



Progetto di Statistical Data Analysis

Barbella Michele
Matr. 0622701341

A.A. 2019 / 2020



Progetto in R: indice

- Preprocessing
- Regressione multipla
- Ricampionamento statistico
- Subset Selection
- Regressione con Regolarizzazione
- Riduzione della dimensionalità

Progetto in Python: indice

- Analisi dei dati
- Naïve Bayes
- K-nearest neighbors
- Naïve Kernel
- Regressione Logistica

Dataset regressione

Dataset che raccoglie dati relativi ai film, composto da 506 righe e 18 colonne:

- **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.
- **Production expense:** le spese per produrre il film in dollari.
- **Multiplex coverage:** la copertura dei multiplex (edificio che proietta più film contemporaneamente)
- **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 hashtags:** in numero di volte che è stato utilizzato l'hashtag 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.

Preprocessing

Il preprocessing è stato effettuato in ambiente RStudio, utilizzando il linguaggio R:

- Verifica della presenza di elementi duplicati.
- Rimozione attributi non decisivi al fine dell'analisi, quali:
 - Multiplex coverage;
 - Movie_length;
 - 3D_available;
 - Genre;
 - Num_multiplex;
 - Collection.
- Verifica presenza valori mancanti: sono risultati 12 elementi mancanti relativi all'attributo Time_taken, pertanto si è provveduto a rimuovere le righe relative a questi valori.

A questo punto il dataset su cui andremo a fare le operazioni è composto da 494 istanze.

Progetto in R

Esecuzione del codice effettuata su RStudio

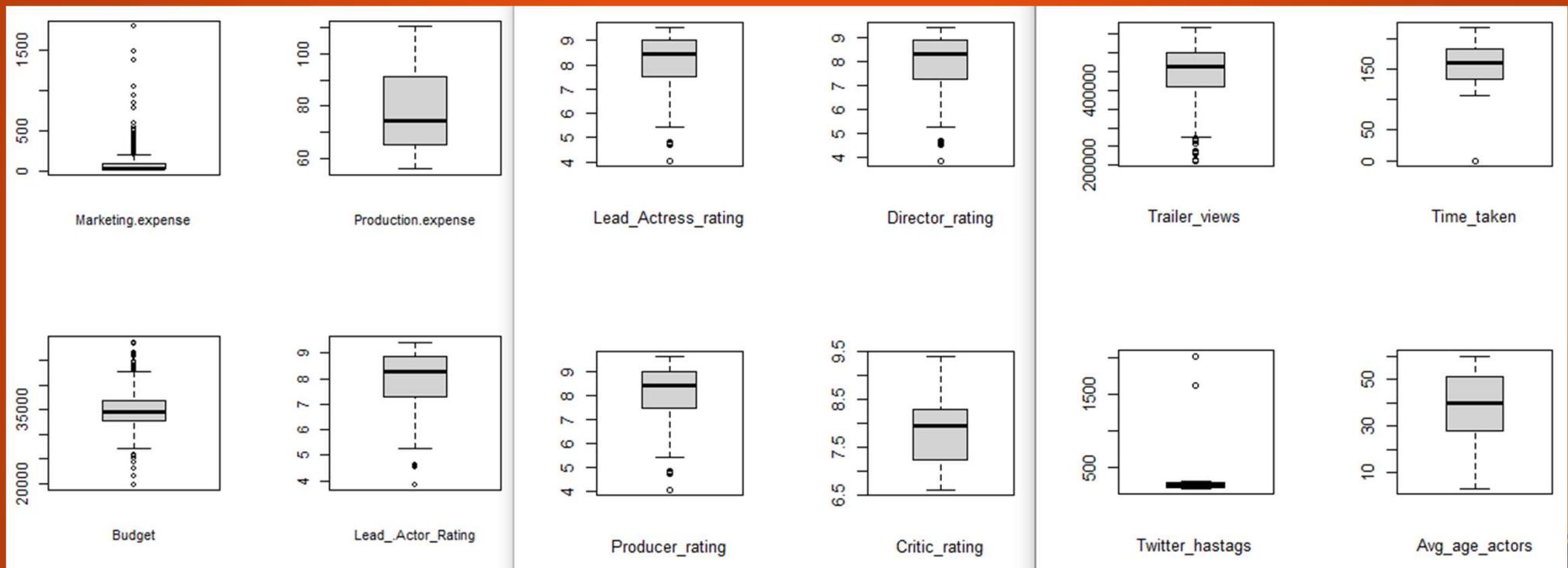
- Feature da predire: **Budget**, variabile continua.
- Features usate per la predizione:
 - Marketing expense,
 - Production expense,
 - Lead Actor Rating,
 - Lead Actress rating,
 - Director rating,
 - Producer rating,
 - Critic rating,
 - Trailer views,
 - Time taken,
 - Twitter hastags,
 - Avg age actors.



Identificazione e rimozione Outliers

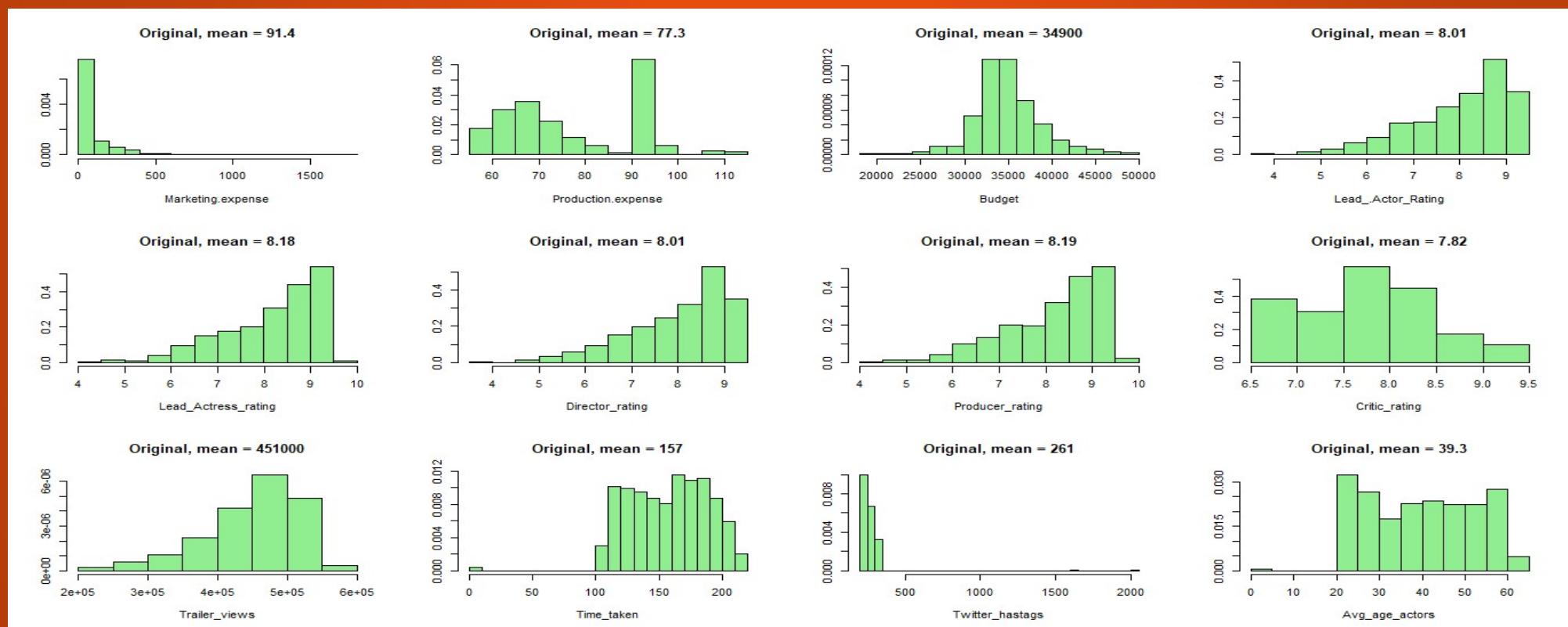
Gli outliers sono delle osservazioni anomale, ovvero che differiscono significativamente dalle altre. L'identificazione è stata effettuata analizzando diversi boxplot.

Per ogni feature del dataset è stato costruito un boxplot e si considerano outliers tutte le osservazioni che giacciono fuori da $1.5 * \text{IQR}$, dove IQR è il range interquartile, ovvero la differenza tra il primo e il terzo quartile.



Analisi degli histogrammi

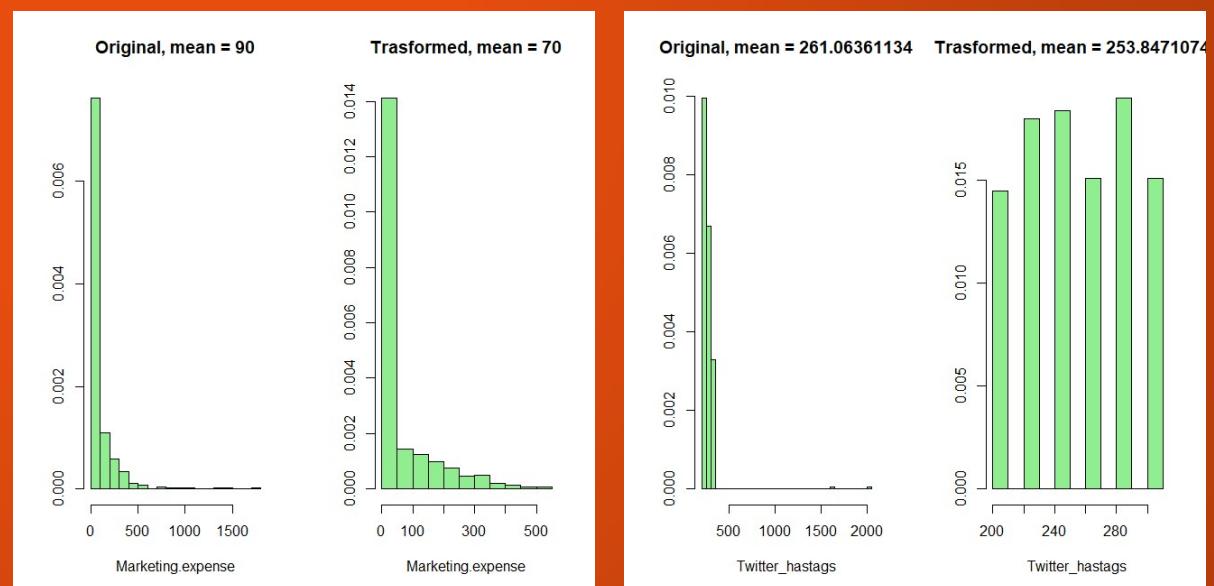
Per capire se rimuovere o meno gli outliers, sono stati usati, oltre ai boxplot, anche gli histogrammi dei grafici costruiti raggruppando i dati in intervalli di classi e poi piazzando il numero di dati che cadono in ogni intervallo. Questi possono essere usati per capire se una variabile è distribuita come una normale.



Analisi degli histogrammi

Dall'analisi si nota che le distribuzioni degli attributi non sono normali ma risultano sbilanciate. Questo può avere un impatto negativo sulla regressione lineare; 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.

In seguito alla rimozione di alcuni outliers le istanze del nostro dataset si sono ridotte a 484 elementi. Nell'immagine in dettaglio il risultato dell'eliminazione degli outliers per gli attributi di Marketing expense e Twitter hashtags. In particolare a destra i dati originali e a sinistra i dati dopo la trasformazione.

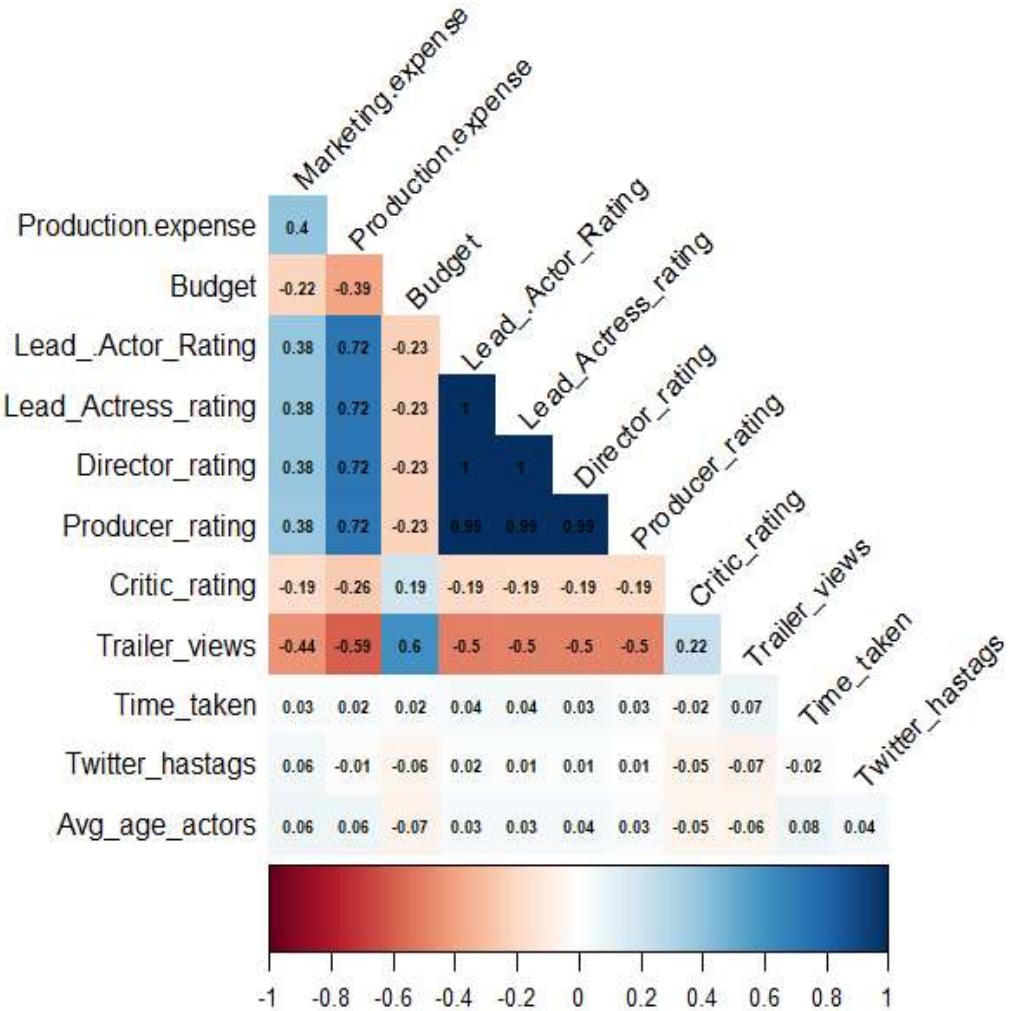


Matrice di Correlazione

L'indice di correlazione esprime un'eventuale relazione di linearità tra le variabili. Può essere usato per capire quanto sia lineare la relazione tra risposta e predittore ma anche per valutare se le variabili sono dipendenti tra di loro.

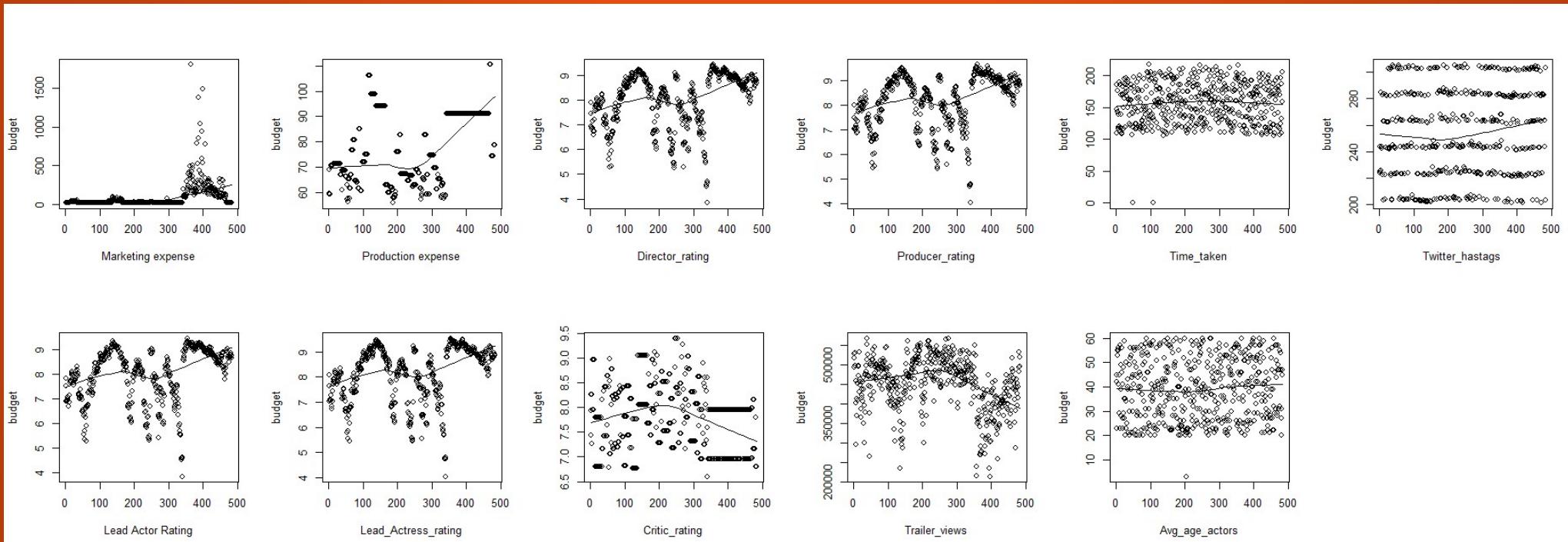
È un valore compreso tra -1 e 1:

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



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. A tal proposito sono stati costruiti degli scatterplot tra la risposta e ogni singolo predittore, tramite la funzione `scatter.smooth`.



Linearità dei dati

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.



Regressione lineare

Lo scopo della regressione lineare è modellare una variabile dipendente Y come funzione matematica di una o più variabili X, in modo da poter utilizzare questo modello di regressione per prevedere Y quando si conosce solo X, tramite una relazione lineare (una formula matematica) tra le variabili predittive e la variabile di risposta, modellabile come segue:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

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).



Regressione lineare

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					

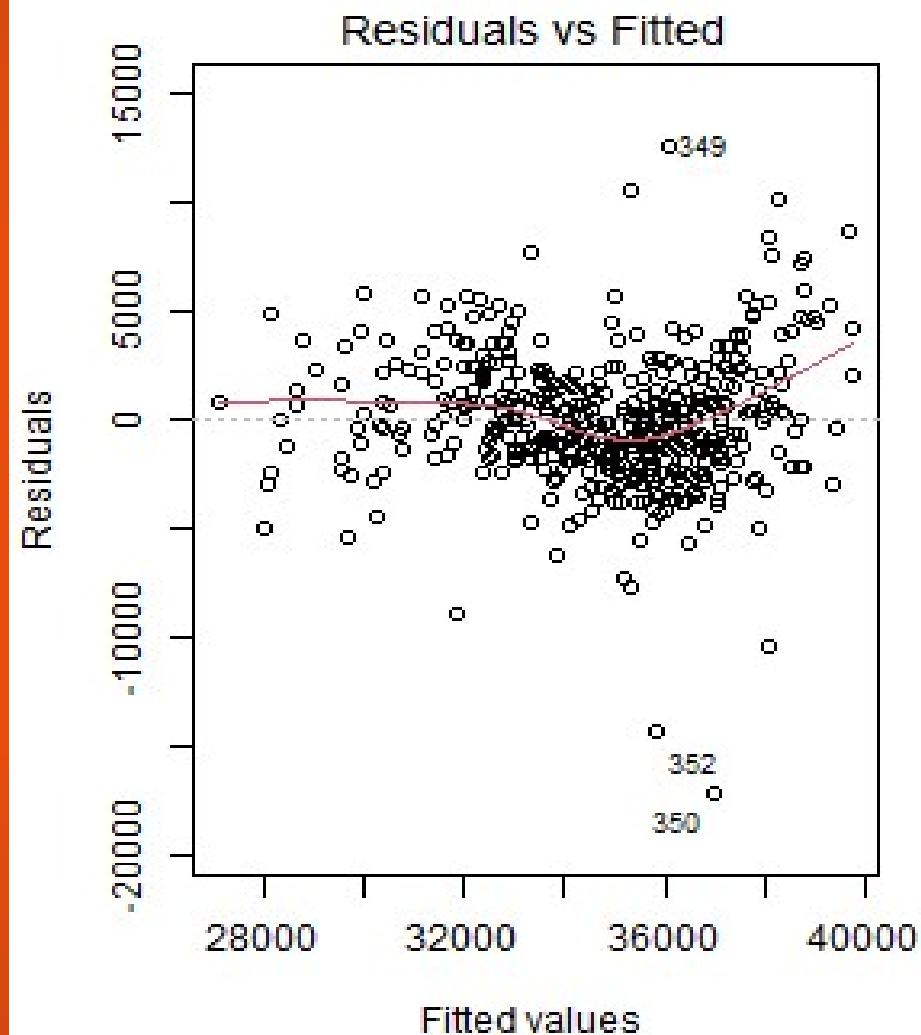


Scelta del modello migliore

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

Residuals vs Fitted:

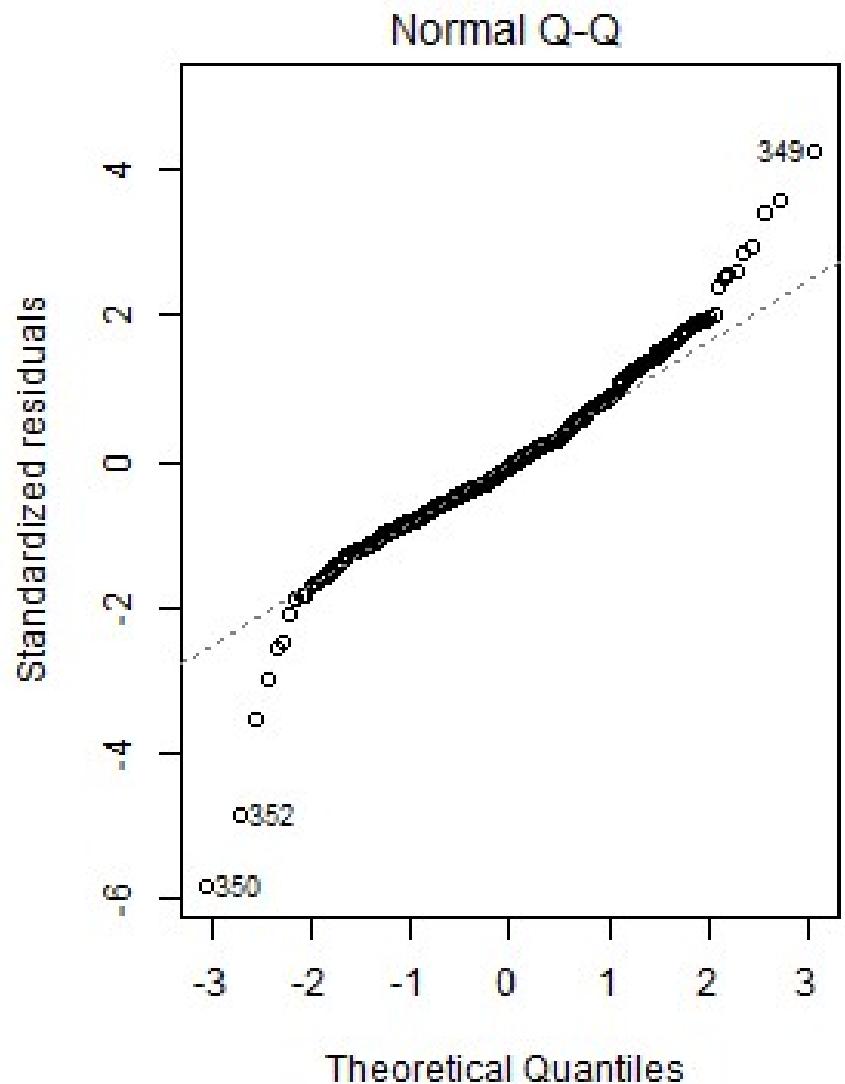
mostra gli errori residui contro i loro valori stimati. I residui devono essere distribuiti in modo casuale attorno alla linea rossa orizzontale che rappresenta un errore residuo pari a zero, cioè non dovrebbe esserci una netta tendenza nella distribuzione dei punti. Non si evidenzia una tendenza particolare 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.



Scelta del modello migliore

Normal Q-Q:

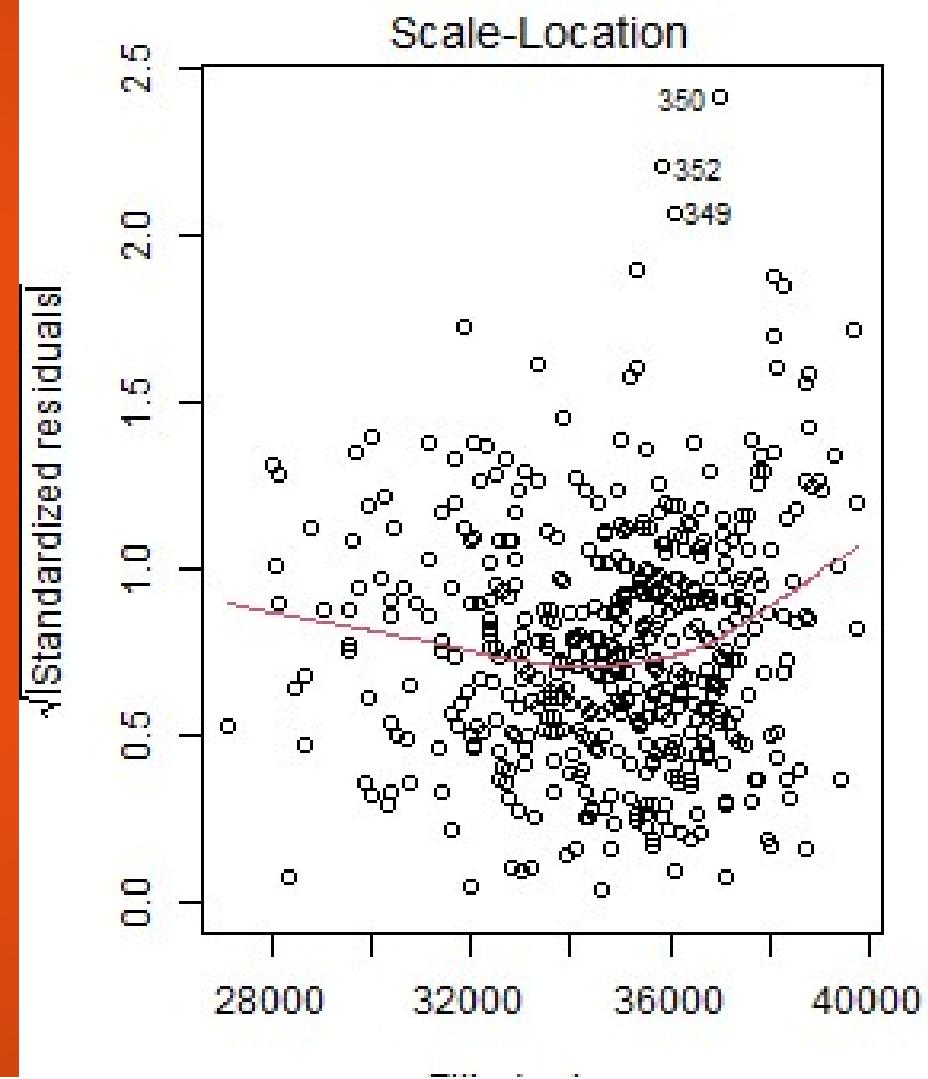
confronta i valori dei residui standardizzati (quantile reale) verso la linea che individua la loro distribuzione normale (quantile teorico), ovvero il grafico rappresenta una figura per cui se i punti si distribuiscono sulla linea, la distribuzione dei residui risulta normale e quindi la regressione 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.



Scelta del modello migliore

Scale-Location:

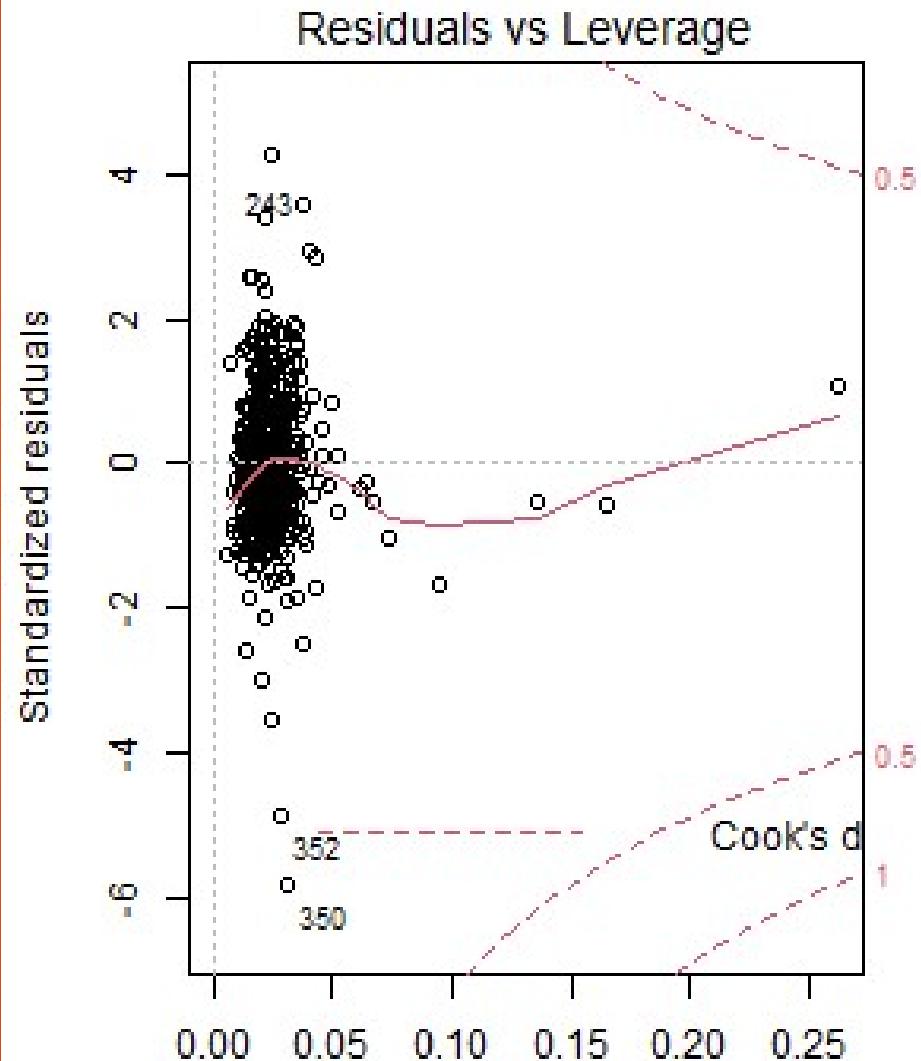
mostra la radice quadrata dei residui standardizzati in funzione dei valori stimati. E' un grafico molto utile per individuare gli outliers che sono rappresentati da residui non distribuiti equamente sopra e sotto la linea orizzontale. Anche in questo grafico notiamo che le osservazioni 349, 352 e 350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



Scelta del modello migliore

Residuals vs Leverage:

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. Sovraposte al plot ci sono linee di contorno per la distanza di Cook. Valori di distanze di Cook bassi per un punto indicano che la rimozione dell'osservazione ha poco effetto sui risultati della regressione. Invece valori di distanze di Cook superiori a 1 sono sospetti ed indicano la presenza di un possibile outlier o di un modello povero.



Regressione polinomiale

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. 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.

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 Inflation 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.

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_hashtags	Avg_age_actors	
	1.094682	1.738411	1.031922	1.020655	1.025137	



Scelta del modello migliore

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.

Dalle analisi fatte, il modello di regressione polinomiale migliore è risultato quello di ordine 4 dei regressori.

Tramite la funzione `summary()`, è possibile analizzare anche i coefficienti ottenuti, individuando i più significativi.

Un maggior numero di '*' segnala un coefficiente rilevante per il modello. 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.

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_Actoress_rating, 4)1	-13641.40	61596.59	-0.221	0.824834
poly(Lead_Actoress_rating, 4)2	35379.03	45119.25	0.784	0.433391
poly(Lead_Actoress_rating, 4)3	2379.91	39070.98	0.061	0.951457
poly(Lead_Actoress_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_hashtags, 4)1	-590.98	2896.59	-0.204	0.838429
poly(Twitter_hashtags, 4)2	-5967.50	2915.72	-2.047	0.041286 *
poly(Twitter_hashtags, 4)3	-384.44	2963.71	-0.130	0.896852
poly(Twitter_hashtags, 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

Ricampionamento statistico

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).

- Validation Set
- K-fold Cross Validation
- Bootstrap



Validation Set

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



K-Fold Cross Validation

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.

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



Bootstrap

E' 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.

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



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;
- Backward Stepwise Selection.

Tutti questi metodi semplificano il modello al fine di migliorare l'accuratezza dei dati.



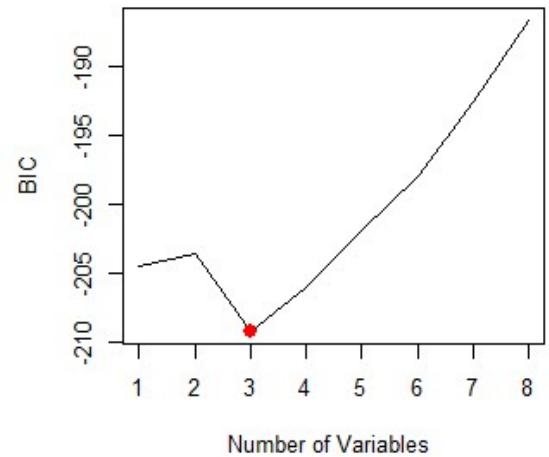
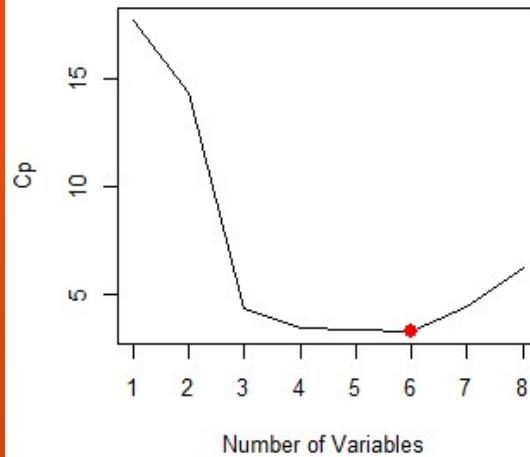
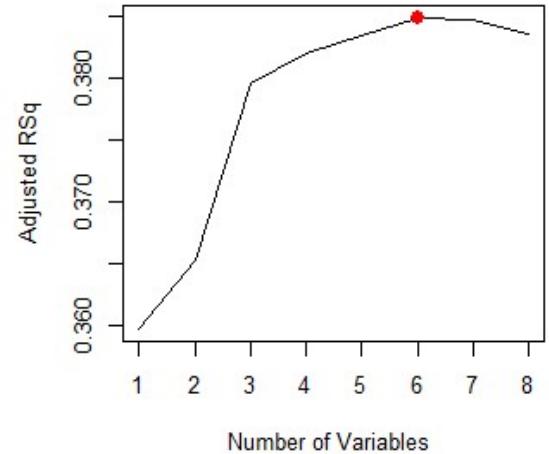
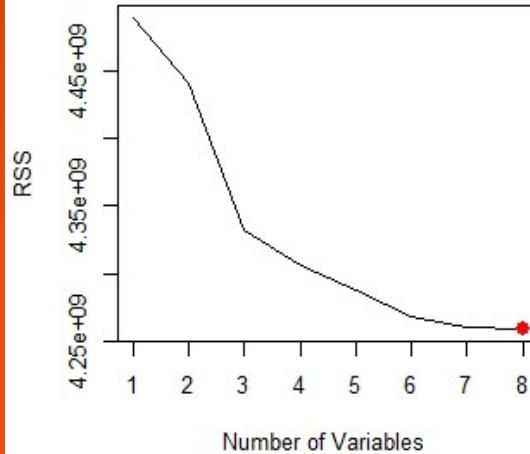
Best Subset Selection

Eseguiamo una regressione lineare per ogni possibile combinazione dei predittori. 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.

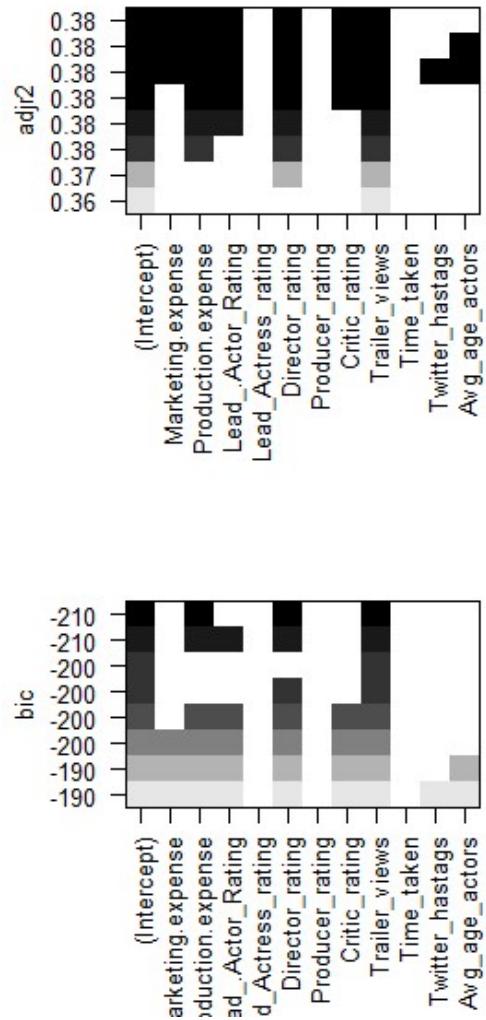
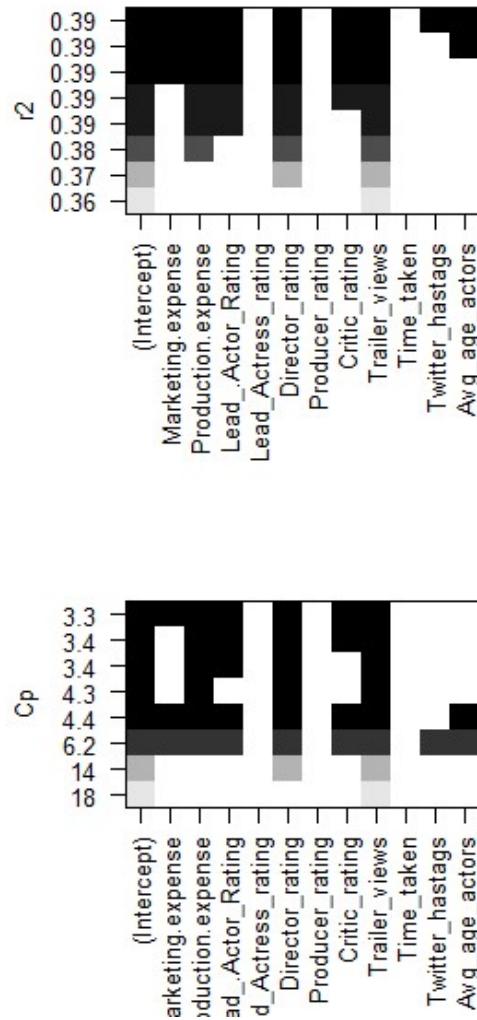
Tutti questi metodi semplificano il modello al fine di migliorare l'accuratezza dei dati.



Best Subset Selection

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.

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.



Best Subset Selection: Confronto dei risultati

Numero di predittori utilizzati al variare dei metodi analizzati per ogni trasformazione non lineare.

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



Stepwise Selection

E' un metodo per adattare i modelli di regressione in cui la scelta delle variabili predittive viene effettuata mediante una procedura automatica.

Forward Selection:

Inizia con una sola variabile nel modello, testa l'aggiunta di ogni variabile. Se l'inclusione di quest'ultima fornisce un miglioramento significativo dell'adattamento, la variabile viene aggiunta e il processo viene iterato. Vantaggio computazionale rispetto alla subset selection perché addestra molti meno modelli, ma non garantisce di trovare il miglior modello possibile.

Backward Selection:

Inizia con tutte le variabili del modello, testa la cancellazione di ogni variabile utilizzando un criterio, ed eventualmente la elimina. Se la perdita di quest'ultima determina un deterioramento significante del modello, la variabile viene eliminata e il processo viene iterato.



Stepwise Selection: considerazioni finali

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



Regressione con Regolarizzazione

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 aumentare l'accuratezza stessa. Le due tecniche di regolarizzazione affrontate in questo progetto sono: Ridge e LASSO.

La funzione `glmnet()` può essere utilizzata per adattare sia la Ridge che la LASSO regression. Con `alpha=0` si effettua la Ridge regression, con `alpha=1` si effettua la LASSO.



Regressione 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.

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

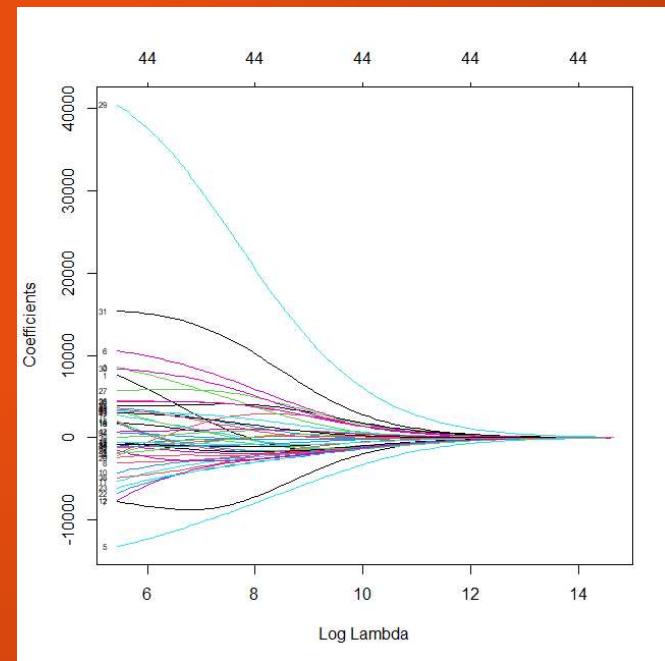
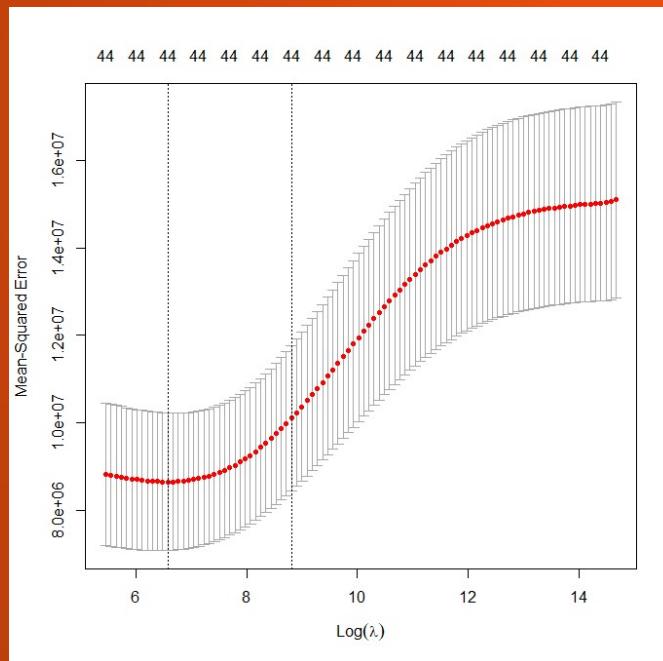


Regressione Ridge

Per mostrare a titolo qualitativo i risultati ottenuti, l'analisi seguente verrà commentata solo sul caso della trasformazione non lineare di quarto grado.

Nei grafici seguenti si può notare:

- nel primo, l'andamento del MSE in funzione del parametro di tuning;
- nel secondo, il diminuire dei coefficienti in funzione di lambda.



LASSO

LASSO riduce i coefficienti di regressione verso lo zero, utilizzando un termine di penalità chiamato norma L1, che è la somma dei coefficienti assoluti.

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.

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

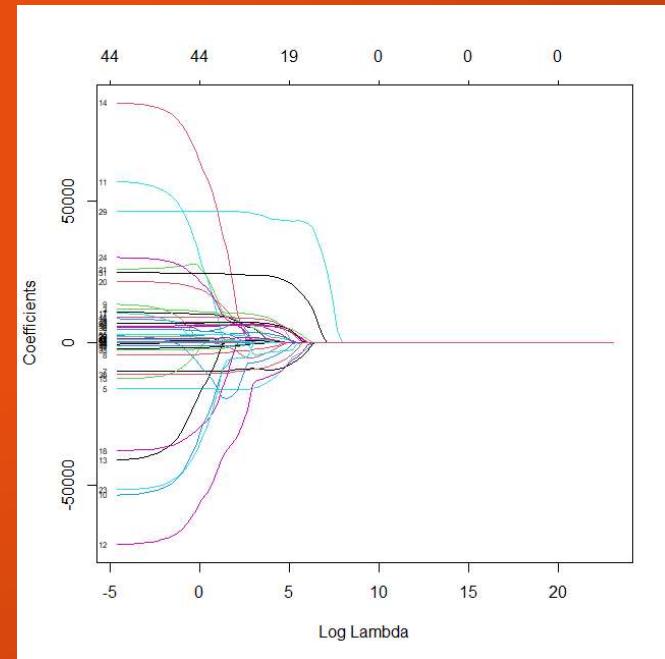
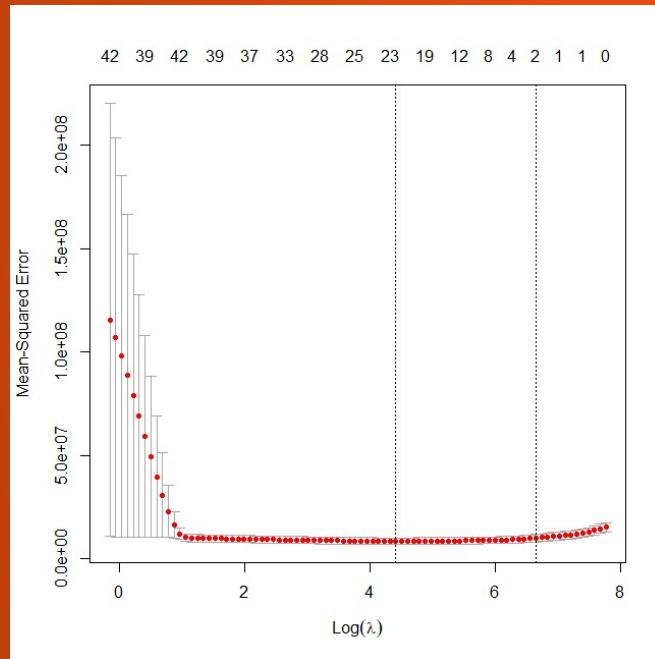


LASSO

Per mostrare a titolo qualitativo i risultati ottenuti, l'analisi seguente verrà commentata solo sul caso della trasformazione non lineare di quarto grado.

Nei grafici seguenti si può notare:

- nel primo, il valore dell'MSE al crescere del fattore λ .
- nel secondo, le curve che disegnano i coefficienti in funzione della misura di λ .



Regressione con Regolarizzazione: conclusioni

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

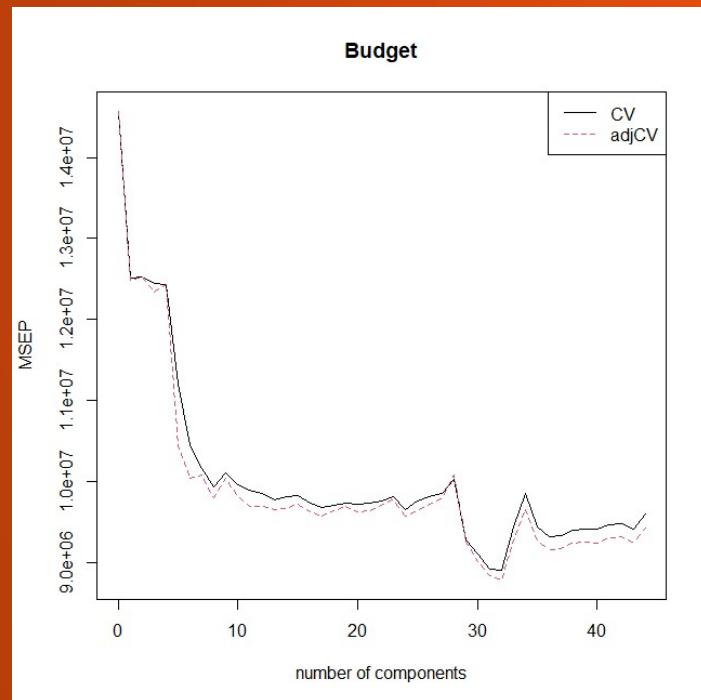
Lasso produce risultati leggermente più alti rispetto a Ridge. Inoltre produce modelli molto competitivi e comparabili con quelli senza regolarizzazione, con il sostanziale beneficio di ridurre i coefficienti a zero e pertanto produrre modelli molto più interpretabili.
Infine, nel caso polinomiale di 4° ordine si può notare un grande miglioramento.



PCR (Principal Component Regression)

Tecnica di regressione basata sull'analisi delle componenti principali (PCA).

Consiste nel costruire M componenti principali (combinazioni lineari) da p predittori e utilizzarli come nuovi predittori in un modello di regressione lineare.



Come si può notare, l'MSE minimo si può ottenere considerando 32 predittori, mentre la dimensionalità originaria era 44.

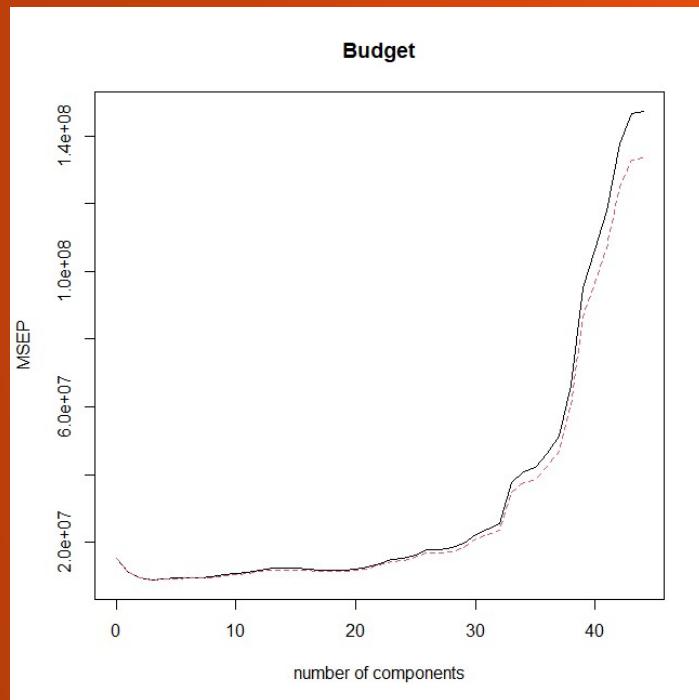
Di seguito viene riportata una tabella contenente un resoconto dei risultati ottenuti con tutti i modelli considerati applicando il Validation Set Approach per la stima dell'MSE sul test set.

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



PLS

Similmente alla PCR, anche la PLS si pone l'obiettivo di creare un nuovo set ridotto di variabili su cui effettuare una regressione lineare. Tuttavia, a differenza della PCR che utilizza solo le informazioni relative alle feature, la PLS avviene in maniera supervisionata, ovvero utilizza anche le informazioni contenute nella risposta Y.



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



Confronto PCR PLS

Viene riportata una tabella contenente, per ogni modello considerato, la proporzione della **varianza spiegata (PVE)** 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

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.

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



Progetto in python

Esecuzione del codice effettuata su Google Colab.

- Feature da classificare: **class**, che può assumere i due valori:
0 (banconote contraffatte)
1 (banconote autentiche)
- Features usate per la predizione: variance, skewness, curtosis, entropy.

Metodi di classificazione impiegati:

- Naïve Bayes
- KNN (K-Nearest Neighbors) - (Apache Spark)
- Naïve Kernel - (Apache Spark)
- Logistic Regression - (Apache Spark)



Dataset Classificazione

Dataset contenente 1372 istanze con 5 attributi.

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.

Descrizione attributi:

- variance: varianza dell'immagine calcolata con la trasformata Wavelet (continua).
- skewness: asimmetria dell'immagine calcolata con la trasformata Wavelet (continua).
- curtosis: curtosì (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



Preprocessing Python

Bilanciamento dei dati:

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 dei dati:

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.

```
1 from sklearn.preprocessing import MinMaxScaler  
2  
3 scaler = MinMaxScaler()  
4  
5 data=pd.DataFrame(scaler.fit_transform(balanced),  
6 | | | | | columns=balanced.columns, index=balanced.index)
```

	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



Matrici di Correlazione e Covarianza

Per verificare eventuali caratteri di indipendenza tra le features presenti nel dataset, si è calcolata la matrice delle covarianze per osservare il comportamento di ognuna rispetto a tutte le altre.

Da un primo sguardo possiamo 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.

Matrice di correlazione

	variance	skewness	curtosis	entropy
variance	1.000000	0.270477	-0.380265	0.282723
skewness	0.270477	1.000000	-0.794782	-0.511239
curtosis	-0.380265	-0.794782	1.000000	0.306923
entropy	0.282723	-0.511239	0.306923	1.000000

Matrice di covarianza

	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



Stima dei parametri statistici dei dati

Per consentire la generazione di un dataset sintetico, la cui analisi fornisce un supporto allo studio effettuato 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. Inoltre, è data la possibilità di selezionare una coppia di features, in maniera arbitraria, su cui vengono analizzate media, varianza, e covarianza tra le due.

```
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
```

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.

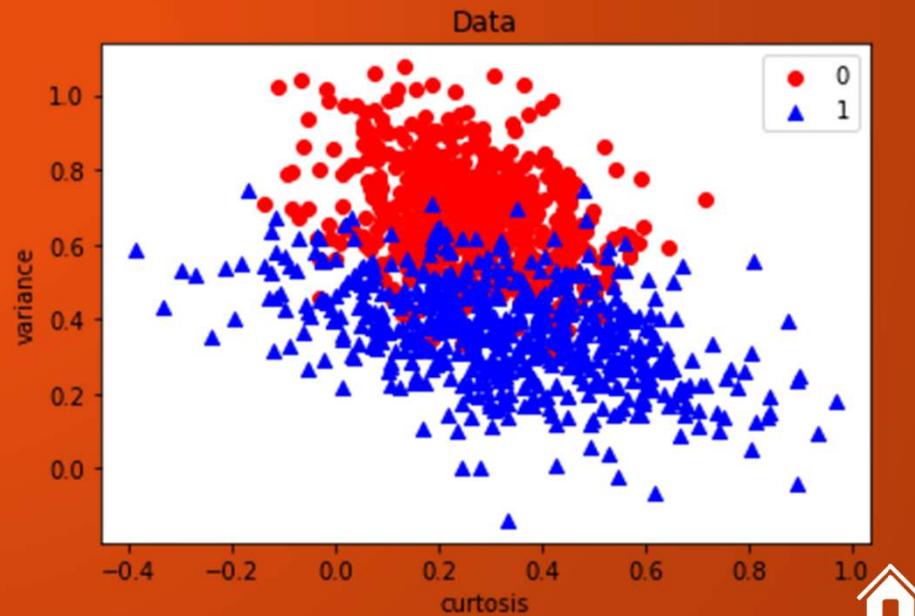


Generazione datasets sintetici

Per l'analisi dei dati si è preferito affiancare alle deduzioni fatte sul dataset reale, quelle fatte su dataset sintetici. In particolare si è optato per l'utilizzo di due distinti dataset:

1. generato a partire dalle stime di media, varianza e matrice di covarianza tra due feature scelte in maniera arbitraria e, quindi, contraddistinto da solamente due colonne per le X. In particolare, le features scelte per le seguenti analisi sono: curtosis e variance;
2. generato a partire dalle stime di media, varianza e matrici di covarianza su tutte le features del dataset.

Per generare tali dataset si è fatta l'assunzione che la distribuzione dei valori seguisse una Normale Multivariata: pertanto, utilizzando l'apposito metodo `multivariate_normal` vengono generati i campioni secondo i parametri di media e covarianza passati. Nell'immagine a lato è rappresentato il plot del dataset sintetico per 2 features.



Naïve Bayes

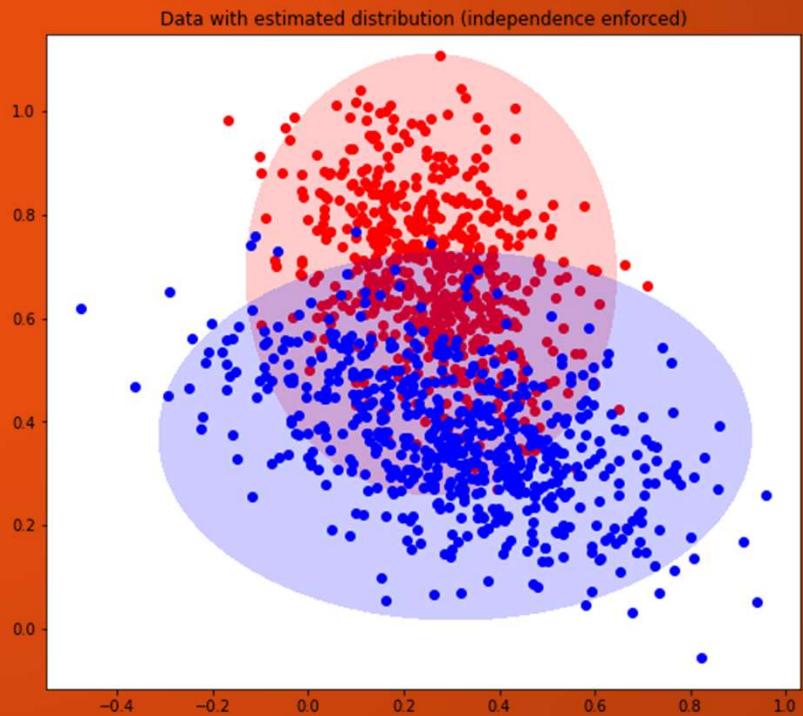
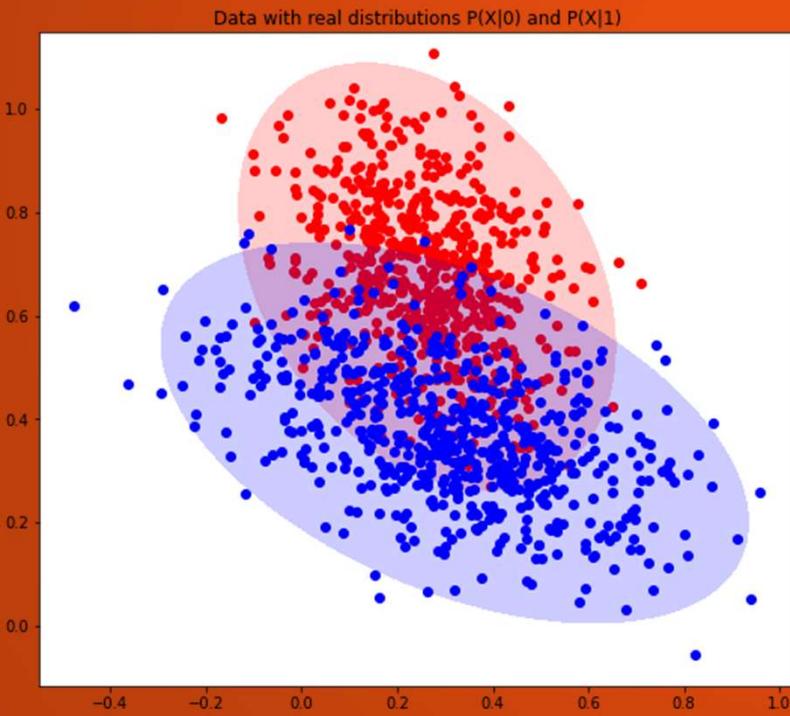
Il classificatore Naïve Bayes utilizza il teorema di Bayes per calcolare la probabilità di osservare una classe data la probabilità a posteriori, cioè le osservazioni delle features. Successivamente applica il criterio MAP (Maximum A Posteriori), vale a dire che è scelta quella classe che massimizza 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.



Naïve Bayes: 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.

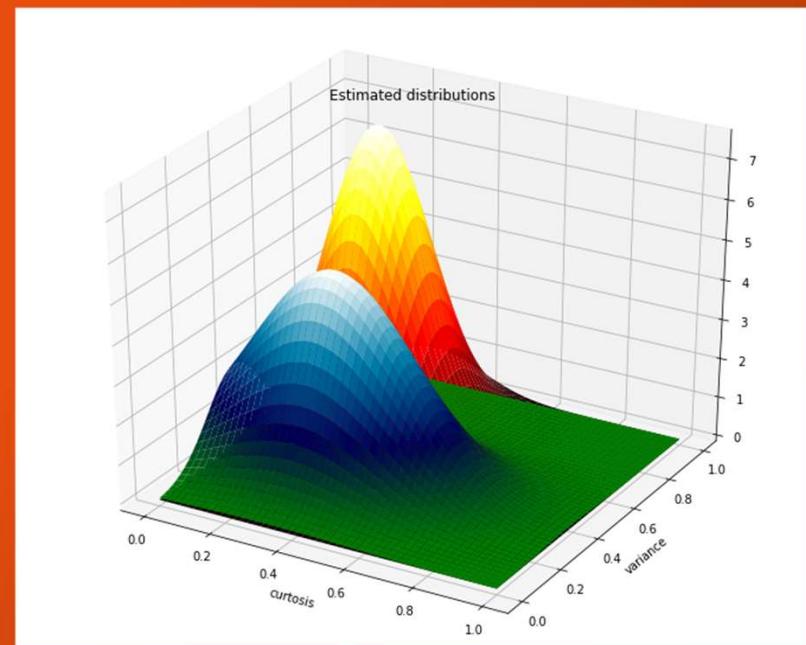
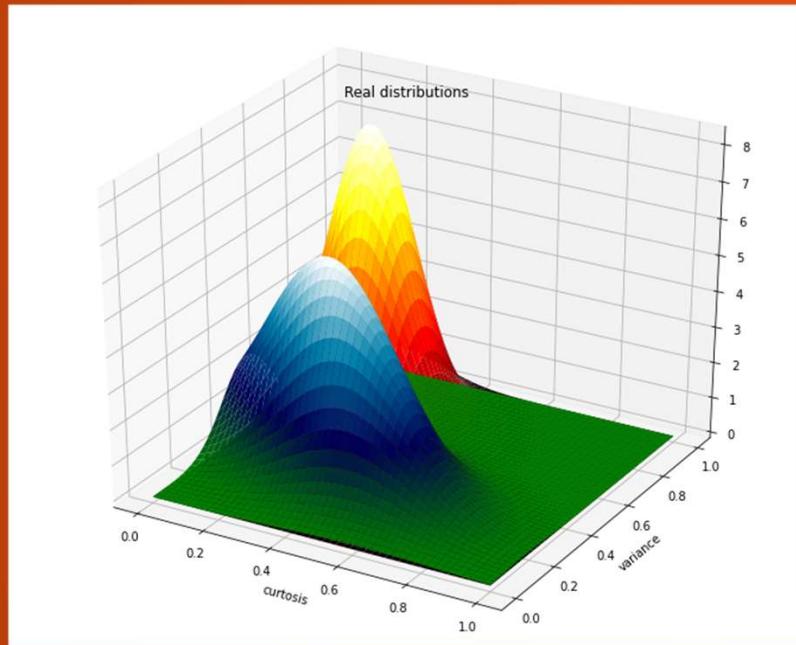
Sono state graficate le distribuzioni dei dati, per le due features curtosis e varianza, sia considerando il dataset sintetico, generato a partire dalle distribuzioni del dataset reale, sia considerando una stima della media e della varianza sui campioni del dataset sintetico stesso.



Naïve Bayes: indipendenza tra le features

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.

Da questi plot possiamo dunque stimare che, forzando l'ipotesi di indipendenza tra le due features, continuiamo comunque ad avere dati ben separabili.



Naïve Bayes: conclusioni

Osservando i grafici in 2D 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.



Naïve Bayes su dataset sintetico con due features

Le performances sono intorno all' 87% il che dipende dal fatto che il classificatore non tiene in considerazione la correlazione tra le 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%
```



Naïve Bayes su dataset sintetico usando tutte le features

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.

```
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     0.89     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%
```



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.

```
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      --       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%
```



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.

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

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



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

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



Naïve Bayes su dataset reale usando tutte le features disponibili

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.

```
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%
```



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 .

Un iperparametro di questo classificatore è il numero di vicini K da utilizzare nell'algoritmo. Un valore di K piccolo rende l'algoritmo più locale, mentre un valore di K più grande comporta la perdita di località. Al crescere del numero di campioni N , K deve necessariamente crescere.

$$m_n(x) = \frac{\sum_{i=1}^n Y_i(x)}{K}$$

Per classificare, l'algoritmo calcola le “distanze” tra le X di test e le X nel training set, e sceglie i K samples più vicini. Tra le diverse tipologie di distanze è stata scelta la distanza euclidea:

$$dist(X, Y) = \sqrt{\sum_{i=1}^N (x_i - y_i)^2}$$



KNN: divisione del dataset

Si è optato per una divisione del dataset in 3 parti:

- Training set 50% : usato per l'estrazione dei K vicini alle X di test;
- 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 performance finali in termini di corretta classificazione.



KNN: implementazione realizzata

Sono state previste due funzioni per l'addestramento e la predizione:

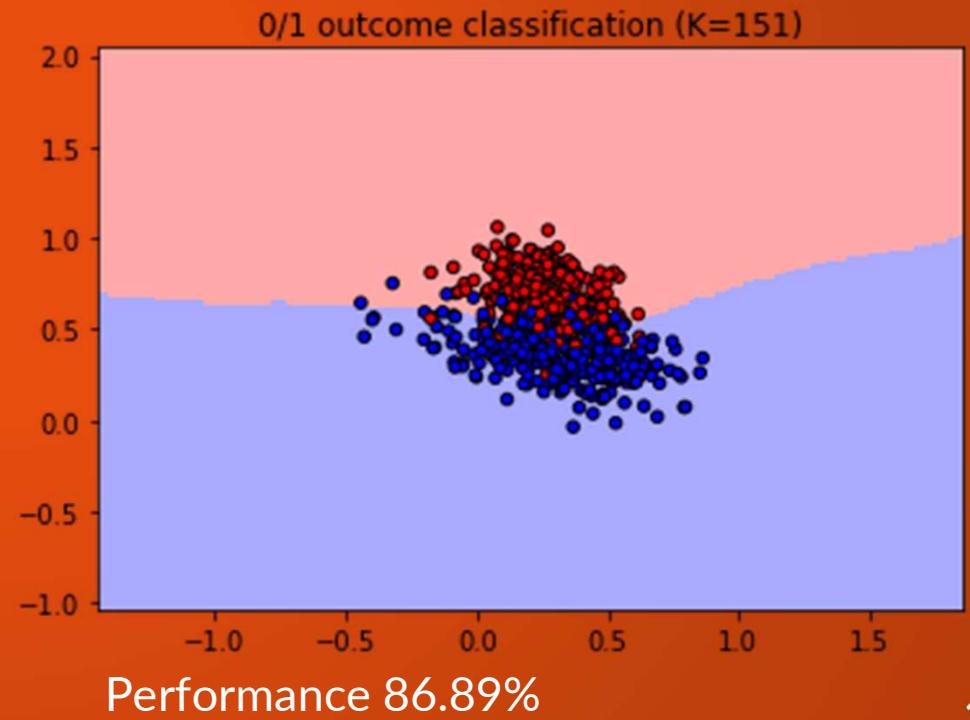
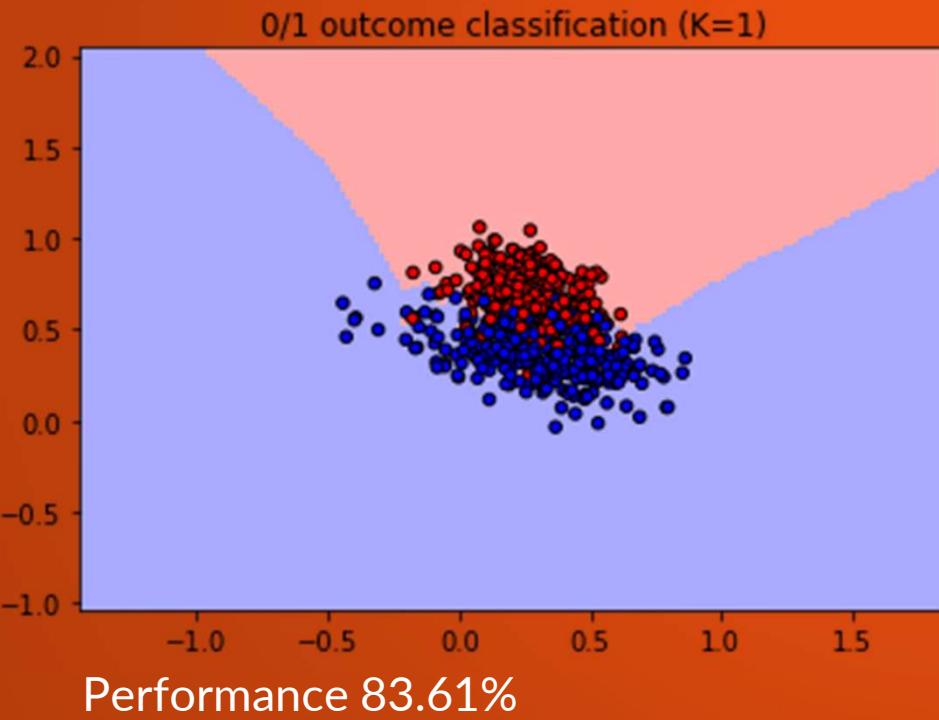
- **KNN_fit_and_predict_spark**: effettua l'addestramento e la predizione utilizzando Spark.
- **KNN_fit_and_predict_no_spark**: effettua l'addestramento e la predizione usando il classificatore KNN messo a disposizione dalle librerie sklearn.
- I risultati tra le due funzioni precedenti possono essere comparati utilizzando **KNN_fit_and_predict_comparison**.
- **KNN_plot**: utilizzata per andare a plottare i risultati della classificazione, andando a realizzare una meshgrid, e facendo predizioni per tali valori.
- **KNN_tuning**: effettua l'ottimizzazione del parametro k.



KNN su dataset sintetico con due features

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.

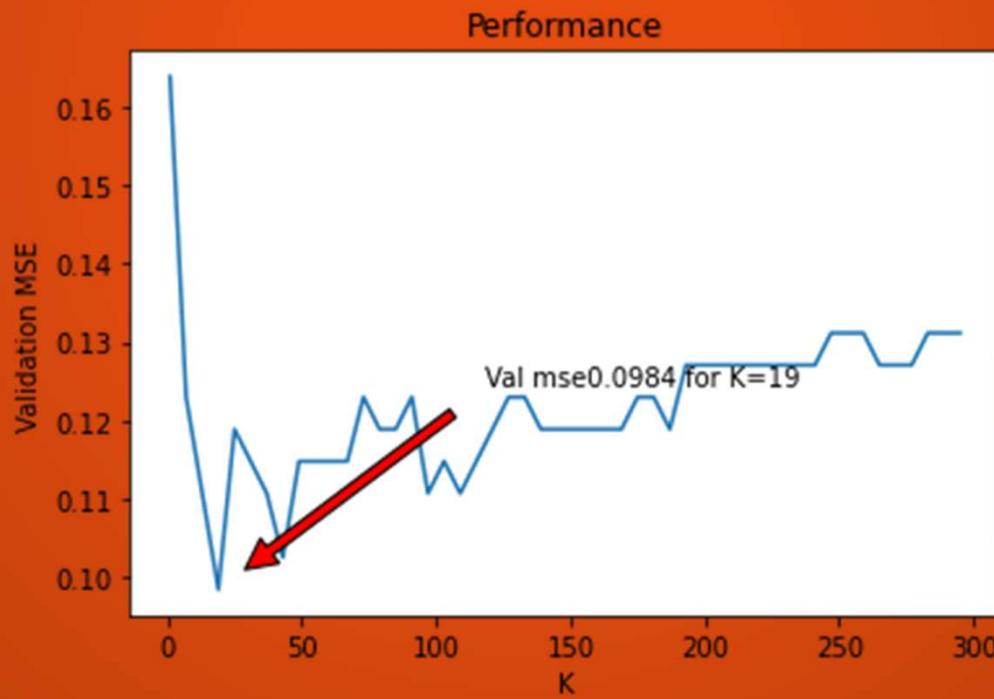
Più aumenta K, più punti si stanno considerando all'interno dell'algoritmo.



KNN su dataset sintetico con due features

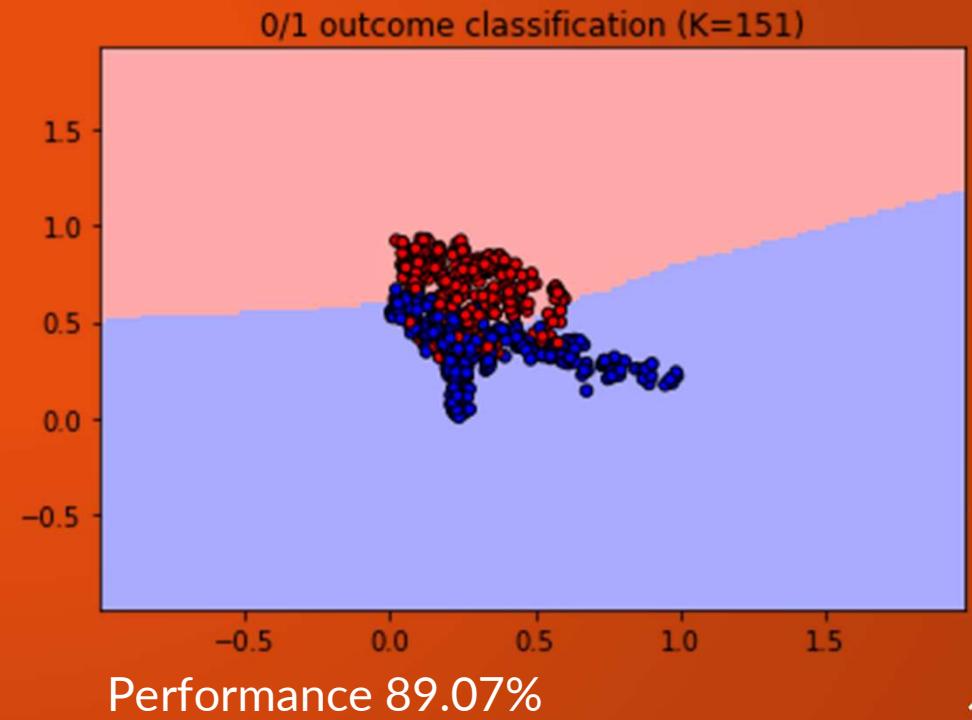
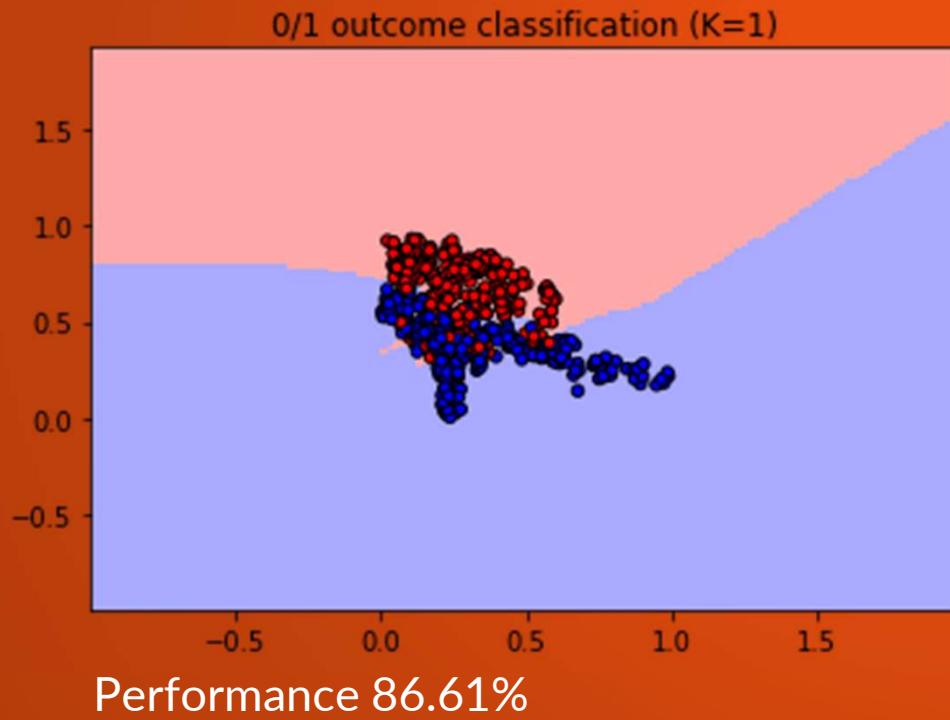
Si può notare che il modello riesce a separare più o meno correttamente le due regioni. Oltre ai grafici delle regioni di decisione, lo script grafica anche l'andamento delle prestazioni del modello, in base al valore dell'MSE, indicando quale sia il valore di K che restituisce il risultato migliore.

Performance migliore: 86.34% per K=19.



KNN su dataset reale con due features

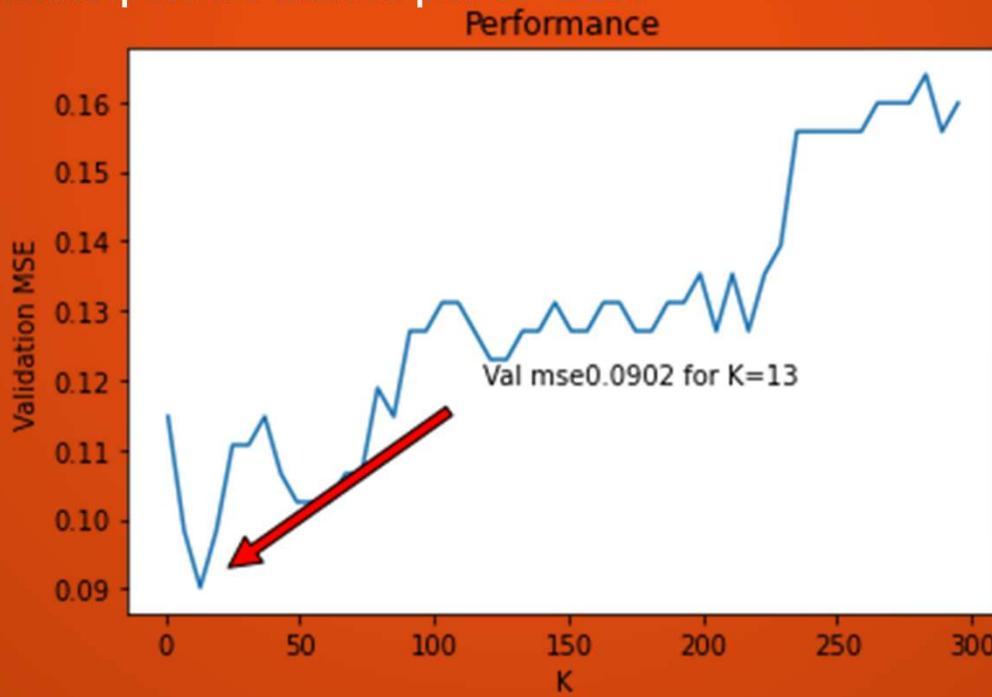
Avendo quindi dimostrato tramite il dataset sintetico che l'implementazione parallelizzata tramite Spark è completamente funzionante e corretta, il classificatore KNN è stato applicato anche al dataset reale utilizzando la sola implementazione di libreria.



KNN su dataset reale con due features

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.

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.



KNN su dataset reale usando una feature per volta

Al fine di verificare quali siano quelle features che maggiormente caratterizzano gli elementi di una classe, si è deciso di effettuare un fit su ogni singola feature presente nel dataset. Osserviamo i valori di K che garantiscono il minor MSE sul validation set, ed i valori per le performances.

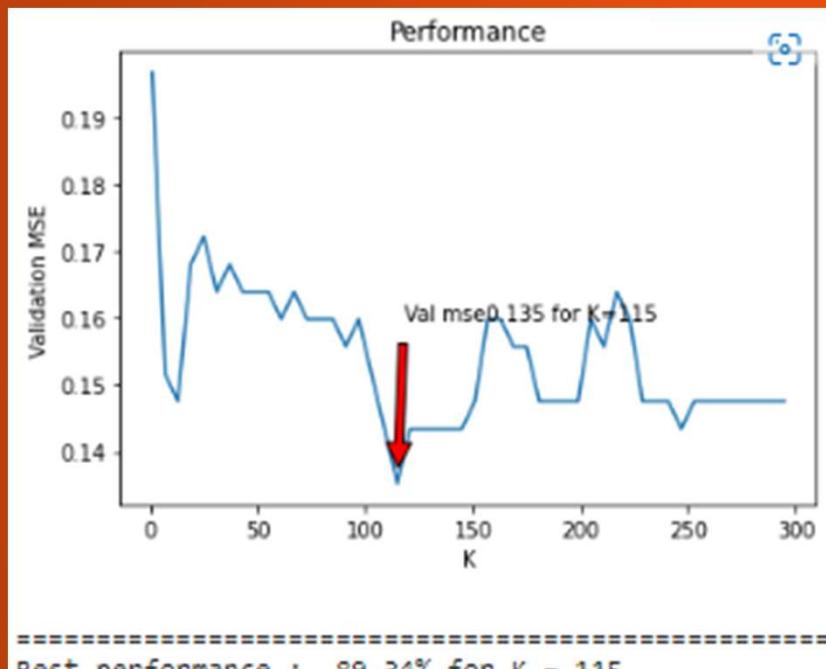
A causa della diversa distribuzione dei dati, i valori di K sono diversi per tutte le features.

	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%

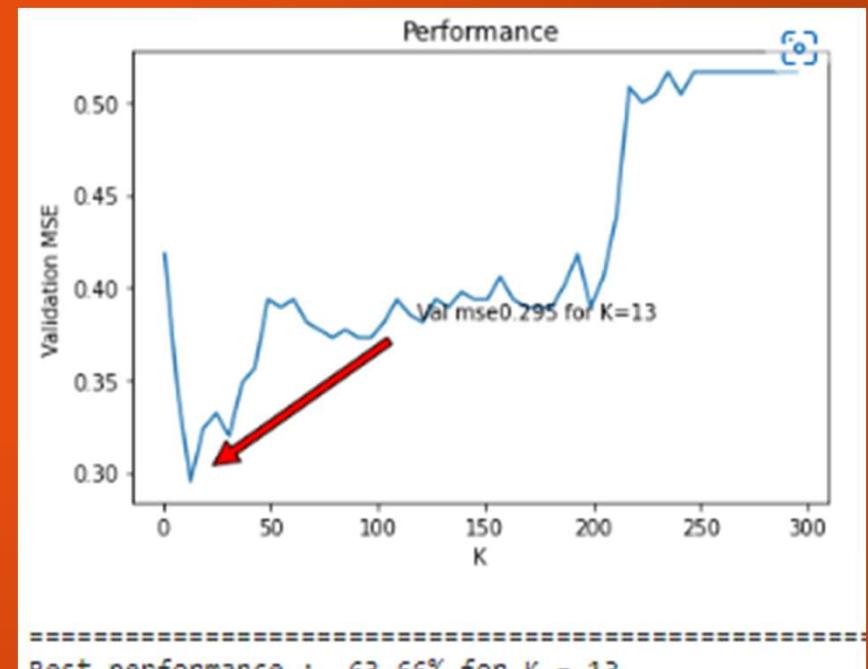


KNN su dataset reale usando una feature per volta

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.



Performance variance



Performance curtosis



KNN su dataset reale usando tutte le features disponibili

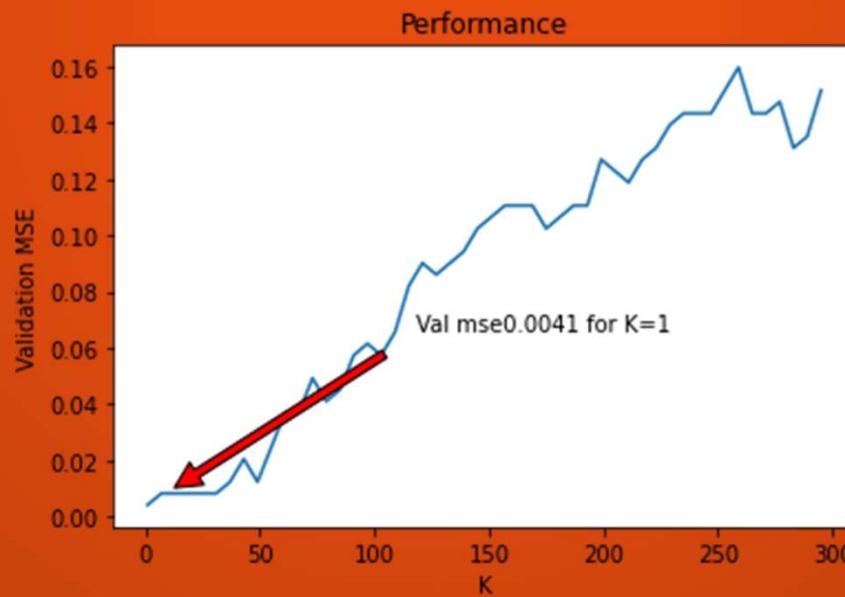
Le migliori performances sul Test set sono ottenute per K prossimi ad 1.

```
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
```



KNN su dataset reale usando tutte le features disponibili

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.



KNN su dataset sintetico usando tutte le features insieme

Per motivi legati alle limitazioni di Colab e di tempo, non si è potuti impiegare l'intero dataset sintetico creato, ma solamente una porzione. In tal senso, sono stati realizzati due test:

- il primo con 1220×2 istanze del dataset sintetico;
- il secondo 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.

Data la non bidimensionalità del problema non risulta possibile riportare i vari grafici.



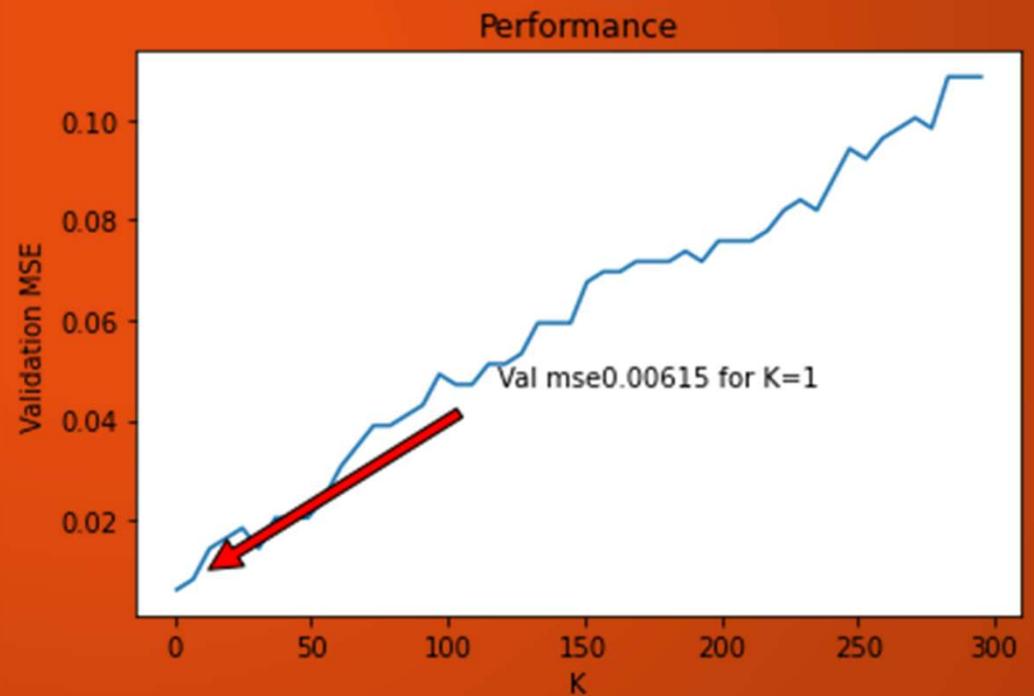
Numero di campioni N*2

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%.

```
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      0.99      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
```

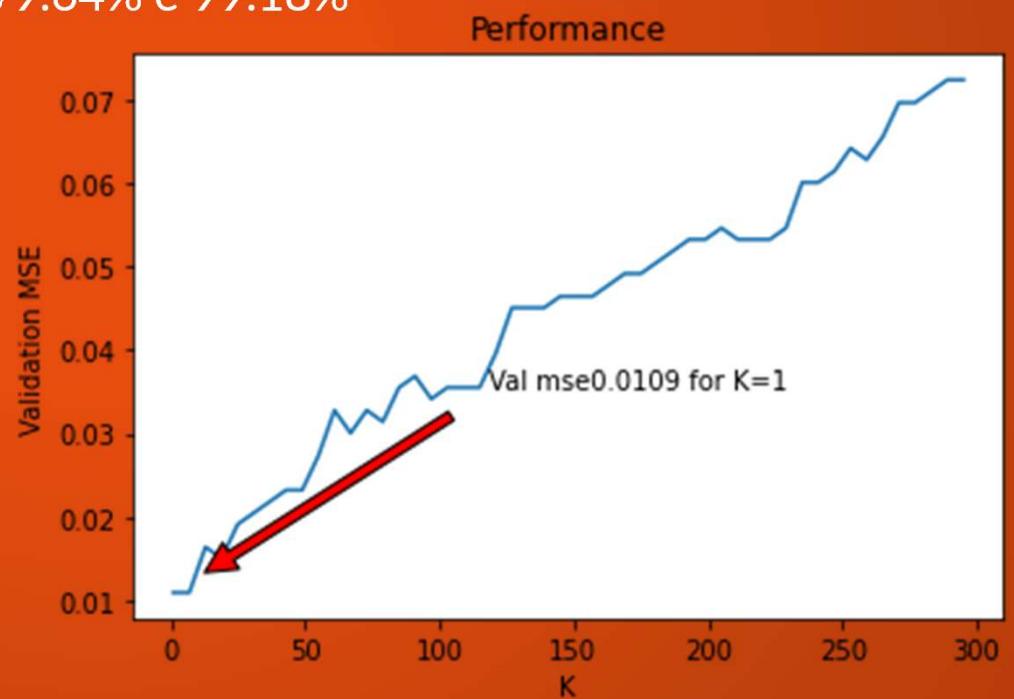


Numero di campioni N*3

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%

```
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
```



KNN: Conclusioni

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.



Naïve 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. 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 è stata scelta 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.

$$m_n(x) = \frac{\sum_{i=1}^n I(|x_i - x| \leq h) Y_i}{\sum_{i=1}^n I(|x_i - x| \leq h)}$$



Naïve Kernel: scelta del parametro h

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()`).



Naïve Kernel: 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.

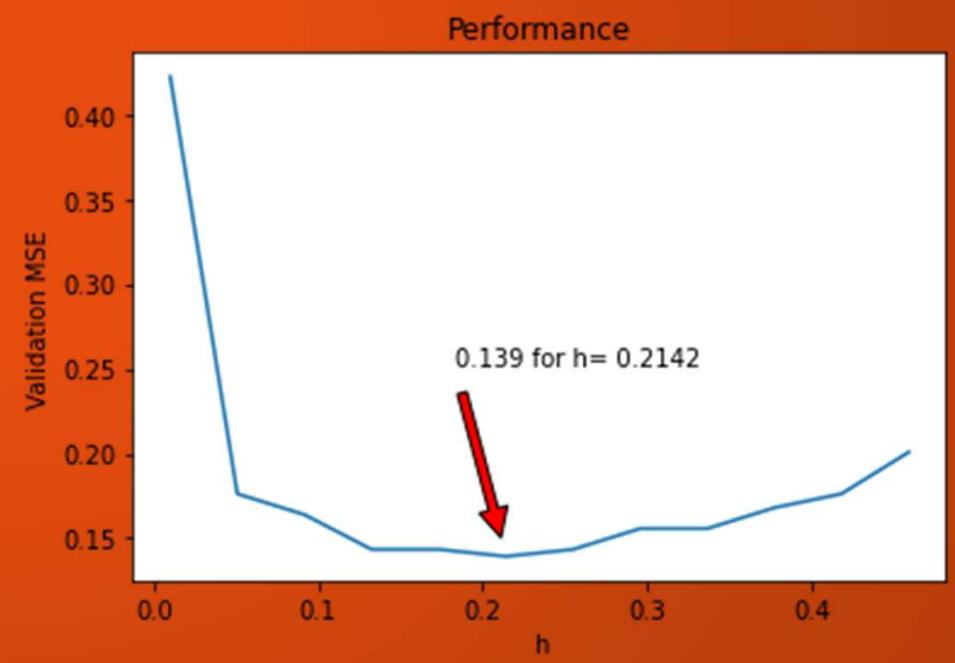
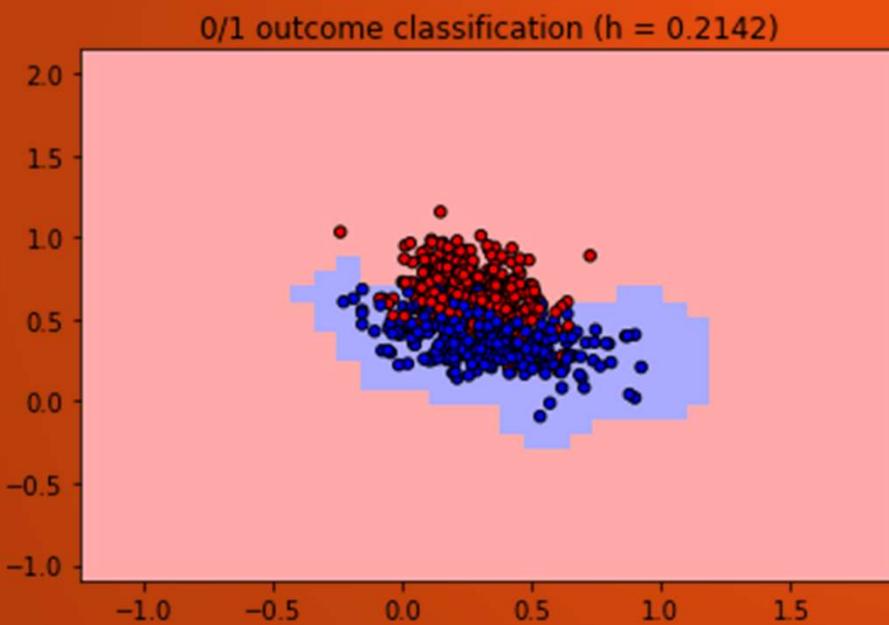
- **NaiveKernel_fit_and_predict:** effettua l'addestramento e la predizione di una Y utilizzando Spark;
- **NaiveKernel:** ottenute le predizioni dal metodo precedente, questa funzione si occupa di andare a determinare le performance sui dati di test, compreso l'MSE;
- **Naive_kernel_plot:** effettua la stampa della predizione usando il valore h passato;
- **NaiveKernel_tuning:** permette di ottimizzare il parametro h. 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 performance ed MSE sul Test set e sul Validation Set. Sulla base dell'MSE di Validation, viene determinato il valore di h ottimale.



Naïve Kernel su dataset sintetico con due features

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%). Osserviamo che i risultati del modello Naïve Kernel su questa coppia di features sono in linea con quelli ottenuti finora.

Test performance: 87.16% e Test MSE: 0.1284 per $h = 0.2142$.

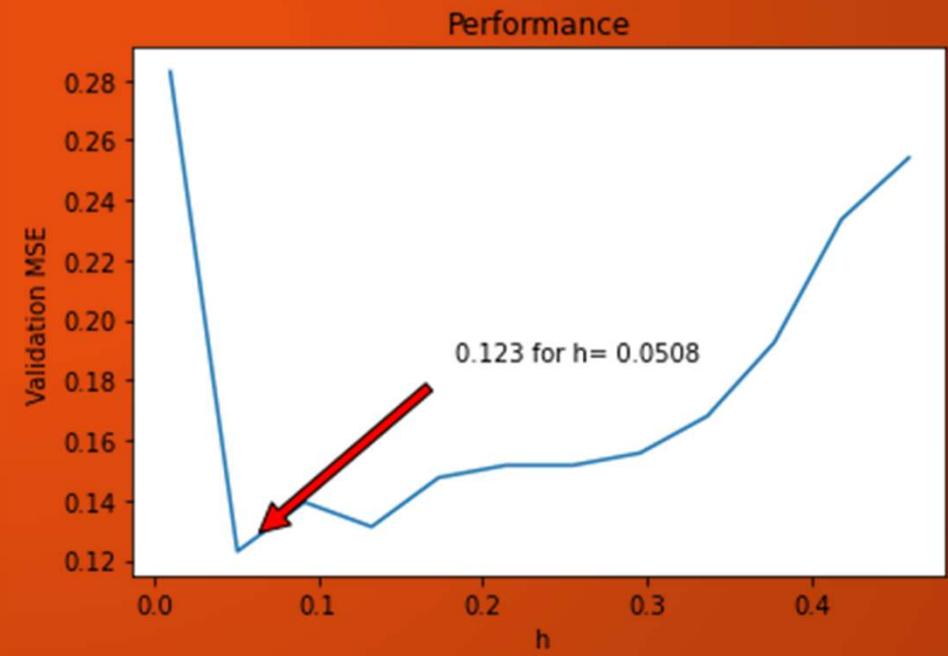
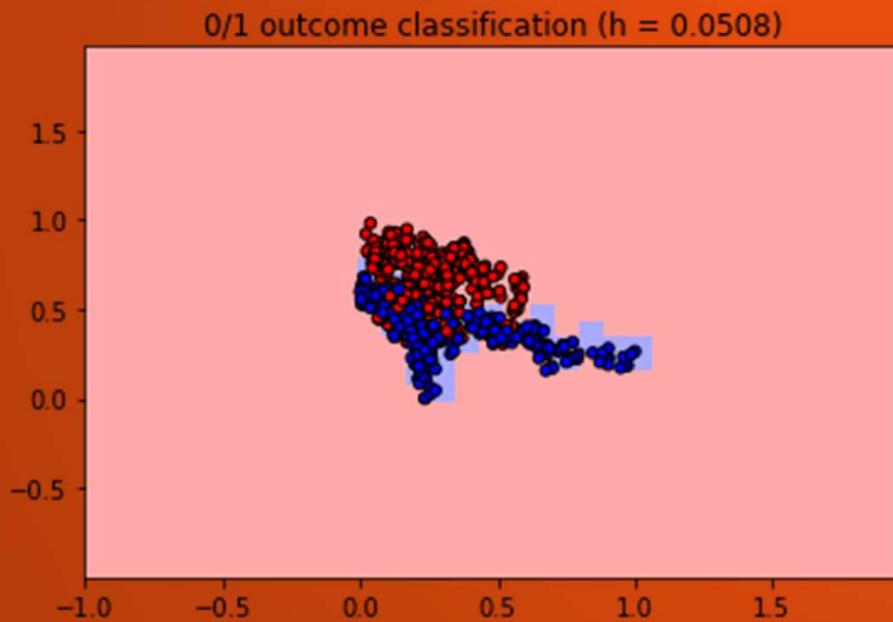


Naïve Kernel su dataset reale con due features

Al contrario di quanto accaduto prima, si evince 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.

Test performance: 90.71% e Test MSE: 0.0929 per $h = 0.0508$.



Naïve Kernel su dataset reale usando una feature per volta

È 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	0.00999999776482582	0.00999999776482582
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%

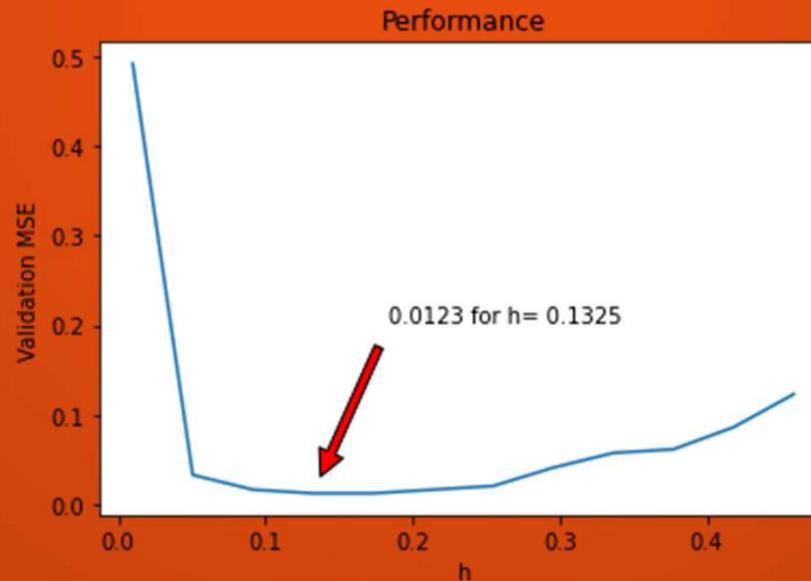


Naïve Kernel su dataset reale usando tutte le features disponibili

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.

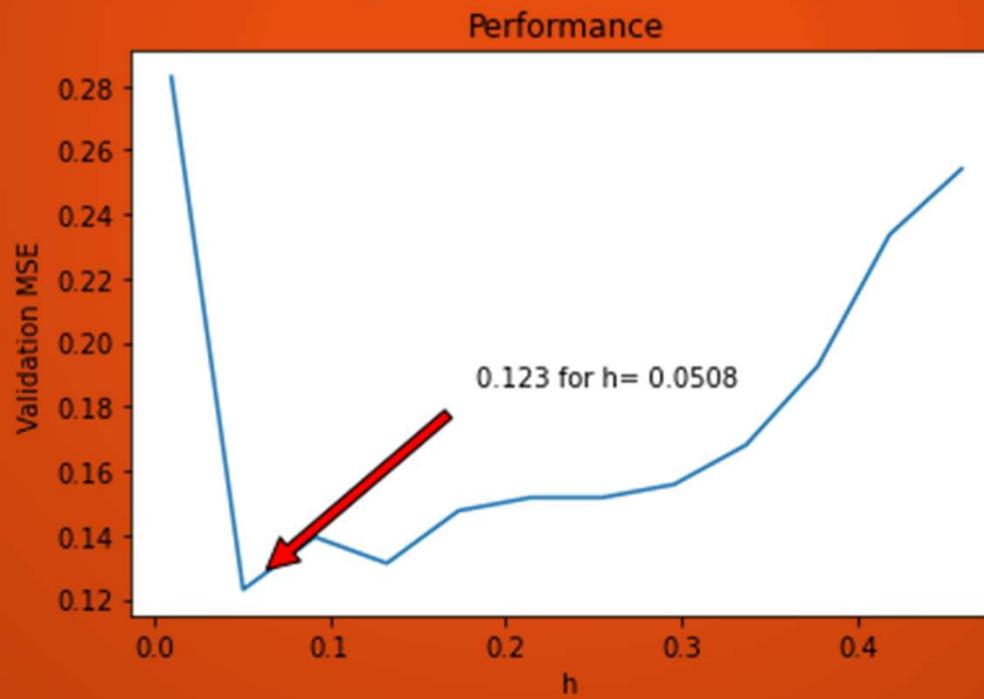
In linea con tutti i metodi di classificazione visti finora si può vedere come, coinvolgendo nella classificazione tutte le features presenti nel dataset si ottiene un miglioramento in termini di performances e calcolo dell'errore.

Test performance: 99.45% e Test MSE: 0.0055 per $h = 0.1325$.



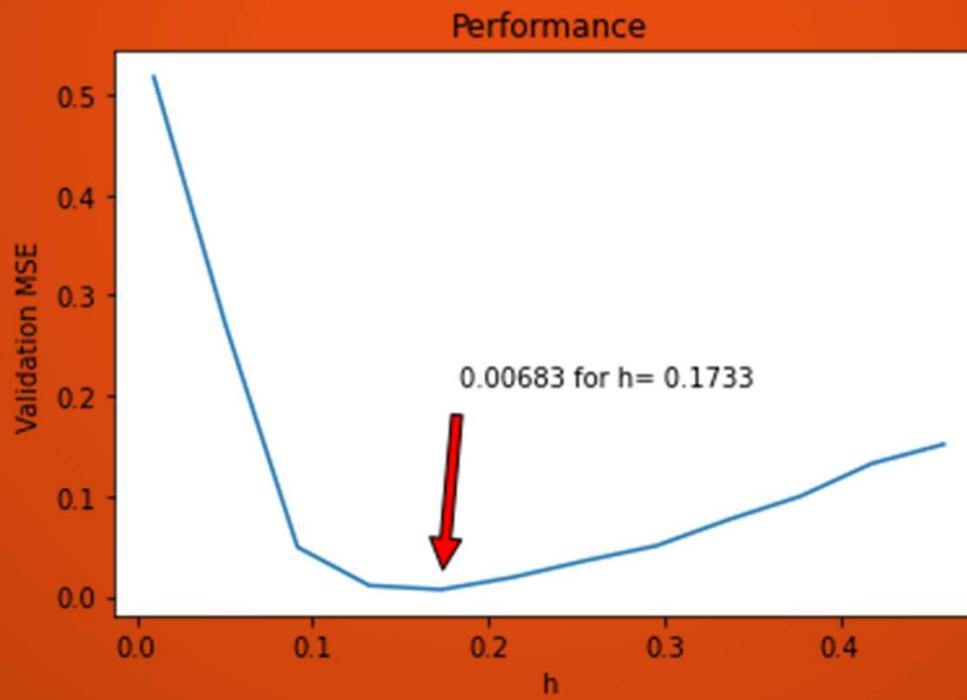
Naïve Kernel su dataset sintetico usando tutte le features: numero di campioni N*2

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.



Naïve Kernel su dataset sintetico usando tutte le features: numero di campioni N*3

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 ottengono ancora una volta per h uguale a 0.1733, con performance sul Test set del 97.63%.



Regressione Logistica

La regressione logistica rappresenta un metodo di classificazione rientrante nella famiglia degli algoritmi di apprendimento supervisionato. Essa si avvale della funzione sigmoide:

$$\sigma(x) = 1 / (1 - e^{-x})$$

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 $Y = (1, 0)$ nel seguente modo:

$$Y = 0 \rightarrow P[Y|X] = 1 / (1 + e^{(w^T x)})$$

$$Y = 1 \rightarrow P[Y|X] = e^{(w^T x)} / (1 + e^{(w^T x)})$$

X è classificato come appartenente alla classe 1 se $w^T x > 0$, ovvero

$P[1 | X] = e^{(w^T x)} / (1 + e^{(w^T x)}) > 1/2$, ed è classificato, al contrario, come appartenente alla classe 0 se $w^T x < 0$, ovvero $P[0|X] = 1 / (1 + e^{(w^T x)}) > 1/2$.



Funzione di costo e gradiente discendente

Si ha 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:

$$J(w) = (-1/N) * \sum_{i=1}^N [y_i * \log(P[1 | x_i]) + (1 - y_i) * \log(P[0 | x_i])]$$

Per l'apprendimento si applica l'algoritmo del Gradiente discendente, che aggiorna ciclicamente il vettore dei pesi w , tenendo in considerazione il learning rate o step size α e determinando la velocità di convergenza:

$$w = w - \alpha * \frac{d}{dw} J(w)$$

$$\frac{d}{dw} J(w) = \sum_{i=1}^N [(\sigma(w^T x) - y_i) * x_i]$$

Il compito di tale algoritmo è quello di andare a minimizzare $J(w)$ andando nella direzione negativa della sua derivata.



Regressione Logistica: divisione dataset

Il dataset è stato diviso in 2 parti:

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



Regressione Logistica: implementazione realizzata

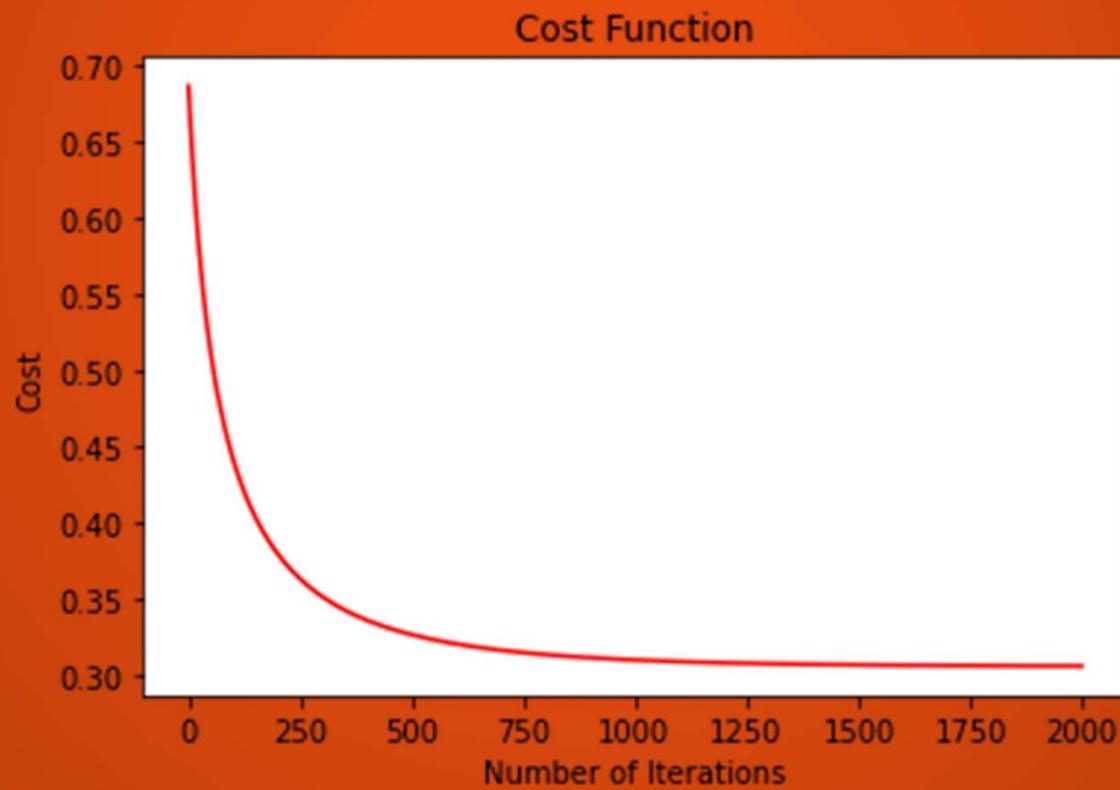
Per implementare il classificatore binario Logistic Regression tramite Spark, è stato necessario definire le seguenti funzioni:

- **sigmoid**: la funzione sigmoide utilizzata;
- **logReg_gradient**: effettua il calcolo del gradiente su un punto del Training set;
- **compute_cost**: da in output il valore di costo, calcolato rispetto al dataset X, il peso corrente w e le labels attuali;
- **gradient_descent_spark**: qui è 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 and reduce;
- **predict_spark**: effettua la predizione delle labels;
- **score_spark**: calcola la percentuale di labels che sono state predette in maniera corretta rispetto a quelle di test passate;
- **logistic_regression_spark**: calcola la regressione logistica utilizzando gli algoritmi precedentemente descritti;
- **logistic_regression**: calcolo della regressione logistica utilizzando le funzioni di libreria.



Regressione Logistica su dataset sintetico con due features

È possibile analizzare il grafo che mette in relazione la funzione di costo con il numero di iterazioni. Al crescere delle iterazioni, il costo tende progressivamente a decrescere.

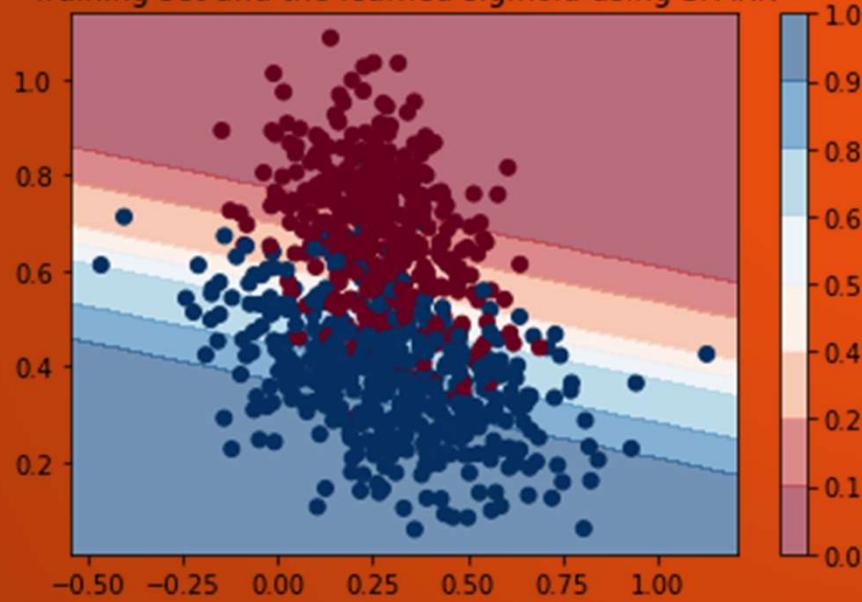


Regressione Logistica su dataset sintetico con due features

Confrontando le performances con l'utilizzo di Spark e senza, queste risultano essere molto vicine. Quindi nel complesso l'algoritmo implementato sembra essere soddisfacente.

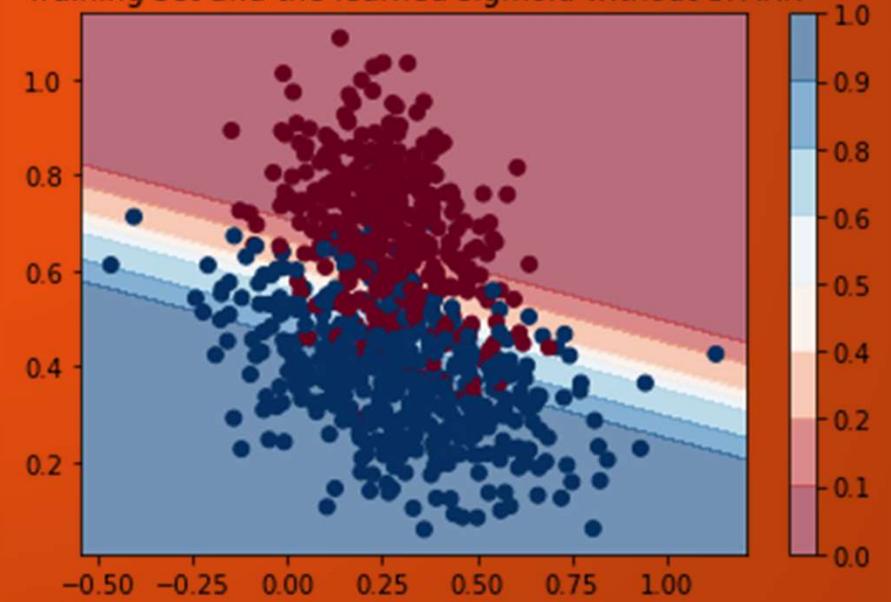
Performance con SPARK: 86.07%.

Training set and the learned sigmoid using SPARK



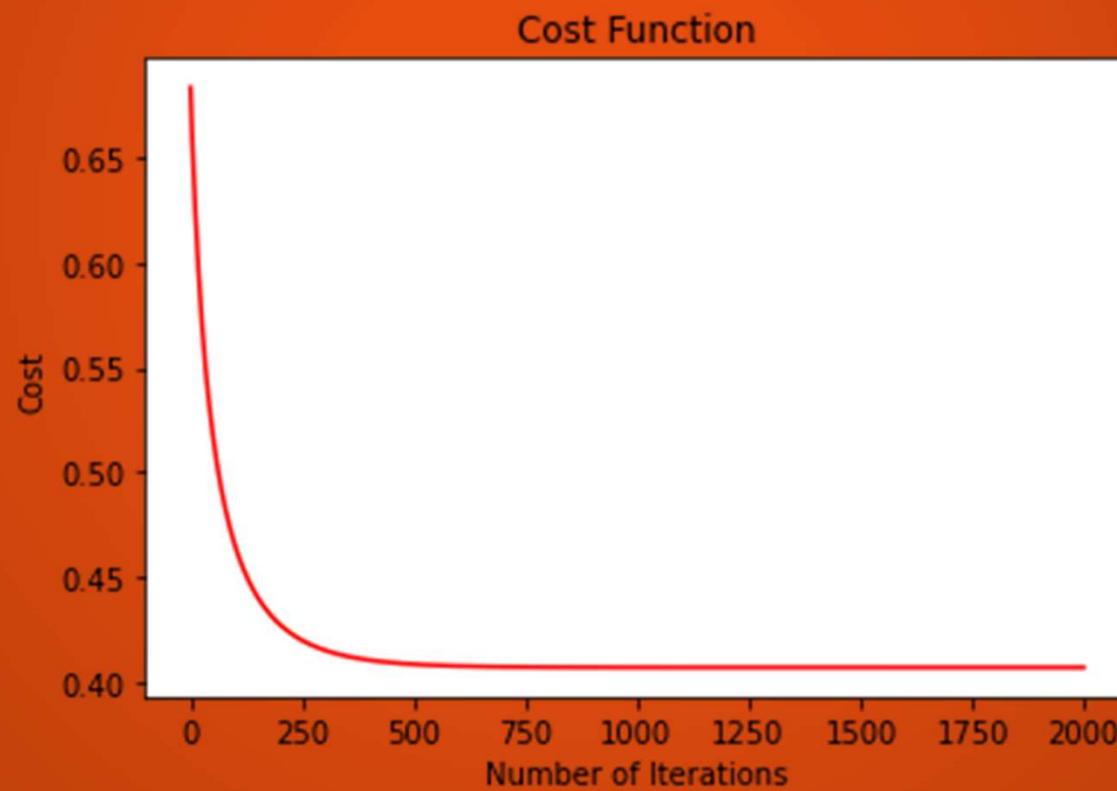
Performance senza SPARK: 85.79%.

Training set and the learned sigmoid without SPARK



Regressione Logistica su dataset reale con due features

Ancora una volta viene graficata la funzione di costo in relazione al numero di iterazioni settate.

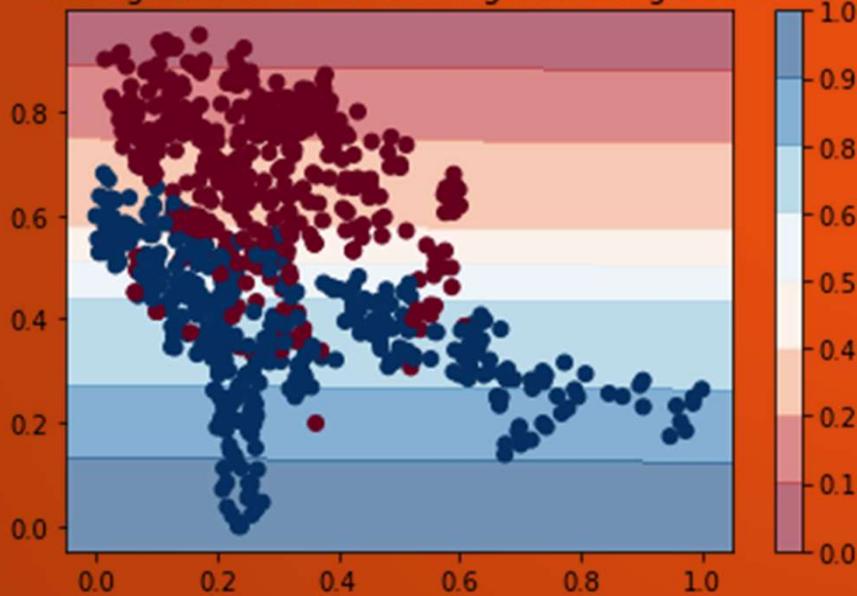


Regressione Logistica su dataset reale con due features

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.

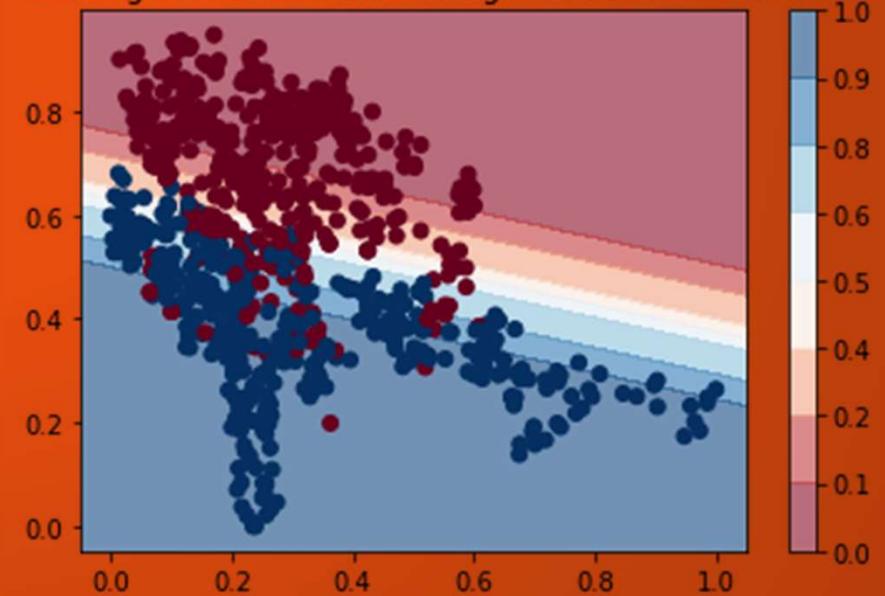
Performance con SPARK: 81.69%

Training set and the learned sigmoid using SPARK



Performance senza SPARK: 88.25%.

Training set and the learned sigmoid without SPARK



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.

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

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



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

Come si nota ancora una volta, la presenza 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.

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



Regressione Logistica su dataset reale usando tutte le features

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

```
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     0.98     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%
```



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.

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 0.99 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%
```





Grazie per
l'attenzione