Data Mining - Prova d'esame del 30.1.2020

Laboratorio: punteggi e soluzione

Punteggi

files	RMSE	Percentuale	Punti
2575.txt	0.0617299	0%	0.0
790430.txt	0.0619043	0%	0.0
790430bis.txt	0.0619043	0%	0.0
804450.txt	0.0254984	77.5%	7.0
805913.txt	0.0351962	56.8%	5.1
807557.txt	0.0246842	79.3%	7.1
807761.txt	0.0300018	67.9%	6.1
807782.txt	0.0475421	30.3%	2.7
807842.txt	0.0165588	96.7%	8.7
808162_bis.txt	0.0577457	8.5%	0.8
808162.txt	0.0340824	59.1%	5.3
808693.txt	0.0645217	0%	0.0
836695.txt	0.0196666	90%	8.1
836897.txt	0.0138414	100%	9.0
849037.txt	0.0349236	57.3%	5.2

Note

Matricola	Note
807498	Nessun file consegnato
808162	File .html non riproduce le previsioni
808644	Solo file .Rmd
849037	File .txt mancante
776970	Solo file .Rmd
836897	File .html non riproduce la previsione

Soluzione

Il predittore X è stato generato da una distribuzione Uniforme$(0, 1)$. La variabile risposta è stata generata nel modo seguente: $Y = f(X) + \varepsilon$ dove

$$f(X) = (X)^2(\sin(5\pi X))^6$$

e $\varepsilon \sim N(0, \sigma^2)$ con $\sigma = 0.01$. Si noti la varianza dell'errore estremamente bassa!

Gli altri predittori Z, W e V sono funzioni di X. Date le n realizzazioni di X, i.e. $x_1, \ldots, x_n$, abbiamo

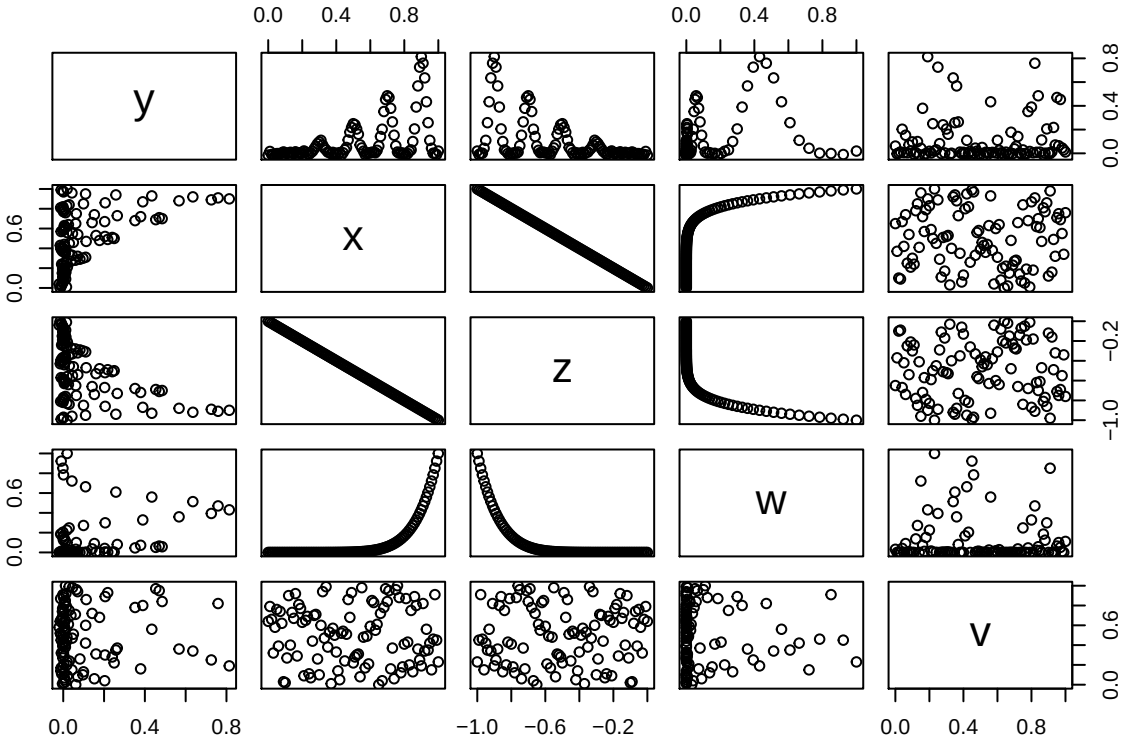
- $z_i = -x_i$

- $w_i = (x_i)^8$
- $v_i = x_{\pi(i)}$

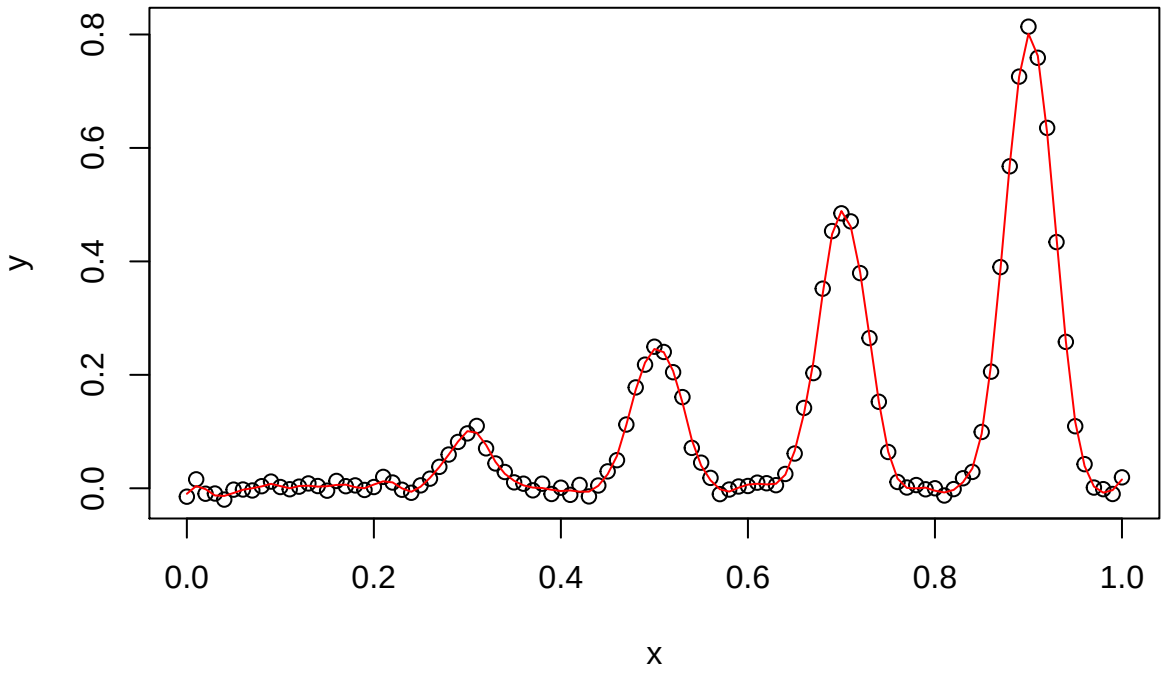
dove $\{\pi(1), \ldots, \pi(n)\}$ è una permutazione causale di $\{1, \ldots, n\}$.

Di conseguenza i predittori Z e W sono "copie" (trasformazioni) di X, e quindi associati alla risposta Y, mentre la dipendenza tra V e tutte le altre variabili (compresa la risposta) viene "distrutta" dalla permutazione casuale.

Queste considerazioni si potevano ricavare dall'analisi della matrice dei diagrammi di dispersione



Considerando solo il predittore X (o una sua trasformazione), notiamo una forte dipendenza (non lineare) tra Y e X, che possiamo ad esempio modellare con le *smoothing spline*:



quindi il RMSE sul test set è molto buono:

```
library(splines)
fit = smooth.spline(tr$x, tr$y)
yhat = predict(fit, x = te$x)$y
( RMSE = sqrt( mean( (yhat - y.te)^2) ) )
```

```
[1] 0.01167488
```

Anche un "semplice" KNN con $K = 2$ si comporta egregiamente:

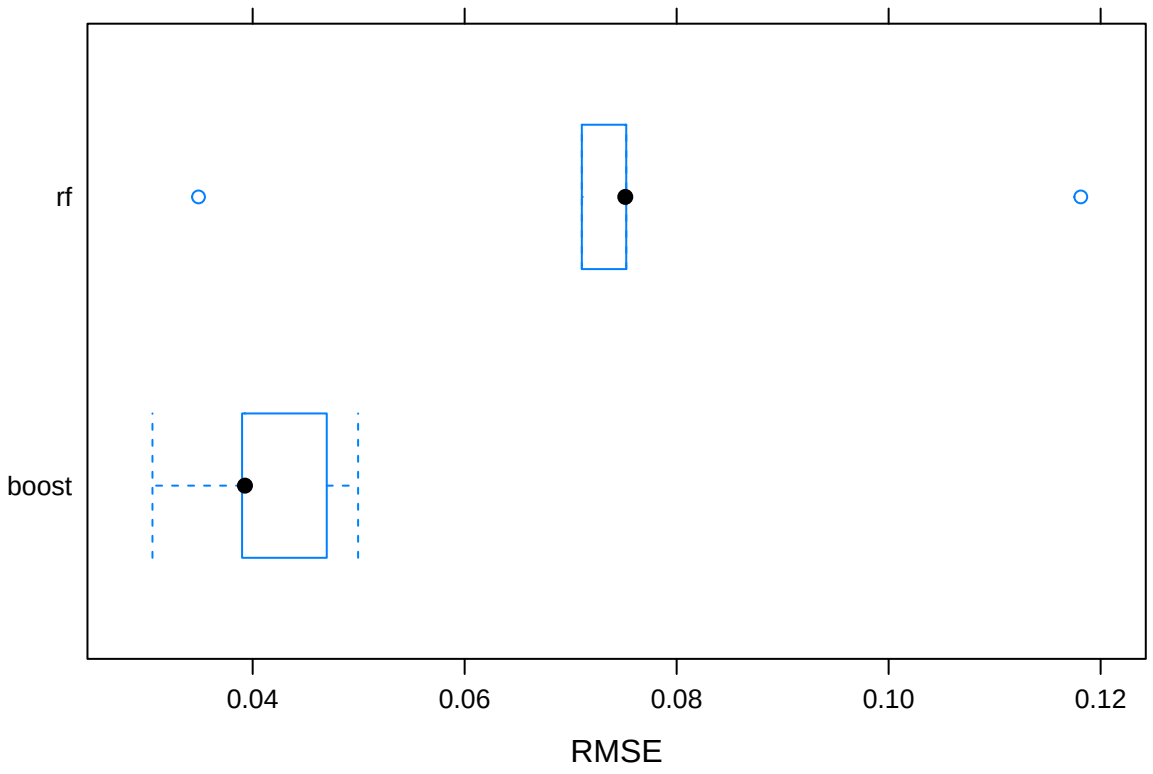
```
library(kknn)
K = train.kknn(y ~ x, data=tr)$best.parameters$k
fit = kknn(y ~ x, tr, te, , k = K)
yhat = fit$fitted.values
( RMSE = sqrt( mean( (yhat - y.te)^2) ) )
```

```
[1] 0.01508941
```

In assenza di queste considerazioni, si poteva adottare l'approccio "forza bruta" considerando diversi modelli (e.g. KNN, Random Forest, Boosting) che spiegano la risposta in funzione di tutti i predittori (includendo quindi anche il "rumore" V), i.e.

$$Y = f(X, Z, W, V) + \varepsilon$$

scegliando poi il "migliore" con il metodo della convalida incrociata.




```
$rf
[1] 0.03544695

$boost
[1] 0.02867743
```