Corso di

Data Mining

Studente: Marco Romanelli 1076067

24/06/2016

Introduzione

Il dataset preso in esame contiene una serie di informazioni ricavate durante la codifica/decodifica di video in altri formati, insieme ad alcune caratteristiche stesse dei video.
Lo scopo è quello di creare un modello predittivo per prevede il tempo di codifica/decodifica basandosi sulle informazioni in possesso.
I video sono stati scelti casualmente da un insieme di campioni presi dal noto sito web di condiviso video Youtube.
Il dataset è reperibile alla pagine web del UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/).

Dati

L'insieme contiene 68784 istanze e 20 variabili. Di seguito viene riportata una breve spiegazione per ognuna:

- *id*: identificativo del video (da YouTube),
- *duration*: durata del video,
- *codec*: codec utilizzato dal video.
- width, h*eight*: larghezza e altezza del video.
- *bitrate*,
- *framerate*,
- i, p, b: numero di frame i/p/v nel video,
- *frames*: numero di frame nel video
- i_size, p_size, b_size: dimensione del video i/p/b,
- *size*: dimensione totale del file,
- *o_(codec, bitrate, framerate, width, height)*: caratteristiche del video di output
- *umem*: memoria totale allocata per la codifica
- *utime*: tempo totale della codifica

Ambiente di sviluppo

Come ambiente di sviluppo è stato scelto l'ambiente R (https://www.r-project.org).

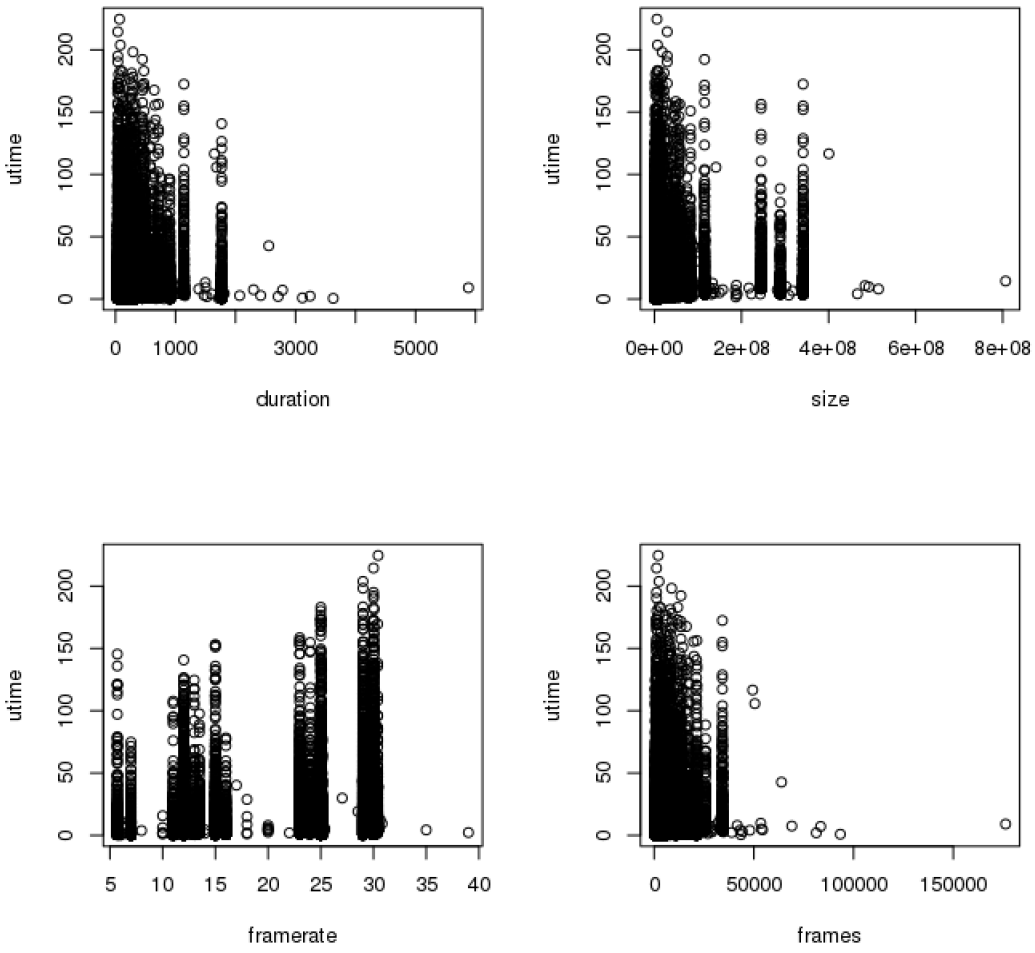
Analisi preliminare dei dati

Per prima cosa si decide di eliminare la variabile *id*, che non aggiunge nessuna informazione utile per lo scopo in questione.
Consideriamo poi le la variabili *codec* e *o_codec* come variabili quantitative.

```
> transcoding$codec <- as.factor(transcoding$codec)
> transcoding$o_codec <- as.factor(transcoding$o_codec)
```

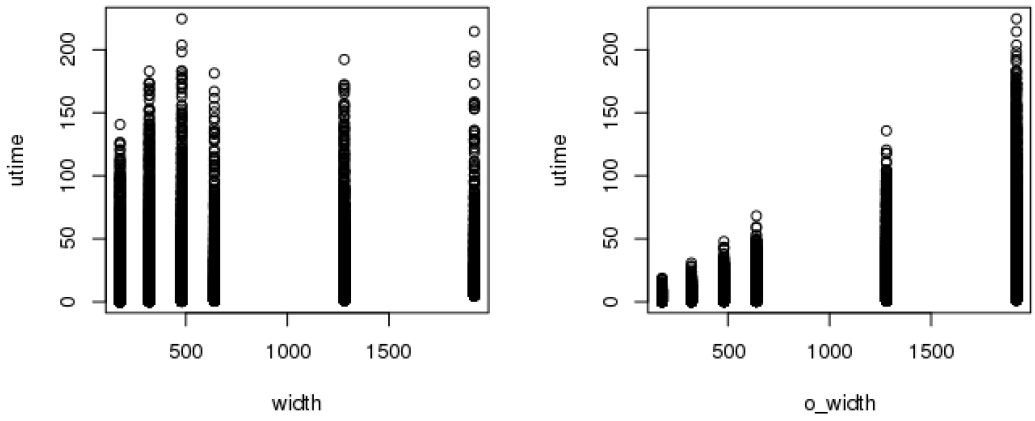
Analizzando le variabile che rappresentano i frame del video si osserva che frames rappresenta esattamente la somma delle variabili *i*, *p* e *b*.
Si pensa, di conseguenza, che questo avvenga anche per le variabili *i_size, p_size, b_size* e *size;* tuttavia si vede come ci siano esattamente 859 istanze per cui questo non avviene.
Osservando in dettaglio si vede come queste istanze siano le sole e uniche ad avere la variabile *b* diversa da '0'.
Per decidere se tali dati siano da modificare o rimuovere si è costruito un modello costruito solamente su questi dati; in questo modello la variabile *b* non sembra significativa.

Inoltre il modello costruito sul sottoinsieme del dataset ottenuto rimuovendo questi dati non sembra ottenere un miglioramento (rispetto al modello sull'intero dataset).
Per questi motivi si è deciso di lasciare queste istanze nel dataset.



Osservando i grafici si vede come molte variabili sembrano avere una relazione inversa con la variabile risposta *utime,* mentre sembra esserci una relazione crescente (lineare o parabolica) con la variabile *framerate*.

Un relazione interessante può essere osservata dai plot riguardanti le variabili width e o_width (sempre in relazione alla variabile risposta). Il comportamento delle due variabili sembra abbastanza diverso fra loro.



Analisi

Il modello lineare

Per prima cosa il dataset viene suddiviso in un *train set* e un *validation set*, scelti casualmente, dove il primo contiene 2/3 del totale mentre il secondo il restante 1/3.

```
> n <- floor(0.75 * nrow(transcoding))
> set.seed(Sys.time())
> index <- sample(seq_len(nrow(transcoding)), size = n)
> train <- transcoding[index, ]
> test <- transcoding[-index, ]
```

Si parte quindi dal modello più semplice che comprende tutte le variabili.

```
> fm <- lm(utime ~., data=train)
> summary(fm)

Call:
lm(formula = utime ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-42.427  -4.559  -1.046   3.143 161.751

Coefficients: (2 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.890e+01  3.750e-01 -77.082  < 2e-16 ***
duration      8.310e-04  4.743e-04   1.752 0.079737 .
codech264     6.834e-01  1.779e-01   3.841 0.000123 ***
codecmpeg4   -2.147e-01  1.992e-01  -1.078 0.281105
codecvp8     -2.834e-01  2.132e-01  -1.329 0.183739
width         9.179e-03  9.314e-04   9.855  < 2e-16 ***
height       -1.704e-02  1.852e-03  -9.201  < 2e-16 ***
bitrate       1.727e-06  9.295e-08  18.584  < 2e-16 ***
framerate     7.962e-02  1.171e-02   6.798 1.08e-11 ***
i            -6.940e-03  1.456e-03  -4.766 1.89e-06 ***
p             8.491e-05  3.062e-05   2.773 0.005551 **
b            -3.352e-05  6.988e-04  -0.048 0.961743
frames               NA         NA      NA       NA
i_size       -1.257e-06  8.260e-07  -1.522 0.127933
p_size       -1.087e-06  8.250e-07  -1.318 0.187476
b_size               NA         NA      NA       NA
size          1.093e-06  8.250e-07   1.325 0.185233
o_codech264   1.202e+01  1.288e-01  93.294  < 2e-16 ***
o_codecmpeg4  2.807e+00  1.184e-01  23.706  < 2e-16 ***
o_codecvp8    8.631e+00  1.185e-01  72.864  < 2e-16 ***
o_bitrate     1.411e-06  2.393e-08  58.977  < 2e-16 ***
o_framerate   2.493e-01  6.277e-03  39.722  < 2e-16 ***
o_width       7.066e-03  6.587e-04  10.727  < 2e-16 ***
o_height      4.599e-03  1.267e-03   3.631 0.000283 ***
umem          7.052e-05  5.287e-07 133.391  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.518 on 51565 degrees of freedom
Multiple R-squared:  0.6518,  Adjusted R-squared:  0.6517
```

```
F-statistic:  4388 on 22 and 51565 DF,  p-value: < 2.2e-16
```

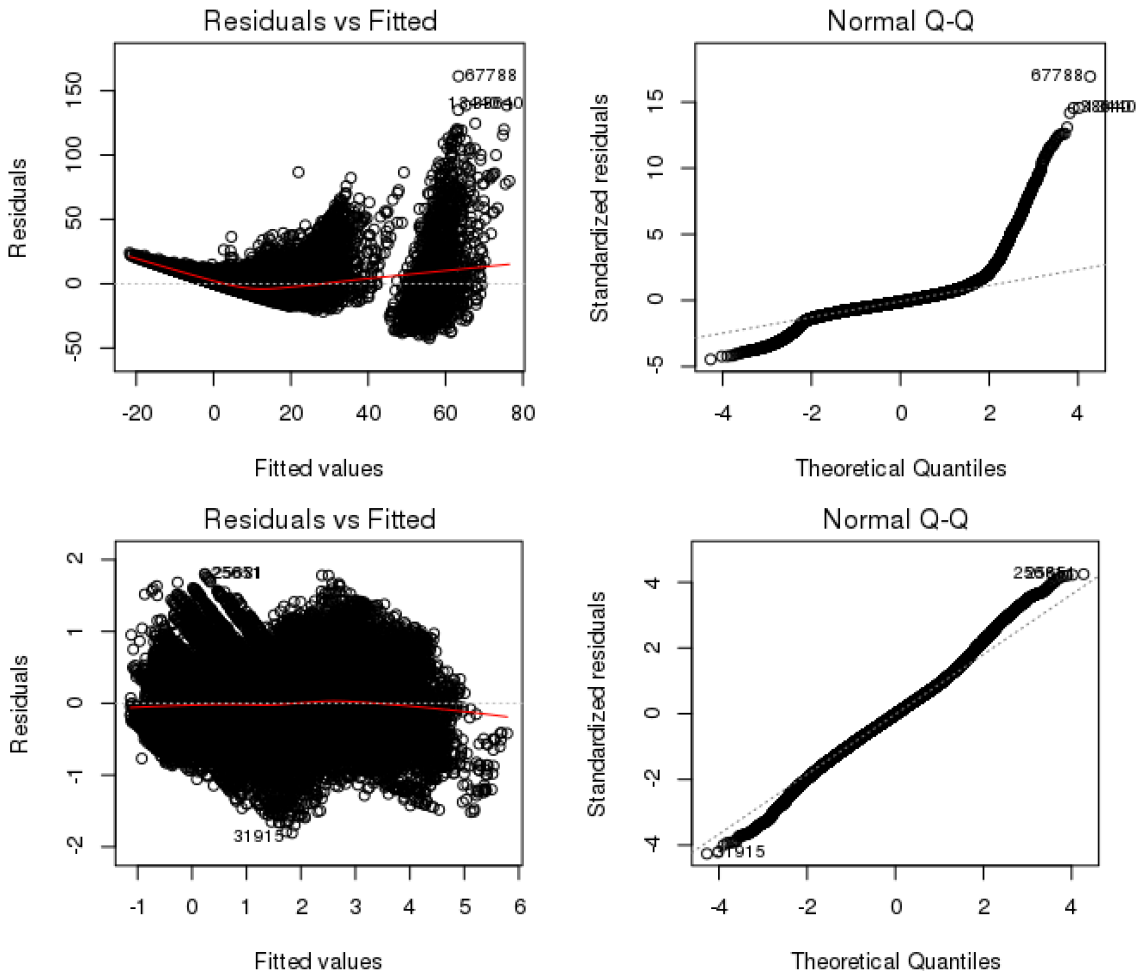
Seppur molto semplice, il modello è abbastanza significativo e cattura un buon 65% della variabilità.
Il passo successivo consiste nell'eliminare le variabili non significative (quelle con *p-value* superiore a 0.01) utilizzando la *backward-elimination*.
Si arriva all'ultimo passo con il modello seguente:

```
> summary(fm2)
Call:
lm(formula = utime ~ width + height + bitrate + framerate + i_size +
    size + o_codec + o_bitrate + o_framerate + o_width + o_height +
    umem, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-42.485  -4.568  -1.038   3.140 161.214

Coefficients:
               Estimate Std. Error  t value Pr(>|t|)
(Intercept)  -2.917e+01  2.654e-01 -109.926  < 2e-16 ***
width         8.695e-03  7.785e-04   11.169  < 2e-16 ***
height       -1.595e-02  1.496e-03  -10.660  < 2e-16 ***
bitrate       1.736e-06  7.577e-08   22.910  < 2e-16 ***
framerate     9.484e-02  7.423e-03   12.777  < 2e-16 ***
i_size       -1.591e-07  1.691e-08   -9.414  < 2e-16 ***
size          6.649e-09  1.396e-09    4.763 1.91e-06 ***
o_codech264   1.202e+01  1.288e-01   93.291  < 2e-16 ***
o_codecmpeg4  2.807e+00  1.185e-01   23.694  < 2e-16 ***
o_codecvp8    8.630e+00  1.185e-01   72.823  < 2e-16 ***
o_bitrate     1.411e-06  2.394e-08   58.963  < 2e-16 ***
o_framerate   2.492e-01  6.279e-03   39.690  < 2e-16 ***
o_width       7.059e-03  6.590e-04   10.712  < 2e-16 ***
o_height      4.608e-03  1.267e-03    3.636 0.000277 ***
umem          7.054e-05  5.276e-07  133.687  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.522 on 51573 degrees of freedom
Multiple R-squared:  0.6514,  Adjusted R-squared:  0.6513
F-statistic:  6884 on 14 and 51573 DF,  p-value: < 2.2e-16
```

Disegnando alcuni grafici si vede come si è ancora molto lontani dall'ottimo.
Il passo successivo è quello di applicare delle trasformazioni alle variabili, sia alla variabile risposta che alle variabile esplicative.
Per iniziare si applica una trasformazione logaritmica alla variabile risposta *utime*.

```
> fm3 <- update(fm2, log(utime)~., data=train)
```

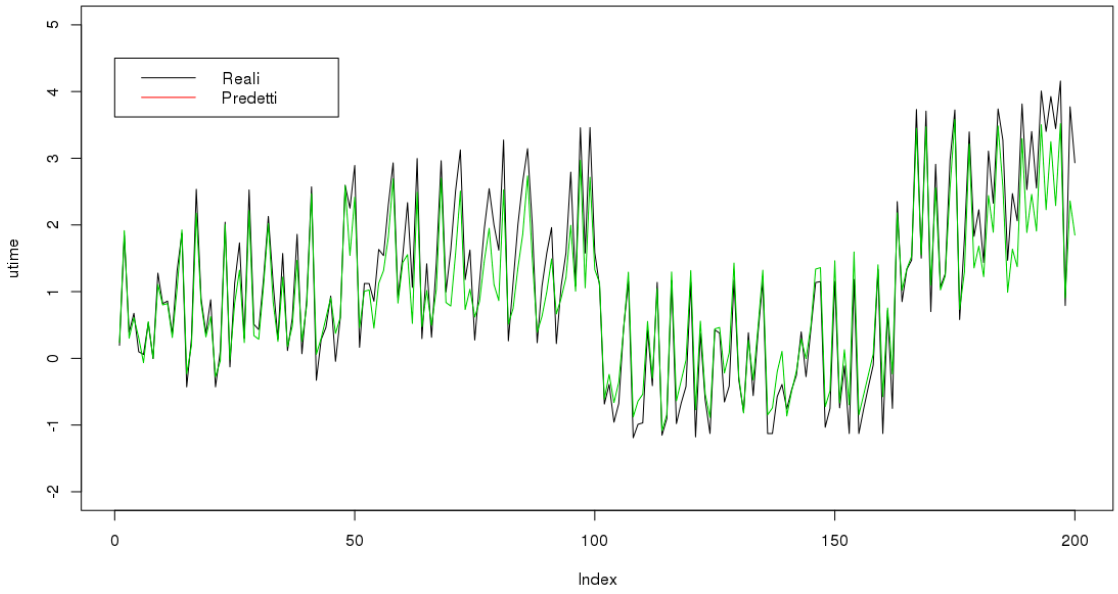
Grazie a questa trasformazione si ottiene un incremento della variabilità spiegata dal modello fino per arrivare a un valore del 87%. Il miglioramento è visibile anche nei grafici sottostanti.



Si osserva tuttavia un curioso effetto: una strana “massa” nella parte inferiore del grafico *Residuals vs Fitted*, all’aumentare dei valori *fitted*. Una massa presente nella parte inferiore ma non in quella superiore.
Pare che questa anomalia si presenti da subito e si va via via più evidente all’aumentare della dimensione del dataset.
Tuttavia dopo varie analisi non si è riuscito a capire la natura di tali dati.

E’ possibile migliorare ancora il modello? Sì, tramite la trasformazione delle variabili esplicative. Dopo varie prove si è arrivati a trasformare la variabile esplicativa *i_size* tramite la funzione inversa.

```
> fm3 <- update(fm3, .~. -i_size + I(1/i_size), data=train)
```



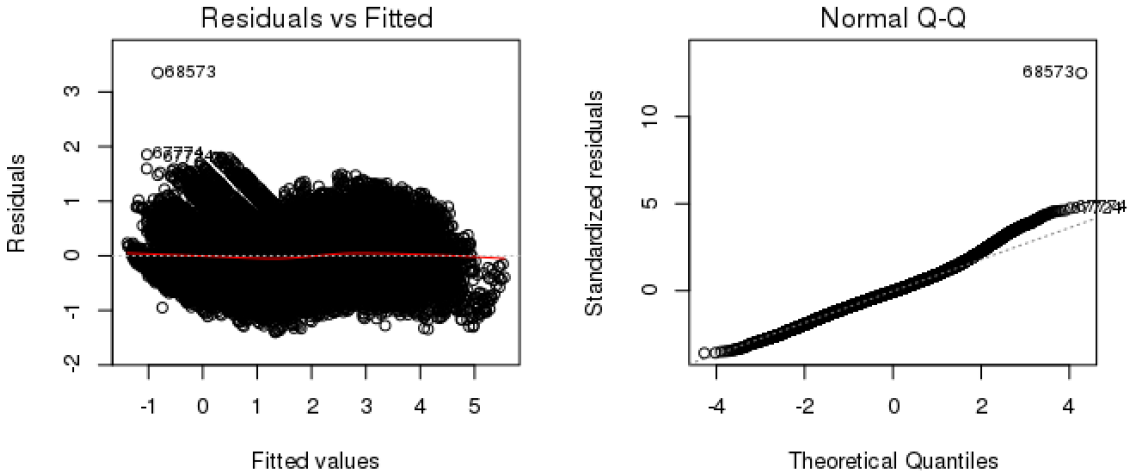
Regressione polinomiale

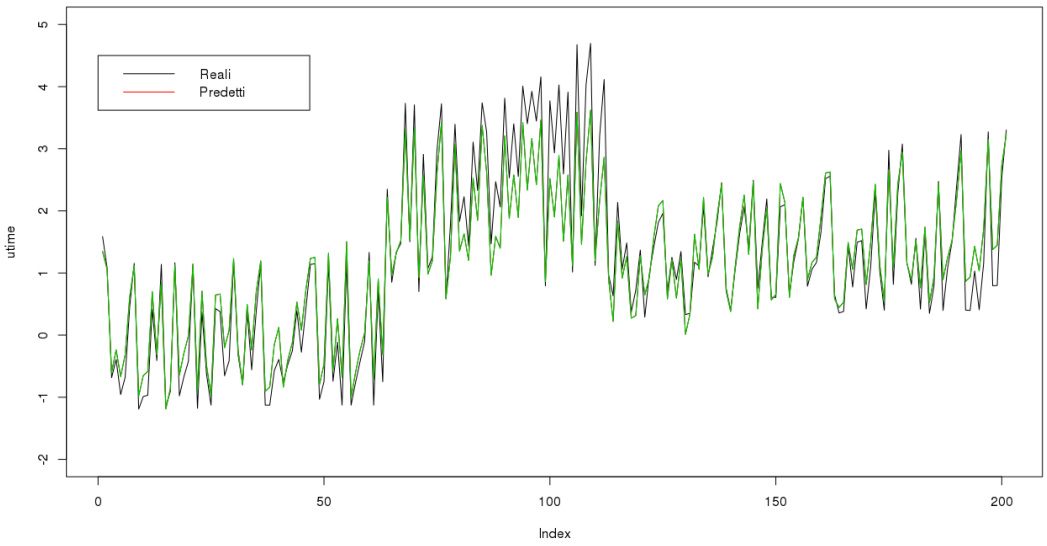
Studiando i grafici delle variabili è stato scelto il 4 grado per i polinomi. Sono state poi effettuate delle prove con diversi gradi per verificare la validità di tale dell'ipotesi.

```
Call:
lm(formula = log(utime) ~ framerate + o_codec + I(1/i_size) +
    poly(width, 4) + poly(bitrate, 4) + poly(framerate, 2) +
    poly(o_bitrate, 2) + poly(o_width, 2) + poly(o_framerate,
    2) + poly(o_height, 2) + poly(umem, 2), data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4008 -0.2490 -0.0207  0.2305  3.3503

Coefficients: (1 not defined because of singularities)
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            3.478e-01  9.968e-03  34.890  < 2e-16 ***
o_codech264            1.846e+00  5.311e-03 347.598  < 2e-16 ***
o_codecmpeg4           8.590e-01  4.864e-03 176.592  < 2e-16 ***
o_codecvp8             1.372e+00  4.865e-03 282.086  < 2e-16 ***
I(1/i_size)           -4.126e+03  3.534e+02 -11.677  < 2e-16 ***
poly(width, 4)1        6.780e+00  9.250e-01   7.330 2.34e-13 ***
poly(width, 4)2        1.158e+01  6.521e-01  17.758  < 2e-16 ***
poly(width, 4)3        8.747e+00  5.673e-01  15.419  < 2e-16 ***
poly(width, 4)4       -9.486e+00  4.359e-01 -21.759  < 2e-16 ***
poly(bitrate, 4)1      6.283e+01  8.291e-01  75.782  < 2e-16 ***
poly(bitrate, 4)2     -3.284e+01  5.923e-01 -55.439  < 2e-16 ***
poly(bitrate, 4)3      1.456e+01  4.739e-01  30.720  < 2e-16 ***
poly(bitrate, 4)4     -6.837e+00  4.311e-01 -15.859  < 2e-16 ***
poly(framerate, 2)1           NA         NA      NA       NA
poly(framerate, 2)2    7.248e+00  4.211e-01  17.215  < 2e-16 ***
poly(o_bitrate, 2)1    2.378e+01  3.910e-01  60.809  < 2e-16 ***
poly(o_bitrate, 2)2   -1.068e+01  3.910e-01 -27.325  < 2e-16 ***
poly(o_width, 2)1      1.175e+02  1.133e+01  10.364  < 2e-16 ***
poly(o_width, 2)2      4.351e+01  9.016e+00   4.826 1.40e-06 ***
```

```
poly(o_framerate, 2)1  3.219e+01  3.910e-01  82.317  < 2e-16 ***
poly(o_framerate, 2)2 -3.794e+00  3.910e-01  -9.704  < 2e-16 ***
poly(o_height, 2)1     4.792e+01  1.094e+01   4.379 1.20e-05 ***
poly(o_height, 2)2    -7.347e+01  9.479e+00  -7.751 9.29e-15 ***
poly(umem, 2)1         1.688e+01  4.910e-01  34.371  < 2e-16 ***
poly(umem, 2)2         5.827e+00  4.140e-01  14.076  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.391 on 51563 degrees of freedom
Multiple R-squared:  0.8919,  Adjusted R-squared:  0.8919
F-statistic: 1.773e+04 on 24 and 51563 DF,  p-value: < 2.2e-16
```

Con queste trasformazioni si è arrivati a sfiorare una variabilità spiegata del 90%. La parte inferiore del grafico QQ Plot è migliorata ma è leggermente peggiorata la parte superiore.



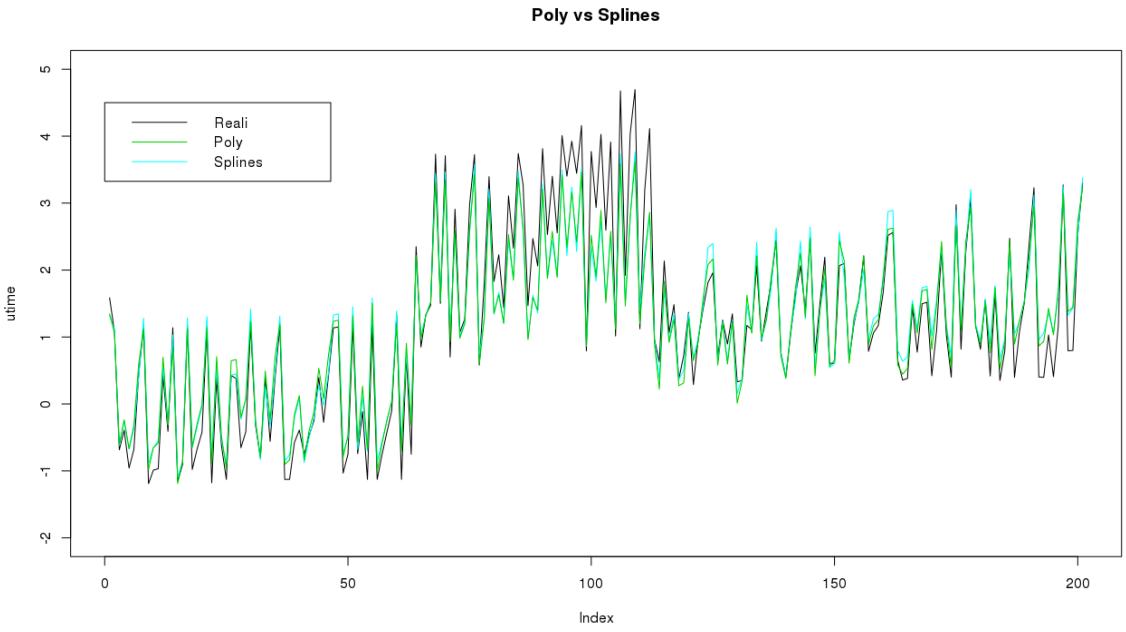


Spline

Si decide ora di utilizzare le S*pline* per le variabili esplicative *with* e *bitrate*.

```
> fmbs <- lm(log(utime) ~ bs(width, 3) + bs(bitrate, 3) + I(1/i_size) + o_codec
+ o_bitrate + o_framerate + o_width + o_height + umem, data = train)
> anova(fmpol, fmbs)
Analysis of Variance Table

Model 1: log(utime) ~ o_codec + I(1/i_size) + poly(width, 4) + poly(bitrate,
    4) + poly(framerate, 2) + poly(o_bitrate, 2) + poly(o_width,
    2) + poly(o_framerate, 2) + poly(o_height, 2) + poly(umem,
    2)
Model 2: log(utime) ~ bs(width, 3) + bs(bitrate, 3) + framerate + i_size +
    size + o_codec + o_bitrate + o_framerate + o_width + o_height +
    umem
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1  51563 7882.1
2  51570 8761.8 -7   -879.72 822.14 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Il test *ANOVA* conferma qualche miglioramento.



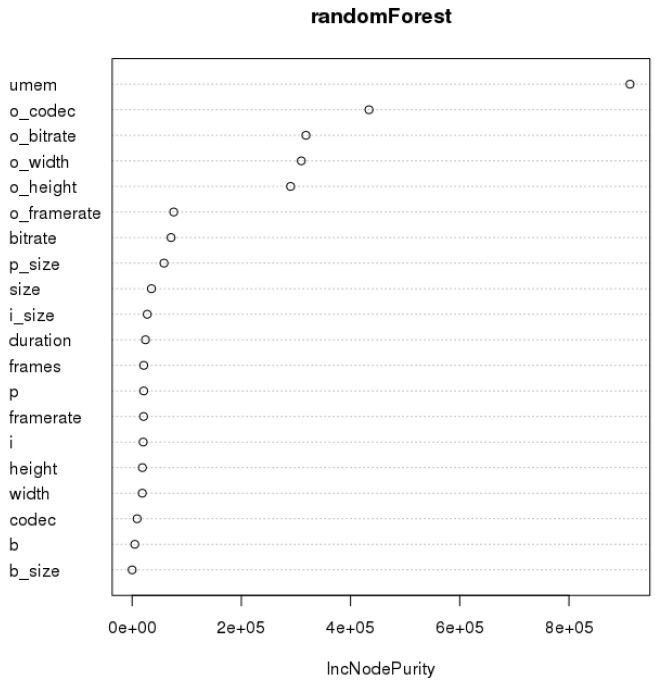
RandomForest

Utilizzando le *randomForest* si è dovuto ridurre la dimensione del dataset a causa di una capacità limita di memoria della macchina su cui si stava operando.
Il numero di *ntree* di default è '500' mentre qui è stato alzato a 1000. Questo perché dopo varie prove empiriche questo numero si è rilevato il miglior compromesso (un valore minore comportava una minore variabilità spiegata, mentre un valore maggiore richiedeva un tempo computazionale maggiore). Ragionamento inverso per il valore di nodesize, dove invece un valore maggiore implica un minore tempo di computazione (alberi più piccoli).

```
> fmforest <- randomForest(log(utime) ~., data=train[1:10000,],
na.action=na.omit, ntree=1000, nodesize=5)
> print(fmforest)

Call:
 randomForest(formula = utime ~ ., data = train[1:10000, ], ntree = 1000,
nodesize = 5, na.action = na.omit)
               Type of random forest: regression
                     Number of trees: 1000
No. of variables tried at each split: 6

          Mean of squared residuals: 13.25609
                    % Var explained: 95.16
```

Nelle prove effettuate, con le randomForests si raggiunge una variabilità spiegata del 95.5%.

Un utile strumento grafico, il *varImpPlot*, viene fornito con la libreria randomForest e mostra le variabili in ordine descrivente di importanza (in alto vuol dire più importante).



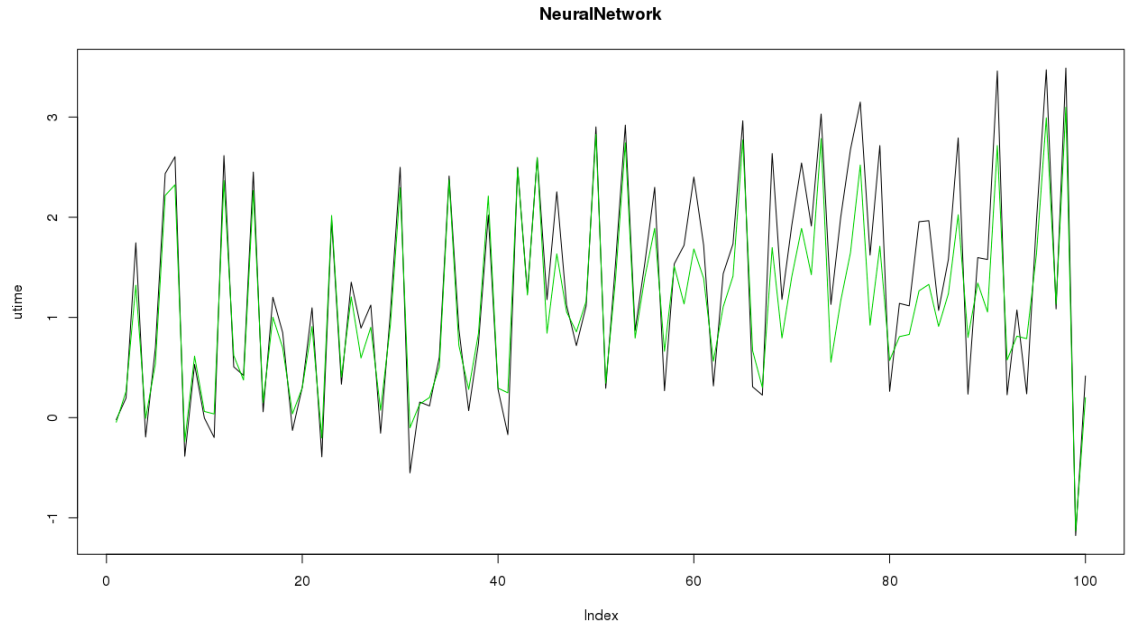
Da qui si può osservare come la variabile *umem* risulti la più significativa.
Inoltre le variabili che rappresentano caratteristiche di output del video risultano più significative di quelle di input.

Reti Neurali

Per l'ultimo modello verranno usate le *Reti Neurali*.
I parametri sono stati ricavati empiricamente; in particolare:

- *lineout* indica che le unità (o neuroni) applicano una funzione lineare dell'input.
- *size*: una regola empirica stabilisce che il numero di unità nel layer intermedio stia fra il numero di unità di input e quello di output.
- skip: ignora le confessioni al *hidden-layer.* E' stata scelta questa opzione dopo aver valutato più combinazioni; con questo parametro disabilitato (come è di default) non si riusciva a ottenere una buon modello predittivo.

```
> fmnet <- nnet(log(utime) ~ width + bitrate + framerate + o_codec + o_bitrate +
o_framerate + o_width + o_height + umem + i_size, data=train, size=6,
decay=1e-3, maxit=1200, linout=TRUE, MaxNWts = 5000, skip=TRUE)
```



Conclusioni

Sul modello

Secondo le prove effettuate la miglior tecnica di regressione per il dataset in esame sembrano essere le randomForest, con una variabilità spiegata del 95,5%.
Tuttavia questa tecnica richiede un tempo computazionale (oltre che risorse di spazio) non indifferente; le tecniche con splines o di regressione polinomiale, invece, seppur con una varianza spiegata minore (89% circa), potrebbero essere preferite in quanto il loro costo computazionale è decisamente inferiore.

Nella tabella sottostante sono riportati i tempi di alcune prove per l'esecuzione delle tecniche effettuate (in secondi).

	Test 1	Test 2	Test 3	Valore medio (s)
Modello lineare (base)	0,140	0,132	0,136	0,136
Modello lineare (con trasformazioni)	0,092	0,108	0,136	0,122
Modello polinomiale	0,276	0,392	0,340	0,336
Spline	0,128	0,132	0,132	0,130
randomForest	316,8	328,8	340,7	328,7
Neural Network	1,196	0,796	0,760	0,877

Le rilevazioni sono state effettuate utilizzando il comando *system.time* nell'ambiente R. Il parametro gcFirst serve per chiamare il garbage collector prima che l'espressione venga valutata.

```
> system.time(EXPR, gcFirst=TRUE)
```

Sui dati

Dopo aver osservato le varie analisi con diversi modelli di regressione sembra certo che le caratteristiche di output del video (o_codec, o_width,…) influiscano maggiormente sul tempo di codifica/decodifica. Questo è abbastanza logico.
Non solo: le caratteristiche di input del video sembrano influire poco o niente sulla decodifica finale, perfino la dimensione del video (variabile esplicativa *size*).