto oneroso da un punto di vista computazionale e richiede la regolazione dell'iperparametro h. A tal fine è stata eseguita una partizione del dataset in 3 parti:

- Training set (50%): usato per l'estrazione dei campioni la cui distanza è inferiore ad h;
- Validation set (20%): usato per ottimizzare il parametro h, valutando su di esso l'MSE, e selezionando il valore di h per il quale il valore di errore è più piccolo;
- Test set (30%): per la valutazione delle performances finali in termini di corretta classificazione. In particolare, il valore h migliore è quello per cui l'MSE sulle stime dei dati appartenenti al Validation Set è minore.

Tale divisione è stata realizzata tramite la funzione *train_test_val_split* che, prendendo in input le percentuali di grandezza dei 3 set, li restituisce opportunamente divisi.

3.4.3 Implementazione realizzata

Dato che le librerie standard non mettono a disposizione dei metodi per l'esecuzione dell'algoritmo Naïve Kernel, è stato deciso di implementarlo manualmente. Questa scelta ha reso possibile anche l'utilizzo del framework Spark per la parallelizzazione del calcolo delle distanze. Tale algoritmo, come già detto, è molto simile al KNN, quindi è stato sufficiente modificare il metodo di aggregazione dei risultati, ovvero non viene più selezionato un numero fisso di campioni, che quindi devono essere ordinati in base alla distanza, ma vengono selezionati tutti

quelli vicini. Quindi è sufficiente, una volta calcolate tutte le distanze, calcolare la classe più frequente.
NaiveKernel_fit_and_predict: effettua l'addestramento e la predizione di una Y utilizzando Spark: per ogni X nel test set viene determinata la distanza Euclidea dai punti del training set, e vengono selezionati quelli che sono al di sotto di una certa soglia. Come prima operazione il dataset viene parallelizzato, ed il valore h viene mandato in broadcast a tutti i cluster Spark. Quindi attraverso l'utilizzo di *mapPartitions()* e di *collect()* di Spark, si ottengono i campioni per cui la distanza calcolata dalla X di test risulta essere inferiore ad h. Le stesse labels vengono parallelizzate per andare a determinare la frequenza di 1: se questi si presentano in più del 50% dei casi, la classe predetta è 1.
NaiveKernel: ottenute le predizioni dal metodo precedente, questa funzione si occupa di andare a determinare le performances sui dati di test, ma anche l'MSE.
Naive_kernel_plot: effettua la stampa della predizione usando il valore h passato. Viene ottenuta una meshgrid, partendo dai valori massimi e minimi di XX e YY passati.Vengono fatte delle predizioni sui valori di questa meshgrid, utilizzando il modello realizzato. Infine, i risultati sono graficati.
NaiveKernel_tuning: permette di ottimizzare il parametro h. Come prima cosa il dataset viene diviso in 3 parti, Training, Validation e Test set. Sulla base di valori minh, maxh ed nk passati, vengono determinati i valori di h da testare, su cui cercare quello ottimo. Per ognuno di questi valori di h, vengono determinate performances ed MSE sul Test set e sul Validation Set. Sulla base dell'MSE di Validation, viene determinato il valore di h ottimale, per il quale vengono anche selezionate le performances ed MSE sul Test set. Alla fine vengono plottate le regioni di decisione ottenute, e la relazione tra MSE di validation e valori di h.

3.4.4 Spark

La realizzazione del Naïve Kernel è stata effettuata direttamente tramite il framework Spark, rendendo possibile la parallelizzazione su più cluster nel caso in cui il software venga eseguito su più macchine.
Per ogni partizione del dataset distribuito viene effettuato un filtraggio che consiste nel selezionare tutti i campioni la cui distanza è inferiore alla soglia h, mandata in broadcast tra tutti i cluster. Per selezionare la classe, viene effettuata la stessa operazione del KNN, vale a dire che la somma delle labels delle classi viene confrontata con la metà del numero di campioni. Se tale somma è maggiore della metà del numero di vicini in un dato raggio, allora la classe assegnata è la classe 1; in caso contrario, viene assegnata la classe 0.

3.4.5 Naive Kernel su dataset sintetico con due features

Anche in questo caso viene mostrata la distribuzione dei dati al variare delle features *curtosis* e *variance*.
Le migliori performance sul Test set sono ottenute per h compreso tra 0.1325 e 0.2142.
Se, invece, si considera l'ottimizzazione del parametro h, si ha che il minimo MSE sul Validation set è ottenuto per h uguale a 0.2142, per cui le performances sul Test set sono comunque vicine all'ottimo (87%). Il fatto che le dimensioni del dataset non siano comunque troppo estese, fa si che il valore di soglia h non sia eccessivamente contenuto. Risulta che i risultati del modello Naïve Kernel su questa coppia di features sono in linea con quelli ottenuti finora.
É possibile apprezzare, di seguito, le regioni di decisione che vengono fuori dalla classificazione, e la relazione che intercorre tra le h e l'MSE di Validation.

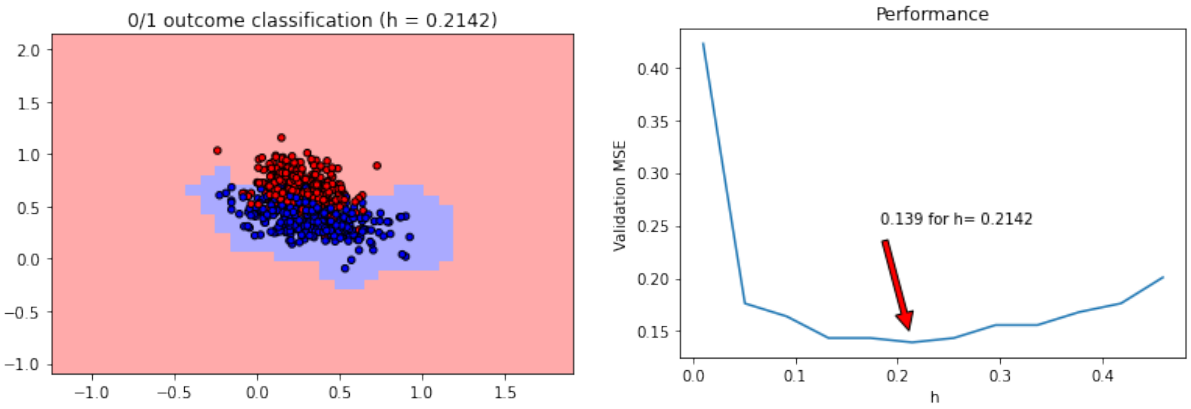


Figura 3.20: Test performance: 87.16% and Test MSE: 0.1284 for h = 0.2142.

3.4.6 Naive Kernel su dataset reale con due features

Le migliori performances sul Test set si ottengono per h pari a 0.0917. Al contrario di quanto accaduto prima, si evince, invece, che il più piccolo valore di MSE sul Validation set (0.123), si ha con un valore di h più piccolo, in particolare con h pari a 0.0508. Ciò è ipotizzabile dall'analisi della classificazione ottenuta e dalla distribuzione dei campioni: la densità dei punti è elevata e, quindi, è ragionevole un valore di h ristretto. I risultati sono di poco superiori rispetto a quelli riscontrati utilizzando il dataset sintetico.

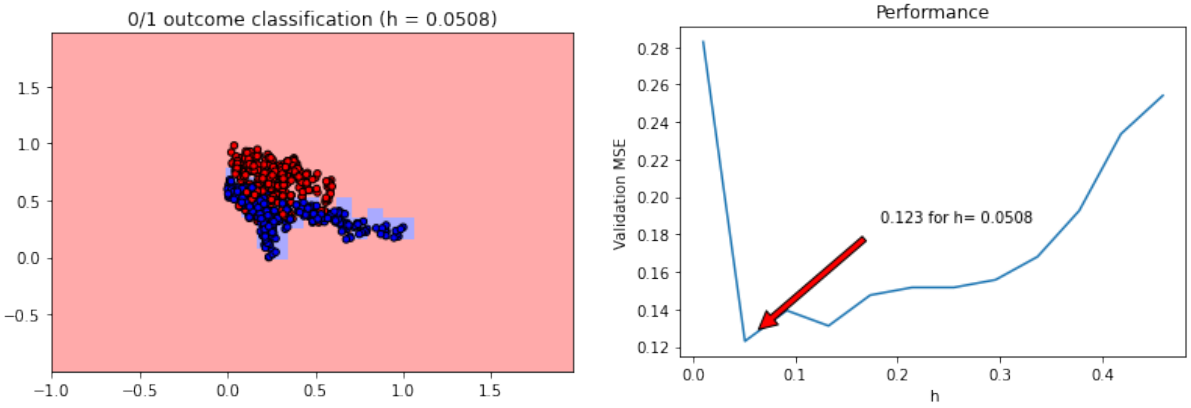


Figura 3.21: Test performance: 90.71% and Test MSE: 0.0929 for h = 0.0508.

3.4.7 Naive Kernel su dataset reale usando una feature per volta

É riportata la tabella contenente per ogni feature i valori di h che garantiscono i valori più bassi dell' MSE sul Validation set, ed in corrispondenza di questi, le performances e l'MSE sul Test set.
I risultati sono sempre in linea con quanto osservato finora, in particolare vediamo che la variabile entropy offre le peggiori prestazioni.

	variance	skewness	curtosis	entropy
h	0.0917000025510788	0.05079999938607216	0.009999999776482582	0.009999999776482582
Validation MSE	14.34%	24.59%	34.43%	47.13%
Test Score	83.06%	78.42%	60.93%	50.00%
Test MSE	16.94%	21.58%	39.07%	50.00%

3.4.8 Naive Kernel su dataset reale usando tutte le features disponibili

Vengono prese in esame tutte le features del dataset. Ancora una volta, non risulta possibile visualizzare i risultati della classificazione, per via della multidimensionalità del problema.
Evidentemente il numero di features è aumentato, e ciò comporta un numero di istanze richieste maggiore: in effetti il numero di campioni risulta effettivamente ristretto, tenendo in considerazione che questo deve crescere con il numero di features. Analizzando i valori di performance di Test, si nota che il valore di h per cui si ottengono risultati migliori è h pari a 0.1325 o 0.1733. Il valore di h per cui si ottiene il valore di Validation MSE più basso (0.0123) è sempre per h pari a 0.1325 o 0.1733, con Test performance del 99.45%.
In linea con tutti i metodi di classificazione visti finora si può vedere come, coinvolgendo nella classificazione tutte le features presenti nel dataset si ottiene un miglioramento in termini di performances e calcolo dell'errore, infatti si ottengono i seguenti risultati:

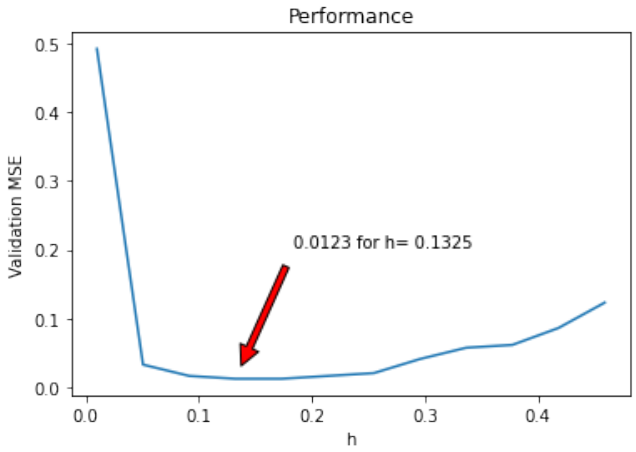


Figura 3.22: Test performance: 99.45% and Test MSE: 0.0055 for h = 0.1325.

Si fa notare che, da un certo punto in poi, l'MSE tende ad aumentare quando h si avvicina ad 1, poiché in questo caso si tende a considerare più vicini del necessario, esattamente come accadeva nel caso del KNN all'aumentare di K.

3.4.9 Naive Kernel su dataset sintetico usando tutte le features

Per motivi di tempistica, vengono analizzati solamente le casistiche in cui il numero di campioni nel dataset sintetico risulta essere il doppio ed il triplo rispetto alle dimensioni del dataset reale. In questo modo si può anche apprezzare come varia la scelta di h, al variare del numero di campioni.

Numero di campioni N*2

Le migliori performances sul Test set (97.13%) si ottengono per h tra 0.1325 e 0.1733. Per quanto riguarda l'ottimizzazione di h, il valore di MSE di validation (0.0164) più piccolo si ha per h di circa 0.1733. Quindi in questo caso, il valore di h che garantisce il più piccolo MSE sul Validation set, è circa lo stesso che garantisce anche le migliori performance sul Test set.

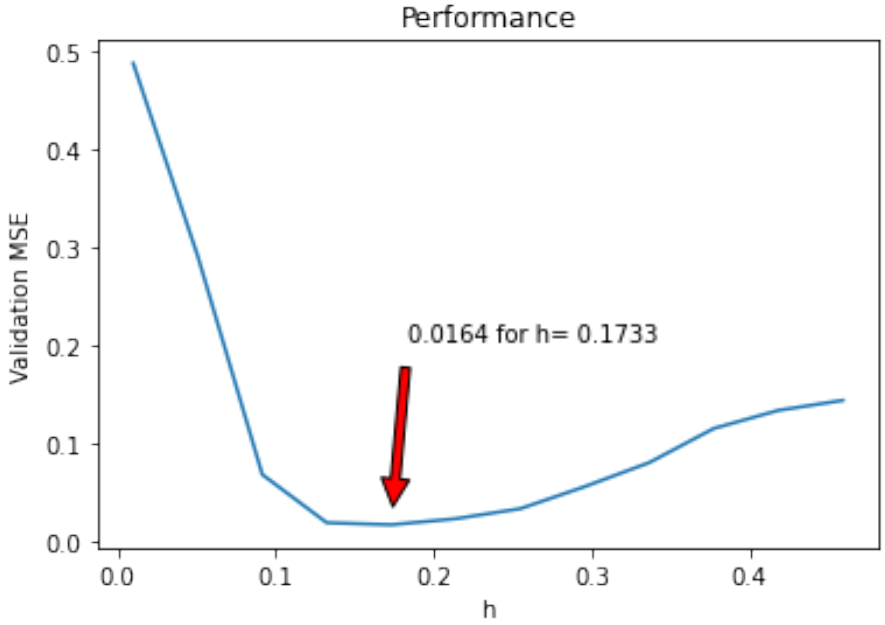


Figura 3.23: Test performance: 97.13% and Test MSE: 0.0287 for h = 0.1733.

Mettendo in relazione i risultati ottenuti con quelli sul dataset reale, con un numero di campioni dimezzato, risultano essere peggiorati. Come già visualizzato nel caso delle due features selezionate e del dataset sintetico realizzato in tale situazione, i campioni per le classi generati casualmente a partire da una distribuzione normale multivariata, possono risultare molto ravvicinati, andando a confondere le classi. Questo può aver indotto a delle performances peggiori.

Numero di campioni N*3

Adesso vengono considerati il triplo dei campioni rispetto alle dimensioni del dataset reale. Il tutto è benefico: come è lecito aspettarsi, con un numero di campioni maggiore, nel complesso le performances, per ogni valore di h, migliorano. Si può notare che i valori più piccoli di MSE sul Validation set (0.006831) si ottengo ancora una volta per h uguale a 0.1733, con performance sul Test set del 97.63%.
In definitiva, i risultati attesi da tale metodo sono in linea con quanto ottenuto finora con gli altri metodi, eccetto per piccole variazioni.

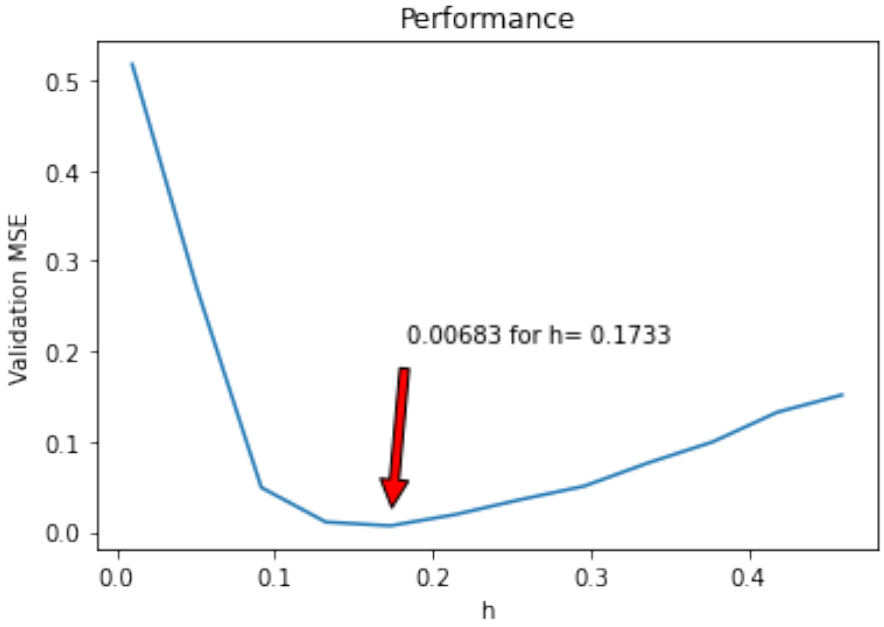


Figura 3.24: Test performance: 97.63% and Test MSE: 0.0237 for h = 0.1733.

3.5 Regressione Logistica

La regressione logistica rappresenta un metodo di classificazione rientrante nella famiglia degli algoritmi di apprendimento supervisionato.
Essa si avvale della funzione sigmoide $\sigma(x) = \frac{1}{1 - e^{-x}}$. Questa può prendere qualsiasi numero di valore reale $(-\infty, +\infty)$ e mapparlo in un valore compreso tra]0,1[. Si può modellare la probabilità che un valore di input X appartenga ad una tale classe Y (0 o 1).

3.5.1 Funzione di costo e gradiente discendente

Come accade per la regressione lineare, abbiamo un vettore dei parametri che deve essere ottimizzato attraverso una funzione di costo. Nel caso in esame prendiamo in considerazione la Cross Entropy, ed in particolare la sua versione approssimata con la media aritmetica.
Per l'apprendimento si applica l'algoritmo del gradiente discendente, che consiste nell'andare ad aggiornare ciclicamente il vettore dei pesi, tenendo in considerazione il learning rate o step size, il cui scopo è quello di andare a determinare la velocità di convergenza dell'algoritmo.

3.5.2 Divisione del dataset

É stata eseguita una partizione del dataset in 2 parti:

- Training set (70%): usato per l'aggiornamento dei pesi w;
- Test set (30%): per la valutazione delle performances finali in termini di corretta classificazione.

3.5.3 Implementazione realizzata

Per implementare il classificatore binario Logistic Regression tramite Spark, è stato necessario definire:

- La funzione **sigmoid()**;
- La funzione **LogReg_gradient()**, per calcolare il gradiente della funzione di costo rispetto ai parametri ed ai dati del Training set;
- La funzione di costo **compute_cost()**, che da in output il valore di costo calcolato rispetto al dataset X, il peso corrente w e le labels attuali;
- L'algoritmo di discesa del gradiente **gradient_descent_spark()** rispetto agli iperparametri:
  - Learning Rate (o step size);
  - Momentum: ulteriore iperparametro utile per velocizzare l'apprendimento.

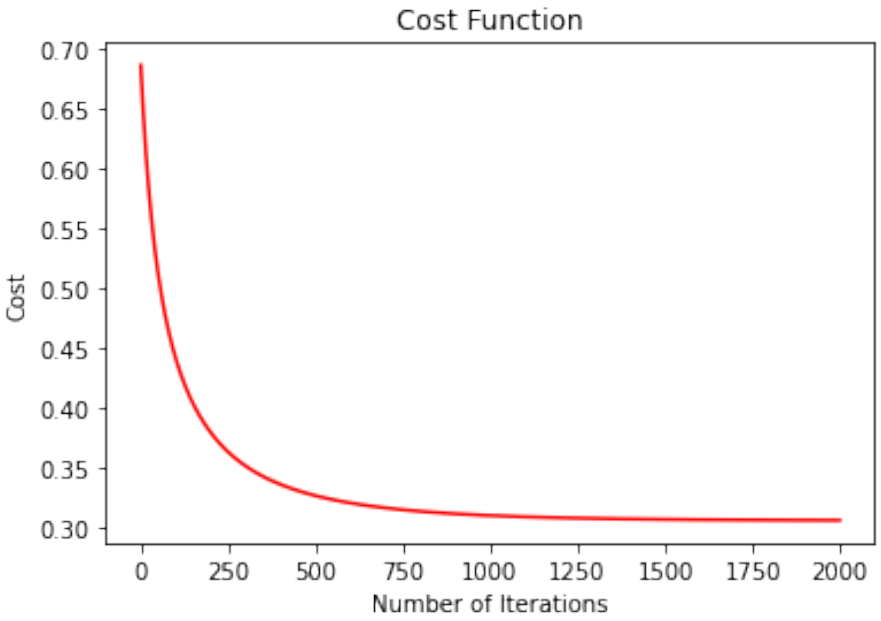
  In questa funzione, viene realizzata la parallelizzazione del dataset, passato come parametro, e viene applicato il calcolo del gradiente sul Training set un numero di volte specificato dal parametro iterations, utilizzando le funzioni Spark *map()* e *reduce()*. La prima consente di calcolare il gradiente su ogni partizione, indicando l'apposita funzione, mentre la seconda di aggregare, per mezzo somma, i risultati ottenuti.
- La funzione di classificazione **score_spark()** rispetto ai parametri calcolati tramite l'algoritmo di discesa del gradiente ed ai dati del Test set;

- La funzione **logistic_regression_spark()** che calcola la regressione logistica utilizzando gli algoritmi precedentemente descritti. Effettua il calcolo dei costi, e l'aggiornamento dei pesi per un numero di volte specificato dal parametro *iterations*. Successivamente effettua la predizione in funzione dei pesi trovati e delle x di test. Inoltre effettua il plot delle funzione di costo, in relazione con il numero di iterazioni. Infine stampa le performances.

- La funzione **logistic_regression()** che calcola la regressione logistica utilizzando le funzioni di libreria. In particolare viene utilizzato l'algoritmo SGD: quindi l'ottimizzazione dei pesi viene fatta usando un singolo campione per volta. Si effettua l'addestramento e la predizione. Infine sono graficate le regioni di decisione e le performances.

3.5.4 Regressione Logistica su dataset sintetico con due features

É possibile analizzare il grafo che mette in relazione la funzione di costo con il numero di iterazioni. Come si evince, al crescere delle iterazioni, il costo tende, progressivamente, a decrescere.



Si possono confrontare le performances nei due casi, quello con l'utilizzo di Spark, e quello senza. Queste risultano essere molto vicine. Quindi nel complesso l'algoritmo implementato sembra essere soddisfacente.
C'è da considerare, inoltre, il fatto che i parametri di *iterations, learning rate e momentum* possono essere comunque regolati per ottenere delle diverse performance. Nel caso in esame sono stati settati: iterations = 2000; learning_rate = 0.9; momentum = 0.001.

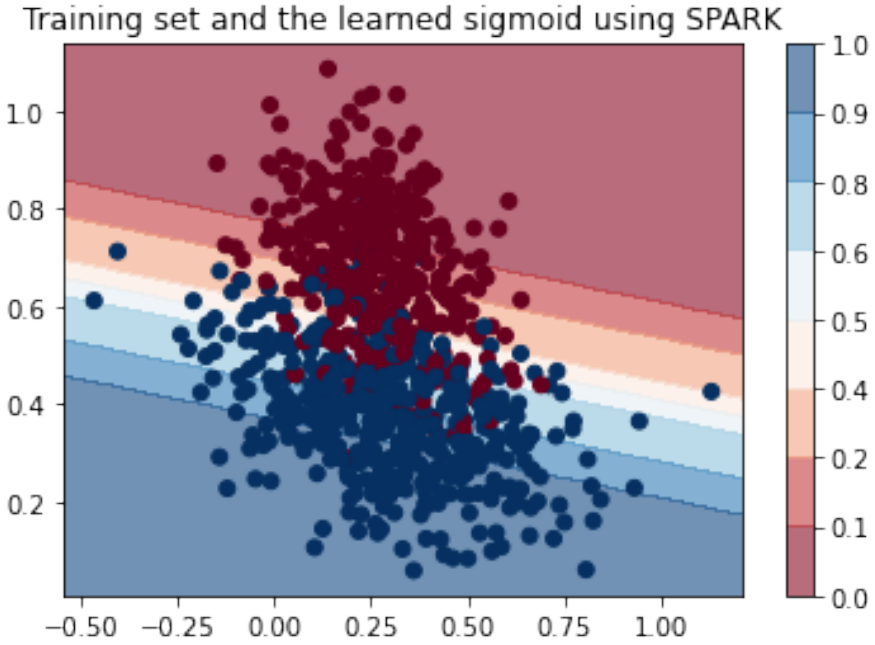


Figura 3.25: Performance con SPARK: 86.07%.

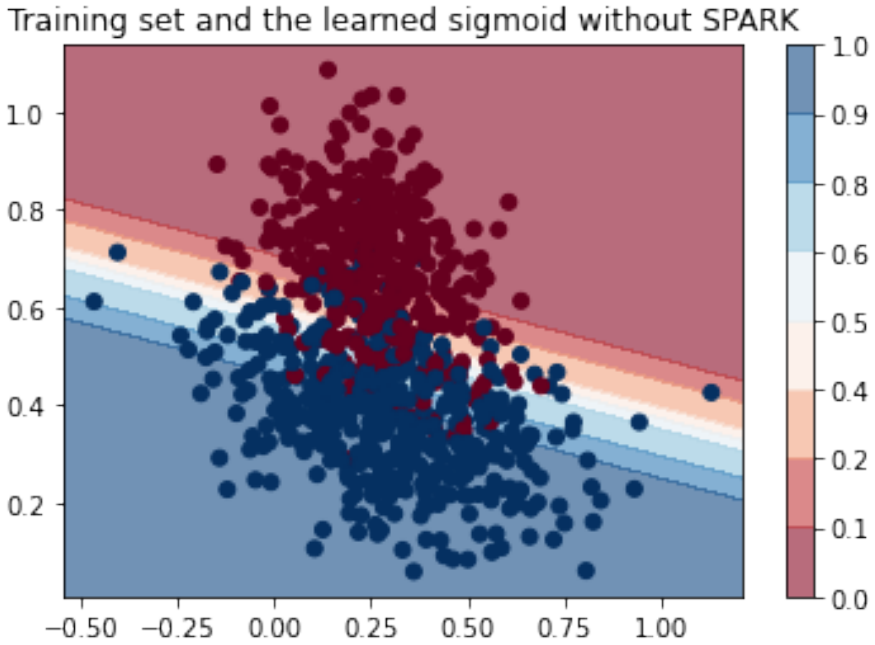
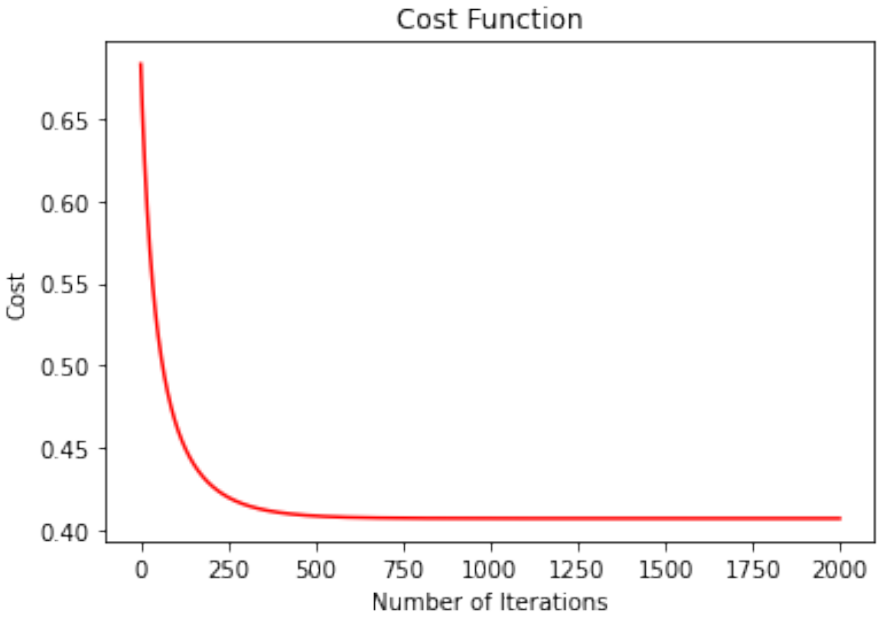


Figura 3.26: Performance senza SPARK: 85.79%.

3.5.5 Regressione Logistica su dataset reale con due features

In questa circostanza sono stati settati a: 1)iterations = 2000; 2)learning_rate = 0.9; 3) momentum = 0.005.
Ancora una volta viene graficata la funzione di costo in relazione al numero di iterazioni settate.



In merito alle performances osserviamo che i risultati con l'utilizzo di Spark sono leggermente inferiori rispetto a quelli riscontrati utilizzando il dataset sintetico. Non utilizzando Spark, invece, risulta vero il contrario.

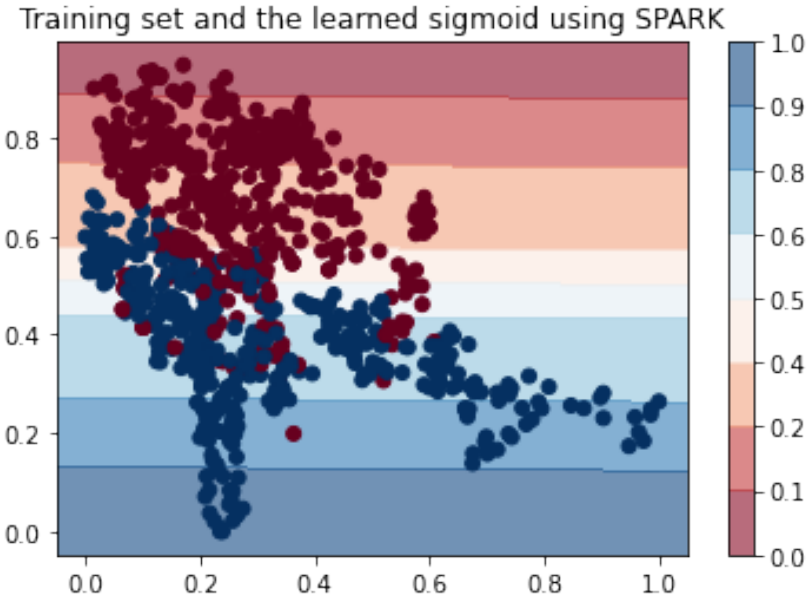


Figura 3.27: Performance con SPARK: 81.69%.

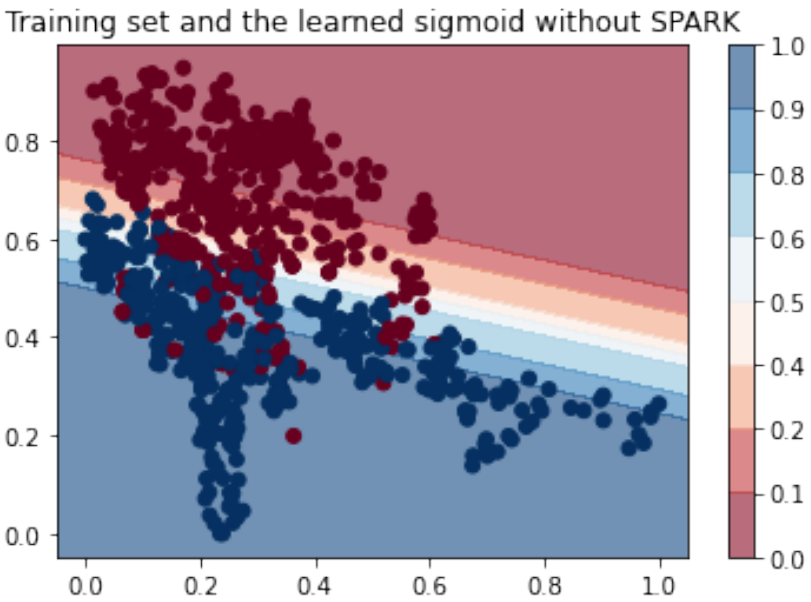


Figura 3.28: Performance senza SPARK: 88.25%.

3.5.6 Regressione Logistica su dataset reale usando una feature per volta

Nel seguente caso è stata presa in considerazione la sola implementazione con funzioni di libreria. Vengono riportate le performances nel momento in cui vengono effettuati apprendimento e predizioni utilizzando una feature per volta, al fine di confermare l'impatto di ogni feature sulla percentuale di corretta classificazione. Queste sono riportate nella seguente tabella:

variance	skewness	curtosis	entropy
83.88%	70.77%	51.09%	49.45%

Come si evince dalla tabella, nella maggior parte dei casi le performances si attestano tra 49% e 70%, ad eccezion fatta per la feature *variance*, per cui le performances superano l'83%. Ciò evidenzia che tale feature, risulta particolarmente utile nella predizione su *class*.

3.5.7 Regressione Logistica su dataset reale usando tutte le possibili coppie di features

Sono riportate le performances per ogni possibile coppia dai regressori:

variance + skewness	variance + curtosis	variance + entropy	skewness + curtosis	skewness + entropy	curtosis + entropy
84.43%	86.34%	84.15%	78.42%	73.22%	55.19%

Come si nota ancora una volta, la presenta della feature *variance* in una coppia, comporta un aumento delle performances, fino a toccare un picco dell'86% quando è in coppia con *curtosis*. Ciò dimostra che questa feature è particolarmente adatta alla predizione su *class*.

3.5.8 Regressione Logistica su dataset reale usando tutte le features disponibili

Anche per le seguenti analisi sono stati scelti i seguenti valori: 1)iterations = 2000; 2)learning_rate = 0.1; momentum = 0.005. Le regioni in output non possono essere graficate per via della multidimensionalità del problema; per questo motivo, saranno riportate le sole performances ottenute con le due metodologie.

```
Performance using SPARK:  89.89%


Confusion matrix:
[[165   7]
 [  2 192]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       0.99      0.96      0.97       172
         1.0       0.96      0.99      0.98       194

    accuracy                           0.98       366
   macro avg       0.98      0.97      0.98       366
weighted avg       0.98      0.98      0.98       366


Precision Score:
0.964824120603015



==========================================================
Performance without SPARK:  97.54%
```

Ovviamente, le performances ottenute con Spark risentono del fatto che i parametri di learning rate e momentum possono essere non ottimali.

3.5.9 Regressione Logistica su dataset sintetico usando tutte le features

In questo caso è stato impiegato il dataset sintetico, realizzato a partire dalla stima di media, varianza e matrici di covarianza per tutte le features. Inoltre il numero di campioni è 12 volte maggiore rispetto al dataset reale.

```
Performance using SPARK:  98.75%


Confusion matrix:
[[2177   16]
 [   9 2190]]


Classifier metrics:
              precision    recall  f1-score   support

         0.0       1.00      0.99      0.99      2193
         1.0       0.99      1.00      0.99      2199

    accuracy                           0.99      4392
   macro avg       0.99      0.99      0.99      4392
weighted avg       0.99      0.99      0.99      4392


Precision Score:
0.9927470534904805



==========================================================
Performance without SPARK:  99.43%
```