Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Magistrale in Informatica

Progetto Machine Learning: Kepler Exoplanet

Relazione di:
Stefano Talamona 822452
Matteo Gagianesi 807316

Febbraio 2021

Indice

1 Introduzione

Questo elaborato ha come obiettivo quello di presentare il progetto svolto per il corso di Machine Learning. In questa relazione verranno presentati in un primo momento i dati sui quali il progetto è strutturato, si passerà poi ad effettuare una breve analisi esplorativa su di essi per poi passare all'applicazione di alcuni modelli di classificazione.

2 Dataset

2.1 Analisi preliminare

Il dataset utilizzato per questo progetto è fornito dalla NASA (DOI 10.26133/NEA4) ed è accessibile sul sito Kaggle presso il seguente link: `https://www.kaggle.com/nasa/kepler-exoplanet-search-results`. Questo è composto da 9564 istanze per 50 variabili che rappresentano i risultati dei rilevamenti effettuati dal telescopio orbitale Keplero, la cui missione è l'esplorazione del cosmo alla ricerca di esopianeti, ovvero pianeti che orbitano attorno ad una stella che non sia il nostro Sole. Alcune variabili sono state scartate a priori in quanto presentavano solo valori nulli. Altre sono state scartate sempre a priori per via del fatto che rappresentano codici identificativi utilizzati dagli scienziati della NASA e di altre compagnie di esplorazione spaziale per far riferimento a tali rilevazioni, senza quindi rappresentare alcuna caratteristica delle istanze in esame. Le informazioni relative ad ogni variabile presente nel dataset sono reperibili al seguente link: `https://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html`.
In totale le variabili che abbiamo utilizzato sono 29, qui di seguito descritte e raggruppate in **continue** e **categoriche**:

- **Categoriche**
  - **koi_disposition**: variabile target del nostro problema di classificazione. Essa indica se il risultato di un rilevamento è già stato confermato (come effettivo esopianeta o come falso positivo) o meno;
  - **koi_fpflag_nt**: flag che indica se la curvatura della luce dell'oggetto in esame è consistente o meno con quella di un pianeta in transito;
  - **koi_fpflag_ss**: flag che indica se per l'oggetto in questione è stato rilevato un evento secondario significativo (evento che mostra caratteristiche significativamente differenti da quelle riscontrate nella prima rilevazione);
  - **koi_fpflag_co**: flag che indica se la fonte della rilevazione proviene da una stella vicina;
  - **koi_fpflag_ec**: flag che indica se l'oggetto in questione mostra stessa epoca e periodo di un altro oggetto in esame;
  - **koi_tce_plnt_num**: rappresenta il numero di thrashold crossing events registrati per tale istanza, ovvero il numero di eventi che oltrepassano una certa soglia di misurazione. Questa stima è effettuata da un software apposito (Robovetter);

- **Continue**
  - **koi_period**: intervallo temporale tra due transiti planetari consecutivi;
  - **koi_period_err2**: errore negativo stimato per la misura dell'attributo *koi_period*;
  - **koi_time0bk**: tempo che corrisponde al centro del primo rilevamento del transito nel Barycentric Julian Day meno una costante fissata a 2,454,833.0 giorni;
  - **koi_time0bk_err2**: errore negativo stimato per la misura dell'attributo *koi_time0bk*;
  - **koi_impact**: distanza proiettata nel cielo tra il centro del disco stellare ed il centro del disco planetario;
  - **koi_impact_err1**: errore positivo stimato per la misura dell'attributo *koi_impact*;
  - **koi_duration**: durata del transito dell'oggetto osservato;
  - **koi_duration_err2**: errore negativo stimato per la misura dell'attributo *koi_duration*;
  - **koi_depth**: frazione di perdita di flusso stellare osservata durante il transito dell'oggetto;
  - **koi_depth_err1**: errore positivo stimato per la misura dell'attributo *koi_depth*;
  - **koi_prad_err2**: errore negativo stimato per la misura dell'attributo *koi_prad*;
  - **koi_teq**: misura approssimata della temperatura del pianeta;
  - **koi_insol_err2**: errore negativo stimato per la misura dell'attributo *koi_insol*;
  - **koi_model_snr**: profondità del transito normalizzata dall'incertezza media nel flusso durante i transiti;
  - **koi_steff**: temperatura fotosferica della stella;
  - **koi_steff_err2**: errore negativo stimato per la misura dell'attributo *koi_steff*;
  - **koi_slogg**: logaritmo in base 10 dell'accelerazione dovuta alla gravità sulla superficie della stella;
  - **koi_slogg_err1**: errore positivo stimato per la misura dell'attributo *koi_slogg*;
  - **koi_slogg_err2**: errore negativo stimato per la misura dell'attributo *koi_slogg*;
  - **koi_srad_err1**: errore positivo stimato per la misura dell'attributo *koi_srad*;
  - **ra**: ascensione retta (misura simile alla longitudine utilizzata sul nostro pianeta)
  - **dec**: declinazione, misura complementare dell'ascensione retta;
  - **koi_kepmag**: magnitudine del telescopio Keplero durante l'osservazione.

2.2 Modifiche ed assunzioni

Il dataset iniziale mostrava diversi valori mancanti sparsi nelle molteplici righe e colonne che lo compongono. Abbiamo deciso quindi di gestire questo problema rimuovendo le

istanze che presentavano valori mancanti, passando così da 9564 istanze a 8744.
Al fine di operare una classificazione significativa, abbiamo deciso di eliminare anche tutte le istanze con label (attributo *koi_disposition*) "CANDIDATE", ovvero tutti quegli oggetti il cui transito è stato rilevato dal telescopio Keplero ma che non hanno ancora passato tutti i test necessari ad affermare che si tratti effettivamente di un esopianeta piuttosto che di un falso positivo. Ciò in quanto la label "CANDIDATE" non rappresenta una vera e propria classe a cui poter ricondurre le istanze, bensì uno stato di incertezza che rende gli oggetti in questione non attribuibili ad alcuna classe con sufficiente accuratezza. Questo ha ridotto ulteriormente la dimensione del dataset, passando a 6630 istanze.

2.3 Distribuzioni

Al fine di ottenere delle stime in grado di darci informazioni utili sulla reale composizione delle variabili del nostro problema, abbiamo effettuato una analisi delle statistiche e delle distribuzioni. Come per la sezione precedente mostreremo solo i risultati ottenuti per le 29 variabili che utilizzeremo più avanti per i nostri modelli di classificazione. Stavolta quelle **continue** verranno ulteriormente raggruppate in base alla distribuzione a cui più si avvicinano tra *Normale, Lognormale* e *Gamma*, e per ognuna di queste verranno mostrati i grafici sviluppati solo per una delle variabili che vi appartengono.

2.3.1 Categoriche

- **koi_disposition:**
  Come si può notare in Figura 1 il numero di istanze con label "FALSE POSITIVE" è molto superiore al numero di istanze con label "CONFIRMED", infatti le percentuali delle due sono rispettivamente del 65.73% e 34.27%

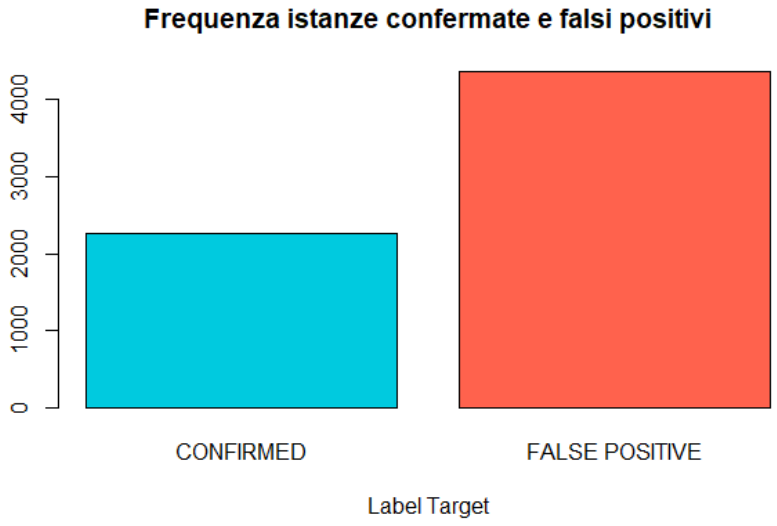

Figura 1: Distribuzione koi_disposition

- **koi_fpflag_np, koi_fpflag_ss, koi_fpflag_co, koi_fpflag_ec:**
  Come visibile in Figura 2, la distribuzione delle quattro variabili flag è fortemente decentrata verso il valore zero, questo in quanto gli eventi che fanno "scattare" tali flag sono eventi particolari e poco comuni.

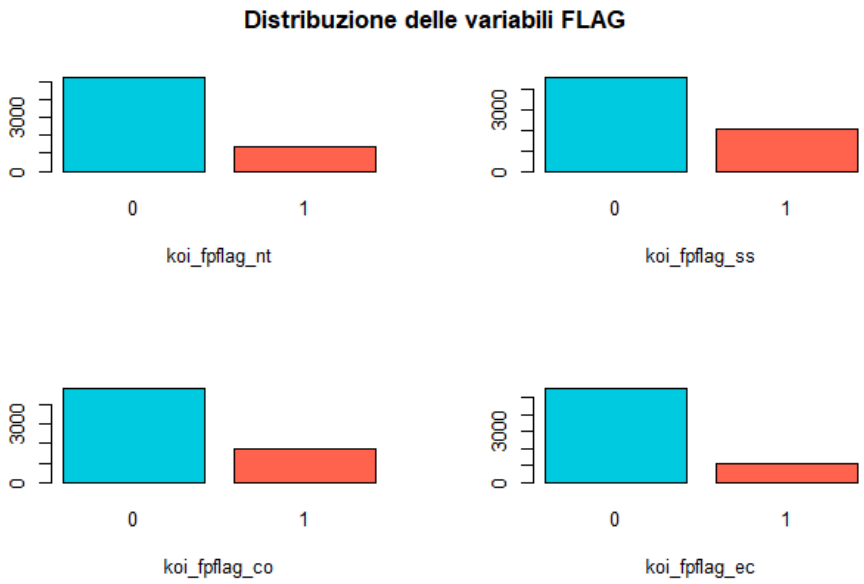

Figura 2: Distribuzione delle quattro variabili flag all'interno del dataset

- **koi_tce_plnt_num:**
  Un discorso analogo a quello delle flag vale anche per la variabile *koi_tce_plnt_num*. È infatti molto raro che per una data istanza vi siano threshold crossing events multipli, come confermato dal grafico in Figura 3. Si può notare anche come il valore varia da un minimo di 1 a un massimo di 8. Il valore minimo pari a 1 indica proprio il fatto che l'oggetto in questione è stato preso in considerazione a fronte di un evento significativo, altrimenti sarebbe stato scartato in partenza e non figurerebbe nel dataset originale.

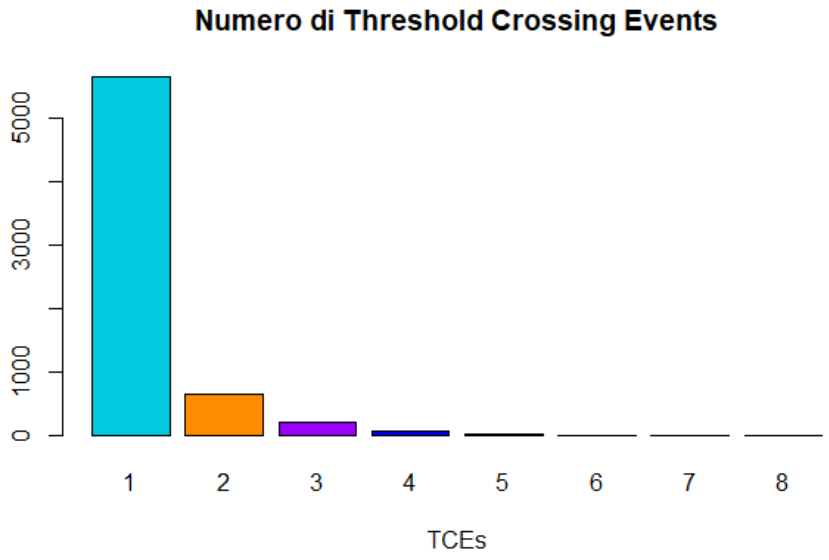

Figura 3: Distribuzione dei valori di koi_tce_plnt_num

2.3.2 Continue

- **Distribuzione Normale**: le variabili continue che presentano una distribuzione riconducibile a una normale sono mostrate in Figura 4 assieme alle loro statistiche principali.

Variabile	Min	Max	Mean	Median	SD
koi_period_err2	-0.157	~0	-0.0017	~0	0.0072
koi_time0bk_err2	-0.569	~0	-0.008	-0.003	0.0194
koi_impact	0	100.806	0.7753	0.58	3.46303
koi_impact_err1	0	85.54	2.0877	0.18	9.6793
koi_duration_err2	-20.2	0	-0.291	-0.107	0.6218
koi_steff_err2	-1762	0	-163.9437	-161	76.0778
koi_slogg	0.047	5.283	4.3037	4.4365	0.444
koi_slogg_err1	0	1.472	0.122494	0.072	0.1342
koi_slogg_err2	-1.007	0	-0.1395	-0.128	0.0803
ra	279.8527	301.7208	292.166	292.3473	4.7515
koi_kepmag	6.966	19.065	14.2481	14.4975	1.3583

Figura 4: Tabella delle statistiche inerenti alle variabili con distribuzione riconducibile a quella di una normale

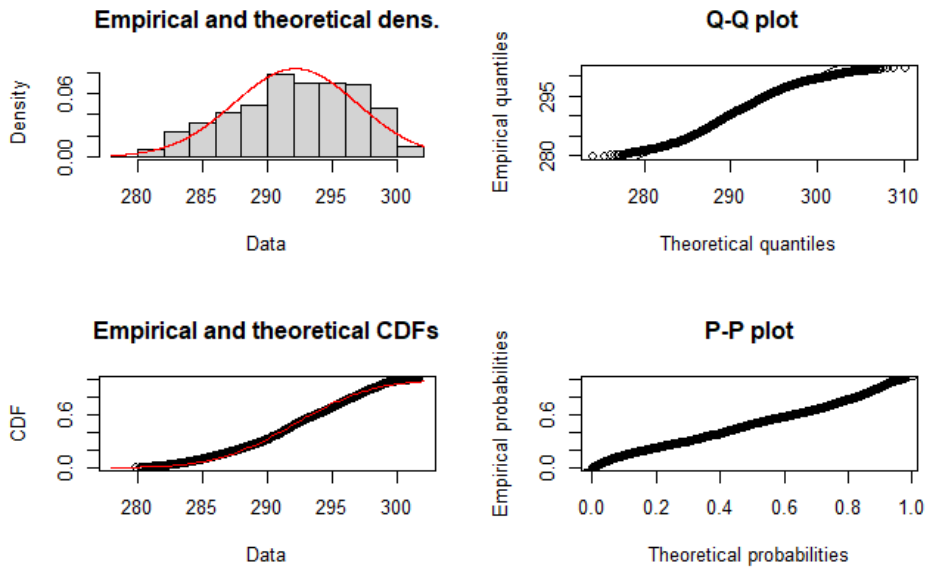

Figura 5: Four panels per la variabile *ra*

- **Distribuzione Lognormale**: le variabili continue che presentano una distribuzione riconducibile a una lognormale sono mostrate in tabella 6 assieme alle loro statistiche principali.

Variabile	Min	Max	Mean	Median	SD
koi_period	0.2997	1071.233	52.7411	7.9674	116.1168
koi_time0bk	120.5159	1472.522	162.9626	136.4294	63.7901
koi_duration	0.167	138.54	5.7672	3.892	6.8256
dec	36.5774	52.336	43.7953	43.6572	3.5999

Figura 6: Tabella delle statistiche inerenti alle variabili con distribuzione riconducibile a quella di una lognormale

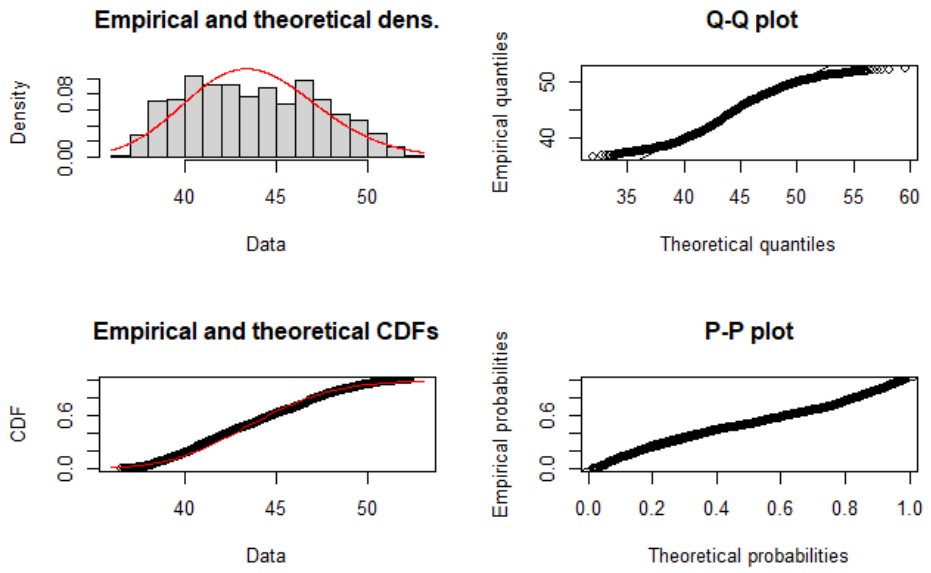

Figura 7: Four panels per la variabile *dec*

- **Distribuzione Gamma**: le variabili continue che presentano una distribuzione riconducibile a una gamma sono mostrate in tabella 8 assieme alle loro statistiche principali.

Variabile	Min	Max	Mean	Median	SD
koi_depth	4.5	1540000	30471.64	513.5	92700.3
koi_depth_err1	0	389000	145.7013	19.4	4823.796
koi_prad_err2	-77200	0	-43.6358	-0.37	1404.522
koi_teq	92	14667	1153.382	934	881.4868
koi_insol_err2	-5362422	0	-3923.682	-54.915	75414.97
koi_model_snr	1.6	9054.7	339.3137	32	911.6467
koi_steff	2661	15896	5717.106	5766.5	829.1846
koi_srad_err1	0	33.091	0.3612	0.248	0.9848

Figura 8: Tabella delle statistiche inerenti alle variabili con distribuzione riconducibile a quella di una gamma

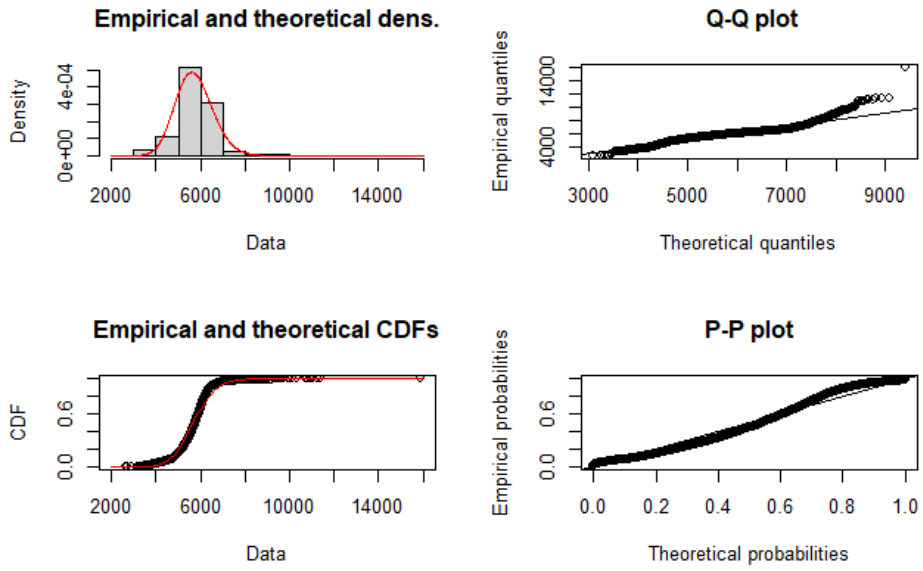

Figura 9: Four panels per la variabile *koi_steff*

3 Analisi esplorativa del dataset

Vi sono casi in cui all'interno del dataset sono presenti una o più feature fra di loro correlate, esprimenti perciò informazioni simili. È quindi utile svolgere un'attività di analisi esplorativa al fine di individuare questi elementi ridondanti ed eliminarli. In tal modo ne risulterà un dataset contenente solamente feature significative e potenzialmente più "leggero".

3.1 Matrice di correlazione

Una prima tecnica applicabile per identificare la correlazione tra le feature del nostro dataset è proprio la matrice di correlazione. Nel grafico in Figura 10 si può notare come la correlazione tra le varie coppie sia espressa da una gamma cromatica. Una forte correlazione viene espressa da una casella di intersezione blu mentre una forte correlazione negativa è evidenziata da una casella rossa.

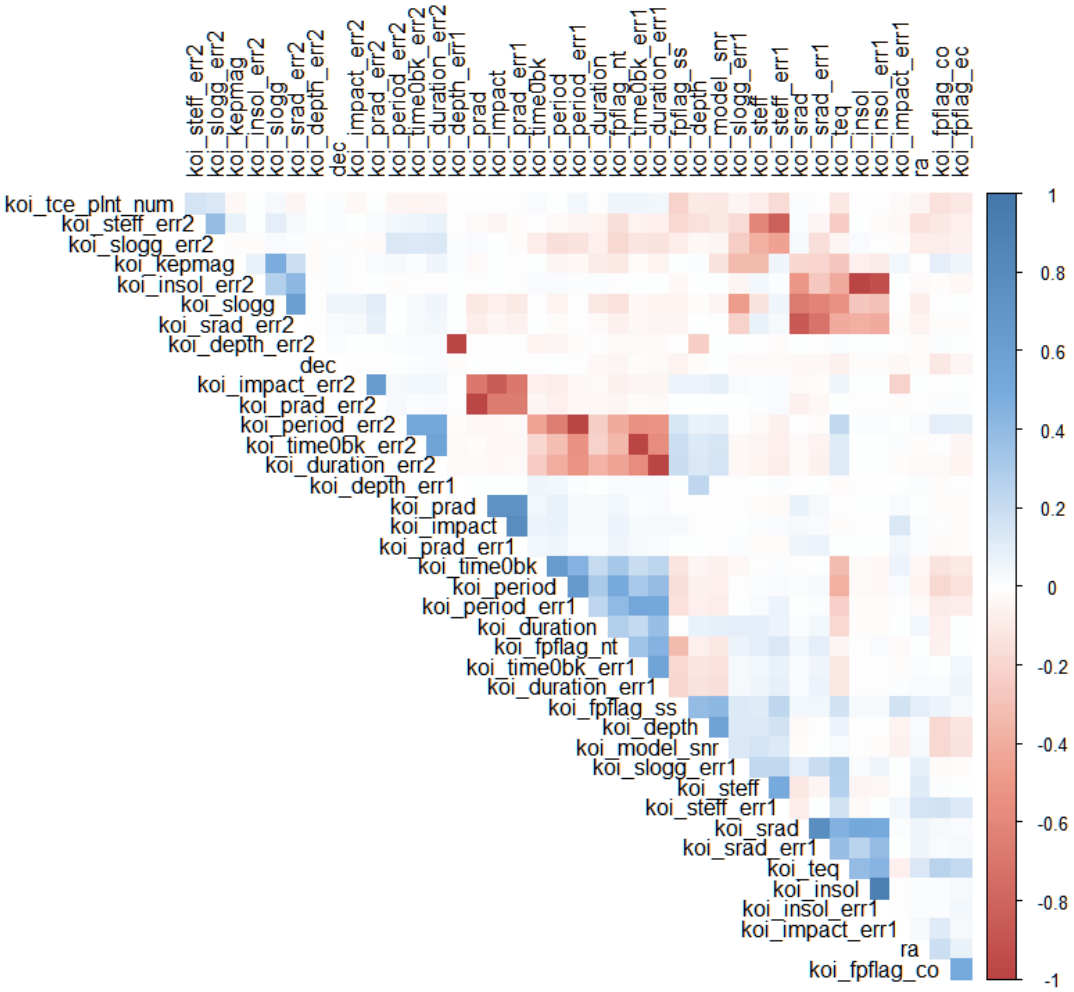

Figura 10: Matrice di correlazione

Analizzando più nel dettaglio questa funzione è possibile evidenziare quelli che sono gli elementi con indice di correlazione più alto e quindi più ridondanti. Si possono identificare quindi questi attributi mediante la funzione *findCorrelation* assegnando un filtro per l'indice di correlazione arbitrario (in questo caso 0.7). Di seguito sono riportati i suddetti attributi:

- koi_duration_err1
- koi_period_err1
- koi_time0bk_err1
- koi_srad
- koi_srad_err2
- koi_steff_err1
- koi_insol_err1
- koi_insol
- koi_impact_err2
- koi_prad
- koi_prad_err1
- koi_depth_err2

3.2 PCA

Un altro metodo per trovare quelle che sono le variabili con una maggiore similarità intrinseca, al fine di ridurre la dimensione del dataset, è la Principal Component Analysis (PCA). Questa tecnica è una trasformazione lineare che permette di ridurre il numero di dimensioni identificando le diverse direzioni all'interno dello spazio delle feature dove queste ultime variano. Una più accentuata varianza tra i dati comporterà anche una maggiore varianza in quella direzione.

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	5.545097e+00	1.386274e+01	13.86274
Dim.2	4.944133e+00	1.236033e+01	26.22308
Dim.3	3.929400e+00	9.823500e+00	36.04658
Dim.4	3.212731e+00	8.031828e+00	44.07840
Dim.5	2.288698e+00	5.721744e+00	49.80015
Dim.6	2.033907e+00	5.084767e+00	54.88491
Dim.7	1.856550e+00	4.641376e+00	59.52629
Dim.8	1.457222e+00	3.643054e+00	63.16934
Dim.9	1.403803e+00	3.509508e+00	66.67885
Dim.10	1.152480e+00	2.881199e+00	69.56005
Dim.11	1.059998e+00	2.649995e+00	72.21005
Dim.12	1.037994e+00	2.594986e+00	74.80503
Dim.13	9.774825e-01	2.443706e+00	77.24874
Dim.14	9.456527e-01	2.364132e+00	79.61287
Dim.15	8.354938e-01	2.088734e+00	81.70160
Dim.16	7.964074e-01	1.991018e+00	83.69262
Dim.17	7.407082e-01	1.851770e+00	85.54439
Dim.18	6.807499e-01	1.701875e+00	87.24627
Dim.19	6.309096e-01	1.577274e+00	88.82354
Dim.20	5.688989e-01	1.422247e+00	90.24579
Dim.21	5.330652e-01	1.332663e+00	91.57845
Dim.22	4.539342e-01	1.134836e+00	92.71329
Dim.23	4.453901e-01	1.113475e+00	93.82676
Dim.24	4.250104e-01	1.062526e+00	94.88929
Dim.25	3.406268e-01	8.515671e-01	95.74086
Dim.26	3.188935e-01	7.972338e-01	96.53809
Dim.27	2.878182e-01	7.195454e-01	97.25764
Dim.28	2.672827e-01	6.682068e-01	97.92584
Dim.29	2.438903e-01	6.097258e-01	98.53557
Dim.30	1.774472e-01	4.436180e-01	98.97919
Dim.31	1.377378e-01	3.443445e-01	99.32353
Dim.32	1.184358e-01	2.960896e-01	99.61962
Dim.33	8.812516e-02	2.203129e-01	99.83993
Dim.34	5.316247e-02	1.329062e-01	99.97284
Dim.35	6.682975e-03	1.670744e-02	99.98955
Dim.36	4.181246e-03	1.045311e-02	100.00000
Dim.37	1.372093e-30	3.430232e-30	100.00000
Dim.38	9.572003e-31	2.393001e-30	100.00000
Dim.39	6.259257e-31	1.564814e-30	100.00000
Dim.40	1.355710e-32	3.389274e-32	100.00000

Figura 11: Risultati della PCA

Dai risultati della PCA, osservabili in Figura 11, si nota come le prime 20 dimensioni spieghino il 90% della varianza totale del dataset. Questo è un chiaro indicatore di come molte variabili siano in realtà correlate tra di loro e quindi ridondanti per l'applicazione di modelli di classificazione.

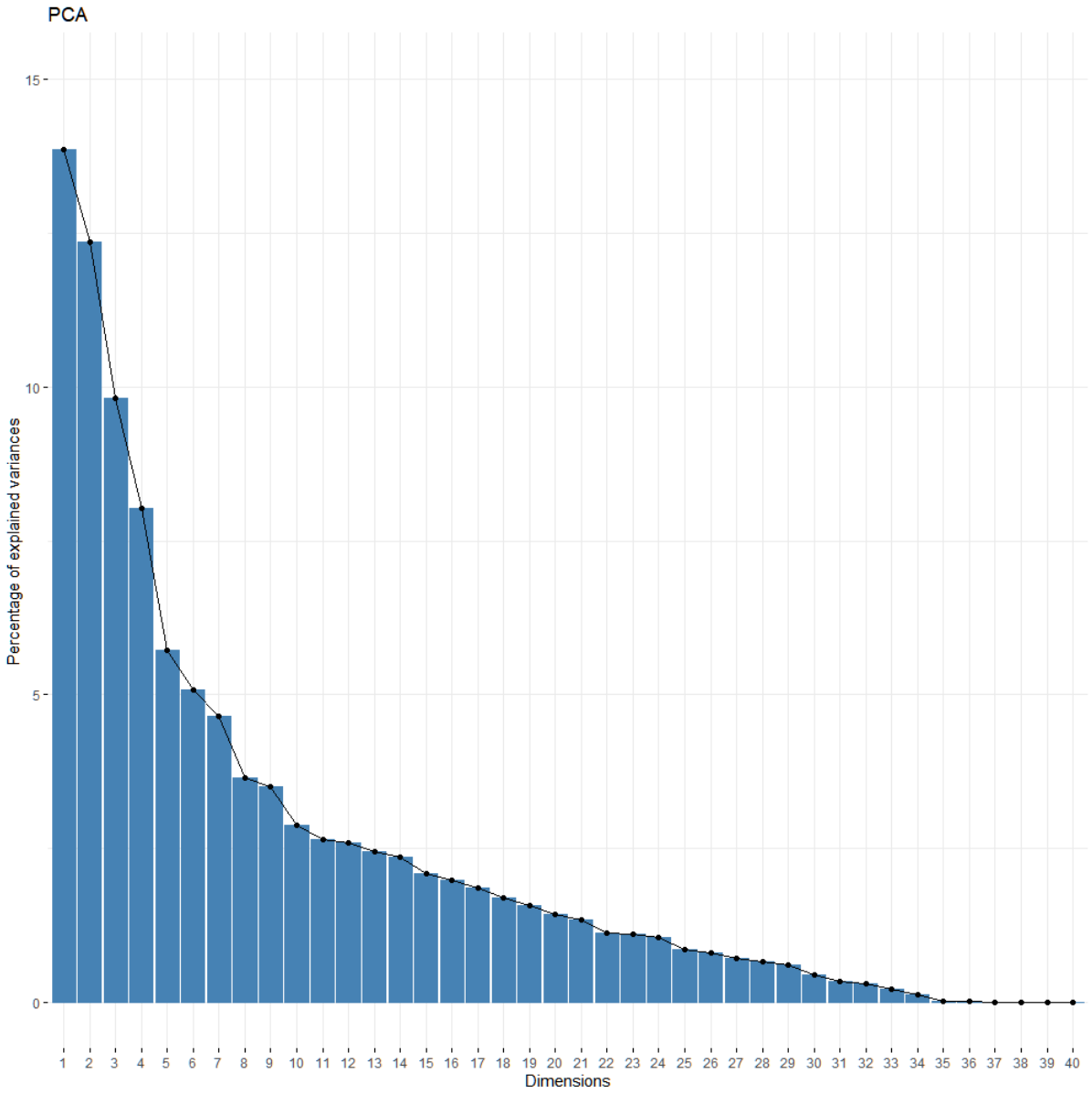

Figura 12: Varianza espressa per ogni dimensione

Nel grafico in Figura 12 è ancora più evidente come, oltre una certa dimensione, la varianza spiegata da quelle successive si riduca sempre di più.

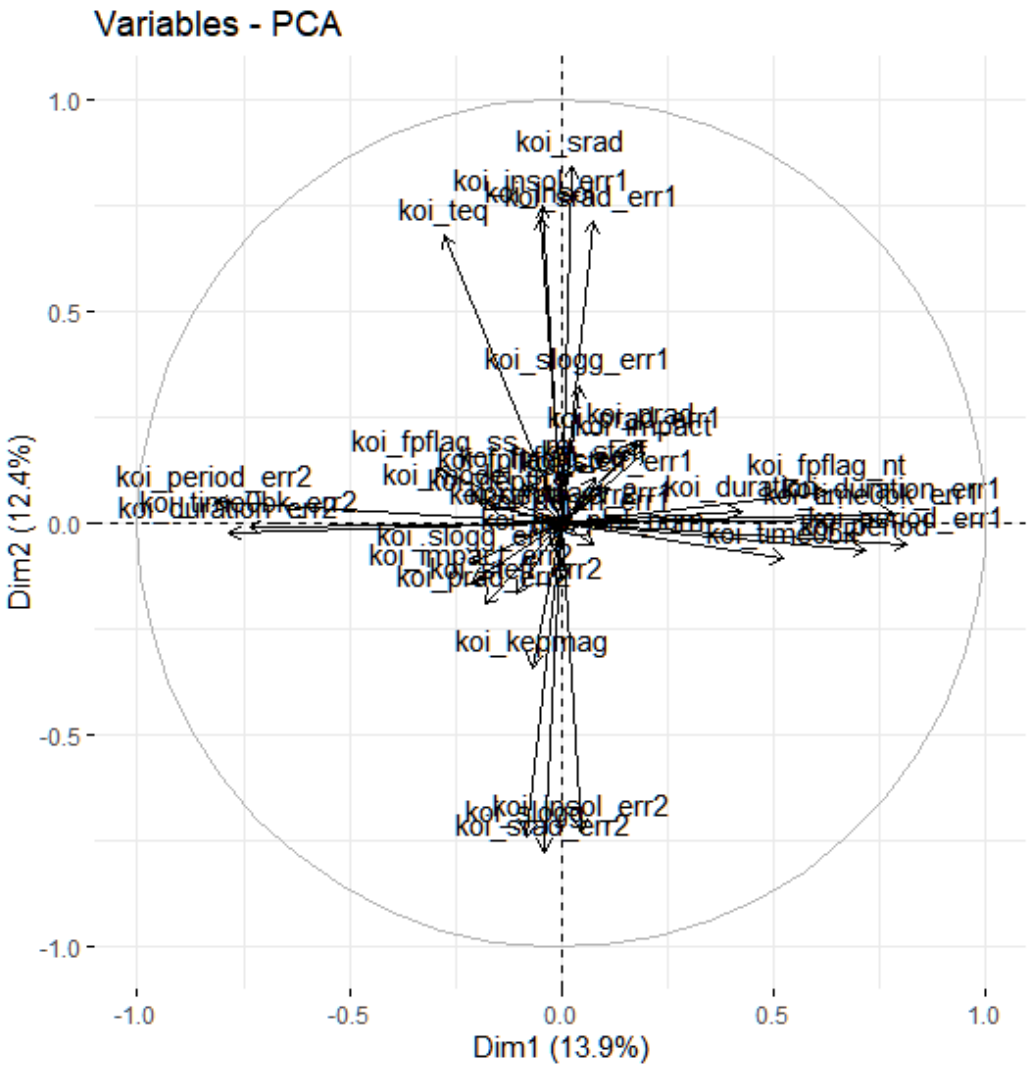

Figura 13: Grafico di correlazione della PCA

Da questo ultimo grafico (Figura 13) è ancora più evidente come alcune variabili siano correlate tra di loro. Si possono notare infatti elementi aventi direzione pressoché uguale, maggiore è questa vicinanza maggiore sarà la correlazione tra di essi. Viceversa, due variabili la cui direzione dovesse essere opposta rispetta all'origine, risulteranno correlate negativamente. Questo grafico permette anche di capire la qualità espressa dalle diverse feature; più esse risulteranno lontane dall'origine e maggiore sarà la qualità.

3.3 Conclusioni dell'analisi esplorativa

Al termine dell'analisi esplorativa del dataset sono state quindi eliminate le 12 variabili aventi indice di correlazione superiore a 0.7, in quanto avrebbero aumentato la complessità dei modelli senza, di fatto, fornire informazioni aggiuntive rispetto al resto del dataset. Il subset derivante da questa operazione risulta quindi composto da 28 variabili e verrà utilizzato in seguito per lo sviluppo dei modelli di classificazione.

4 Modelli

Data la presenza di un campo target (*koi_disposition*), la scelta dei modelli di classificazione che sono stati applicati è ricaduta su quelli di tipo supervisionato. In particolare i modelli utilizzati sono:

- **SVM**
- **Neural Network**
- **Nayve Bayes**
- **Decision Tree e Random Forest**

Per tutti i modelli utilizzati il dataset principale è stato suddiviso in trainset (70%) e testset (30%). Inoltre per ciascun modello è stata eseguita la 10-fold cross-validation in fase di analisi dei risultati.

4.1 SVM

La Support Vector Machine rappresenta un modello di apprendimento la cui logica di base è quella di stabilire l'iperpiano separatore che meglio si presta a dividere lo spazio delle feature, massimizzando quindi il margine tra le due parti. La SVM sfrutta il cosiddetto "kernel trick" per mappare dataset non lineari in spazi di dimensione maggiore, in modo da facilitare la classificazione dei dati. I kernel che abbiamo utilizzato sono:

- *Lineare*: kernel più basilare, molto efficace per dataset con molte feature. Utilizzato soprattutto per problemi di text-classification.
- *Polinomiale*: forma più generalizzata del kernel lineare, meno efficiente e preciso.
- *Radiale*: considerato il kernel migliore per dataset non lineari.
- *Sigmoide*: derivante dallo studio sulle reti neurali, questo kernel si comporta come un percettrone a due strati.

Kernel	**Accuracy**	**Precision**	**F1 Score**	**Recall**	**95% CI**
Linear	0.9910	0.9985	0.9869	0.9756	(0.9858, 0.9947)
Polynomial	0.9900	0.9942	0.9855	0.9770	(0.9846, 0.9939)
Radial	0.9925	0.9971	0.9891	0.9813	(0.9877, 0.9958)
Sigmoid	0.9526	0.9426	0.9309	0.9195	(0.9423, 0.9615)

Tabella 1: Performance SVM

Si può notare, nella Tabella 1, come tutti i kernel applicati abbiano raggiunto livelli molto alti di Accuracy. Coerentemente a quello che ci aspettavamo, il picco (del 97%) lo si è raggiunto per il kernel Radiale, in quanto è quello che si presta meglio per dataset non lineari come lo è il nostro.

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	679	1
FALSE POSITIVE	17	1306

Tabella 2: Confusion Matrix: Linear SVM

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	680	4
FALSE POSITIVE	16	1303

Tabella 3: Confusion Matrix: Polynomial SVM

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	683	2
FALSE POSITIVE	13	1305

Tabella 4: Confusion Matrix: Radial SVM

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	640	39
FALSE POSITIVE	56	1268

Tabella 5: Confusion Matrix: Sigmoid SVM

Coerentemente ai risultati ottenuti, come si può vedere dal grafico in Figura 14, la curva ROC si avvicina di molto a quella di un modello perfetto, con dimensione dell'area sotto la curva (AUC) pari a 98.9982.

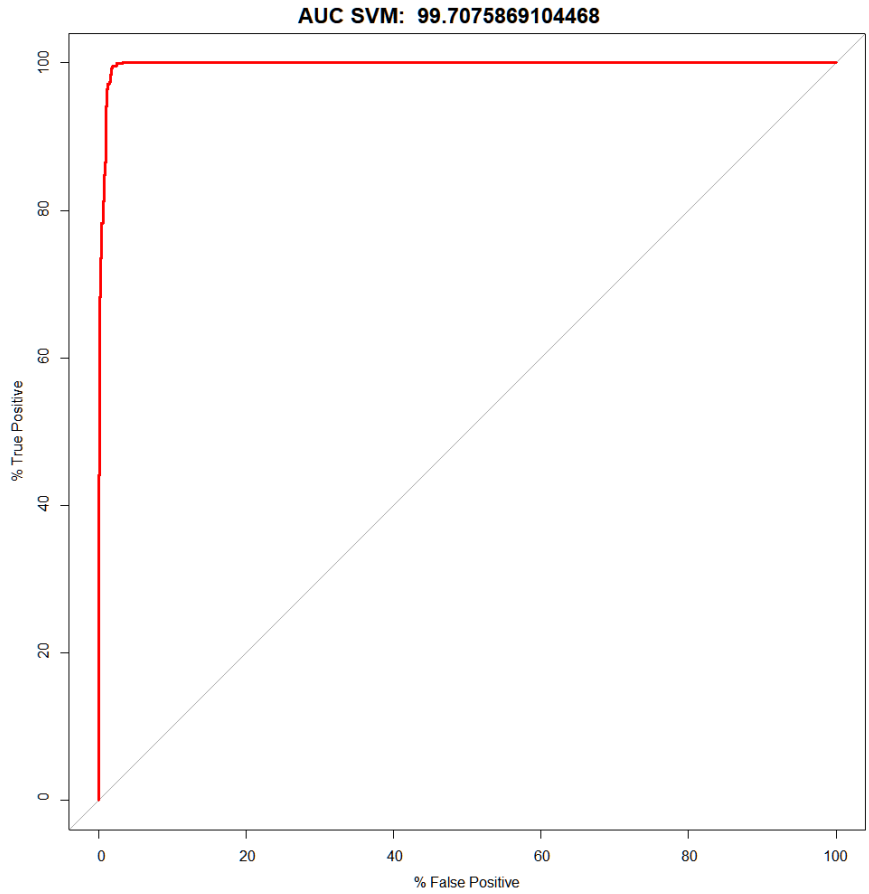

Figura 14: Curva ROC: Radial SVM

4.2 Neural Network

La rete neurale è un modello costituito da un insieme di nodi (neuroni) interconnessi tra loro che implementano l'algoritmo del percettrone. Quest'ultimo si basa sul *teorema di convergenza del percettrone*, per il quale, dato un insieme di dati x dove questi sono ripartiti in due classi A e B linearmente separabili, allora esiste un vettore di pesi w tale che:

$$\begin{aligned} < w,\, x > \;\geq\; 0, \quad x \in A \\ < w,\, x > \;<\; 0, \quad x \in B \end{aligned}$$

Per implementare questo modello abbiamo sfruttato la funzione *train()* del package *caret* con parametro *method* impostato a *nnet* (package *nnet*). La funzione ha testato varie "architetture" per la rete, mantenendo un solo layer nascosto ma modificando il numero di nodi per quest'ultimo e il valore del parametro *decay* (weight decay), utilizzato sia per migliorare il processo di ottimizzazione, sia per evitare l'overfitting. I risultati presentati sono quelli ottenuti con la configurazione migliore fornita dalla funzione, ovvero con 5 nodi che compongono il layer nascosto, e parametro *decay* pari al suo valore di default, cioè zero.

Accuracy	**Precision**	**F1 Score**	**Recall**	**95% CI**
0.9830	0.9729	0.9756	0.9784	(0.9764, 0.9882)

Tabella 6: Performance Neural Network

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	681	19
FALSE POSITIVE	15	1288

Tabella 7: Confusion Matrix Neural Network

Come si può notare dalle precedenti Tabelle 6 e 7, i risultati ottenuti con questo modello si sono dimostrati piuttosto soddisfacenti, riuscendo a classificare correttamente circa il 97% delle istanze del testset. L'efficienza del modello è visibile anche nel seguente grafico in Figura 15 che mostra la ROC prodotta.

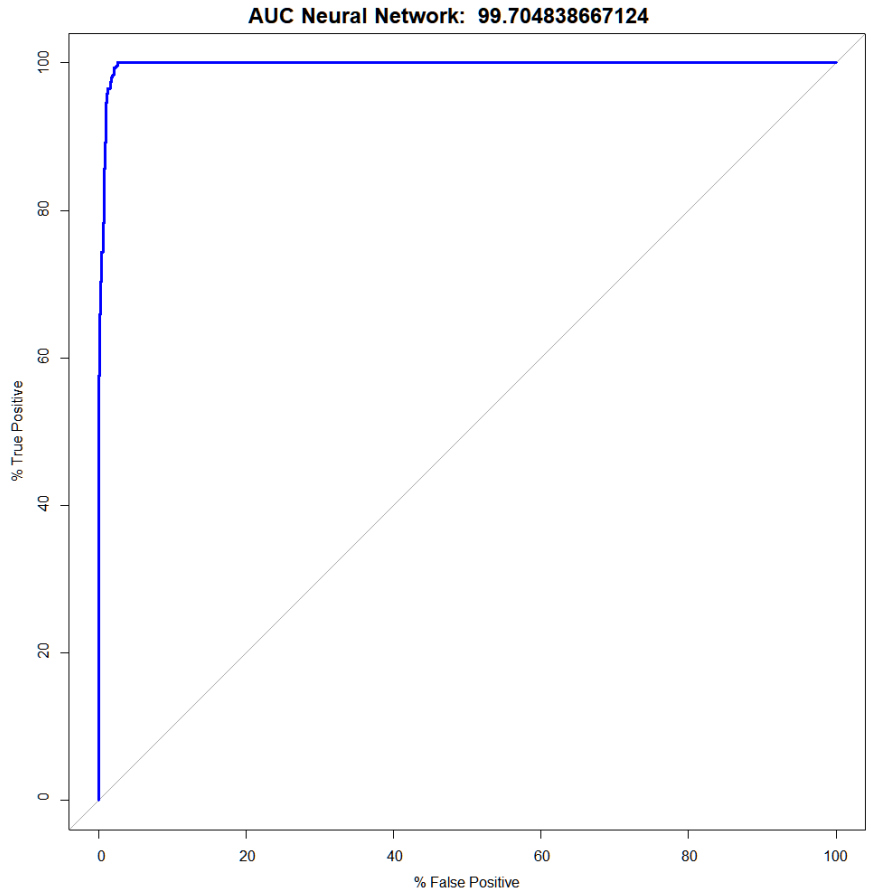

Figura 15: Curva ROC: Neural Network

4.3 Nayve Bayes

Il classificatore Naive Bayes non rappresenta un singolo algoritmo di apprendimento automatico, bensì una famiglia di algoritmi nella quale tutti i componenti condividono il principio chiave di supporre che ogni caratteristica dei dati del problema sia indipendente dalle altre (da qui il termine "naive", in inglese "ingenuo"). Nel nostro codice questo modello è stato implementato con la funzione *naiveBayes* fornita dal package omonimo. Di seguito vengono presentati i risultati ottenuti con tale modello:

Accuracy	Precision	F1 Score	Recall	95% CI
0.9651	0.9347	0.9506	0.9670	(0.9561, 0.9727)

Tabella 8: Performance Naive Bayes

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	673	47
FALSE POSITIVE	23	1260

Tabella 9: Confusion Matrix Naive Bayes

Come si può notare dalle Tabelle 8 e 9 i risultati, seppur più distanti dai valori ottimali rispetto ai modelli precedenti, mostrano come questo modello costituisca anch'esso un valido classificatore per il nostro problema. Ciò è visibile anche in Figura 16 qua di seguito.

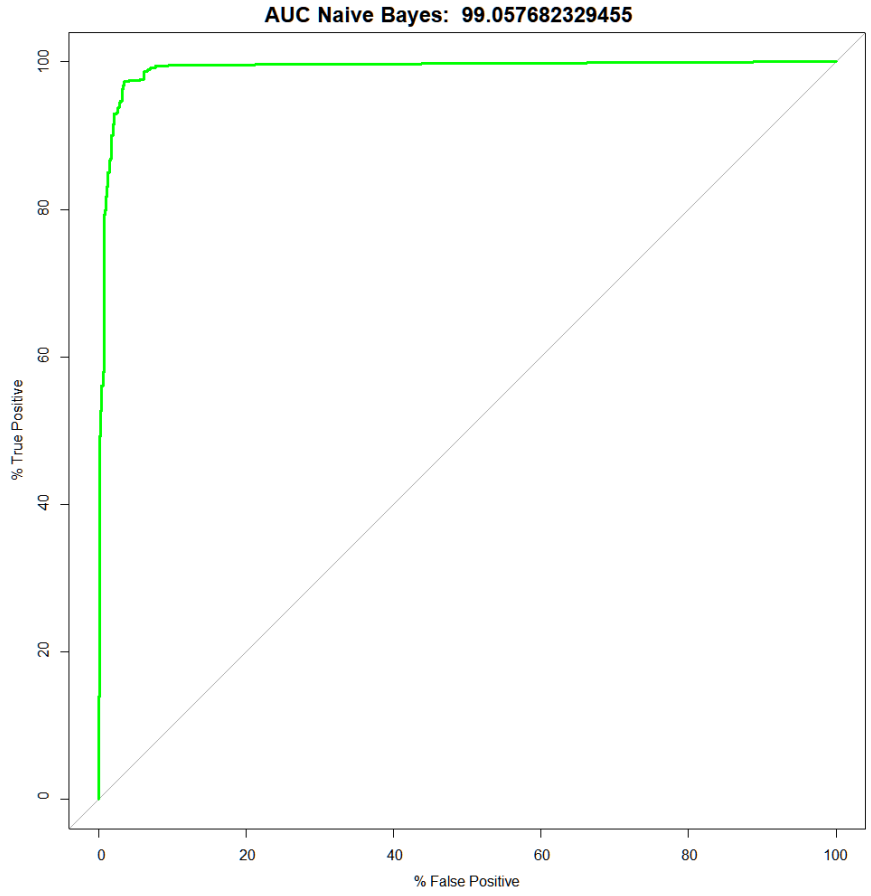

Figura 16: Curva ROC: Naive Bayes

4.4 Decision Tree e Random Forest

4.4.1 Decision Tree

Decision Tree rappresenta un metodo di apprendimento automatico supervisionato non parametrico che può essere utilizzato, così come gli altri modelli trattati nel corso di

questo elaborato, sia per problemi di classificazione che di regressione. L'idea generale è quella di costruire un albero in cui ogni diramazione rappresenta una scelta del tipo *if_then_else* e ogni foglia rappresenta una decisione finale (nel nostro caso la classe di appartenenza). Di seguito in Figura 17 è mostrato lo schema dell'albero ottenuto senza porre vincoli sulla massima profondità di diramazione. Tale vincolo non è stato preso in considerazione in quanto l'albero ottenuto non è mai risultato eccessivamente complesso in alcuna delle esecuzioni.

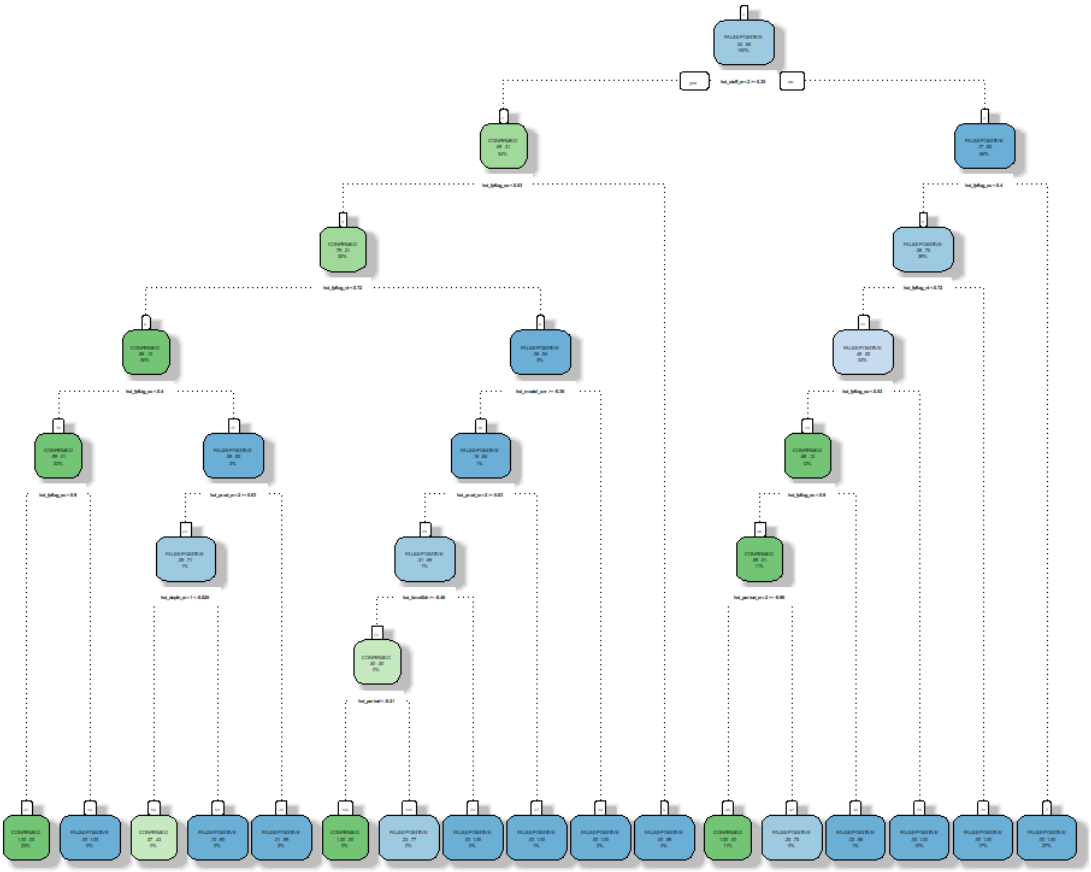

Figura 17: Decision Tree con massima libertà di crescita dei rami

I risultati ottenuti con tale modello sono mostrati in Tabella 10.

Accuracy	**Precision**	**F1 Score**	**Recall**	**95% CI**
0.9820	0.9853	0.9738	0.9626	(0.9752, 0.9874)

Tabella 10: Performance Decision Tree

Prediction\Reference	**CONFIRMED**	**FALSE POSITIVE**
CONFIRMED	670	10
FALSE POSITIVE	26	1297

Tabella 11: Confusion Matrix Decision Tree

Come si può notare dalle Tabelle 10 e 11 i risultati si mostrano molto vicini a quelli di un classificatore perfetto, come mostrato dalla curva ROC in Figura 18. Ciò rendere difficile pensare che il modello seguente, la Random Forest, possa fornire una performance maggiore.

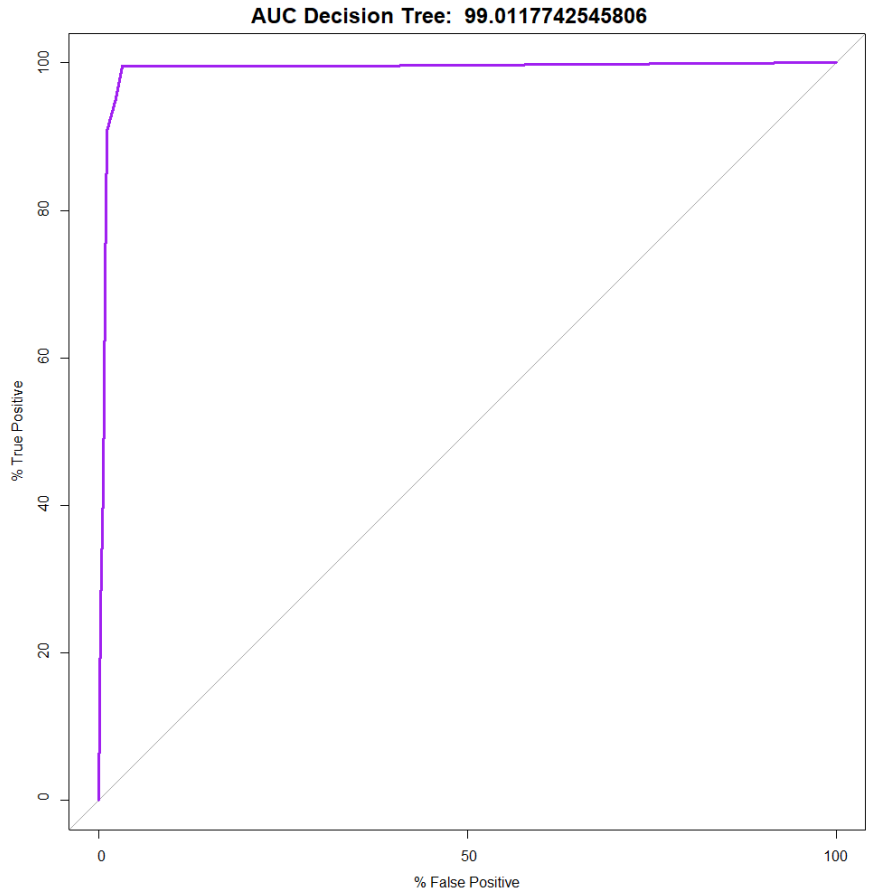

Figura 18: Curva ROC: Decision Tree

4.4.2 Random Forest

La Random Forest (foresta casuale) è un modello di apprendimento automatico supervisionato il cui funzionamento si basa sull'apprendimento di più modelli decisionali, spesso (come in questo caso) alberi di decisione, per formare un unico modello di previsione più potente. Abbiamo implementato questo modello con la funzione *randomForest* fornita dall'omonimo package. I risultati sono quelli mostrati nelle tabelle qui riportate di seguito.

Accuracy	**Precision**	**F1 Score**	**Recall**	**95% CI**
0.9885	0.9985	0.9832	0.9684	(0.9828, 0.9927)

Tabella 12: Performance Random Forest

Prediction\Reference	CONFIRMED	FALSE POSITIVE
CONFIRMED	674	1
FALSE POSITIVE	22	1306

Tabella 13: Confusion Matrix Random Forest

Come si può vedere dalle tabelle precedenti e dalla ROC in Figura 19, vi è un leggerissimo incremento nella performance, quasi nullo. Ciò, in linea con le nostre previsioni, è da attribuire certamente al fatto che i risultati ottenuti col metodo precedente sono già molto vicini all'ottimo. In generale un modello aggregato come quello in questione fornisce risultati migliori se e solo se il modello singolo (albero di decisione) soffre di instabilità, cosa che non è certamente riscontrata nel nostro caso.

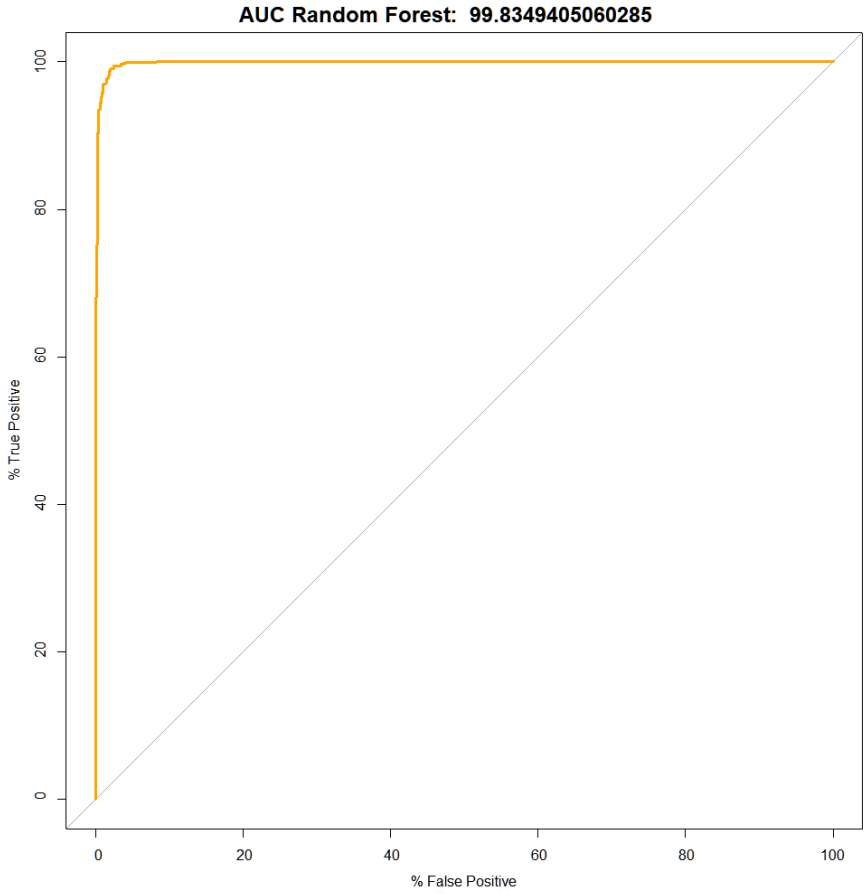

Figura 19: Curva ROC: Random Forest

4.5 Considerazioni sui risultati

Come si può vedere in Tabella 14 il modello Random Forest risulta essere nettamente il più oneroso in termini di tempi di calcolo per la 10-fold cross-validation. Anche Support Vector Machine e Neural Network, anche se in misura nettamente inferiore rispetto a Random Forest, risultano più complessi rispetto agli altri modelli, ovvero Naive Bayes e Decision Tree, che data la loro scarsa complessità sono risultati essere i modelli più semplici dal punto di vista computazionale.

Model	**Time (s)**
SVM	77.25
Naive Bayes	3.75
Neural Network	132.06
Decision Tree	2.41
Random Forest	318.80

Tabella 14: Tempi di calcolo per la 10-fold cross-validation

Di seguito vengono mostrati in Tabella 15 i risultati riguardo alle statistiche *AUC*, *sensitività* e *specificità* messi a confronto per ogni modello. Come si può notare, tutti i risultati si mostrano molto accurati, in linea con le nostre previsioni derivanti dallo studio del dataset e delle distribuzioni delle variabili.

ROC	**Min.**	**Max.**	**Mean**
SVM	0.9952065	0.9999165	0.9983278
Naive Bayes	0.9787093	0.9969911	0.9896631
Neural Network	0.9929832	0.9998538	0.9981900
Decision Tree	0.9705111	0.9991905	0.9887615
Random Forest	0.9909157	1.0000000	0.9984002

SENSITIVITY	**Min.**	**Max.**	**Mean**
SVM	0.9430380	0.9936709	0.9750437
Naive Bayes	0.9367089	0.9873418	0.9676342
Neural Network	0.9554140	1.0000000	0.9794875
Decision Tree	0.9119497	1.0000000	0.9670209
Random Forest	0.9177215	0.9936306	0.9517684

SPECIFICITY	**Min.**	**Max.**	**Mean**
SVM	0.9934426	1.0000000	0.9982517
Naive Bayes	0.9245902	0.9770492	0.9565188
Neural Network	0.9836066	1.0000000	0.9962845
Decision Tree	0.9701987	1.0000000	0.9933993
Random Forest	0.9934426	1.0000000	0.9986892

Tabella 15: Analisi dei risultati

5 Conclusioni

Dei modelli che abbiamo implementato nel corso di questo elaborato, SVM, Neural Network e Random Forest si sono dimostrati i modelli migliori a pari merito dal punto di vista dell'accuratezza dei risultati. Coerentemente a ciò, questi rappresentano anche i modelli più complessi dal punto di vista computazionale. In particolare Random Forest ha impiegato un tempo quasi tre volte superiore a Neural Network, il secondo più oneroso. In generale tutti e tre hanno impiegato un tempo di gran lunga superiore a Naive Bayes e Decision Tree, che seppur mostrando un'accuratezza generale dei risultati meno vicina all'ottimo rispetto agli altri, hanno impiegato un tempo mediamente inferiore di quasi 60 volte. Questi risultati, molto vicini a quelli di un classificatore perfetto, sono sicuramente da attribuire alle caratteristiche del dataset utilizzato. La distribuzione non omogenea delle variabili ha sicuramente giocato un ruolo rilevante in questo. Infatti come si può vedere nelle figure qui di seguito vi è grande disparità nell'importanza delle variabili utilizzate per il training e testing dei diversi modelli (vengono mostrate le prime 15 variabili sulle 28 utilizzate).

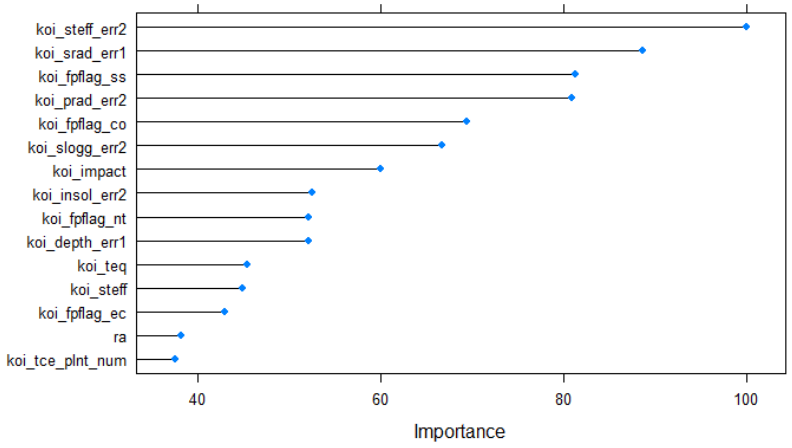

Figura 20: Importanza delle variabili nel modello SVM

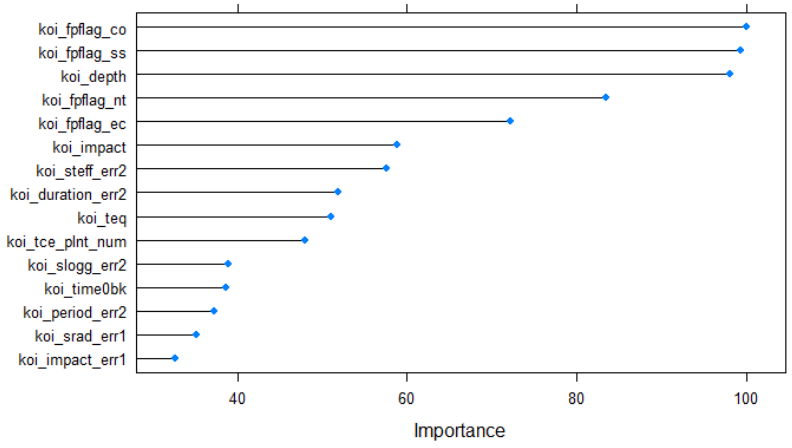

Figura 21: Importanza delle variabili nel modello Neural Network

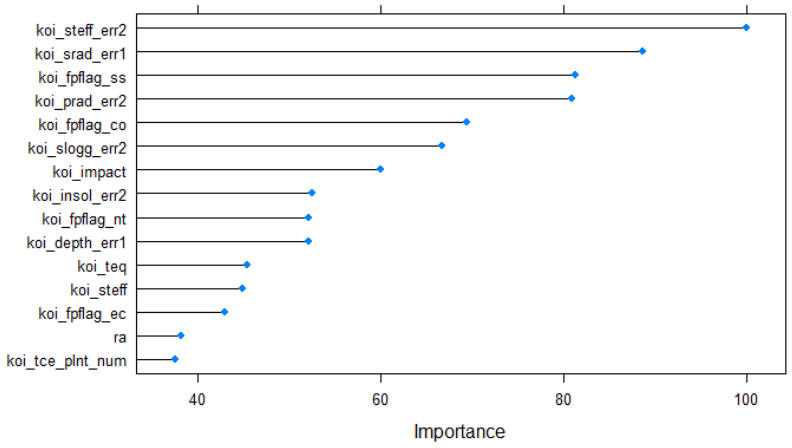

Figura 22: Importanza delle variabili nel modello Naive Bayes

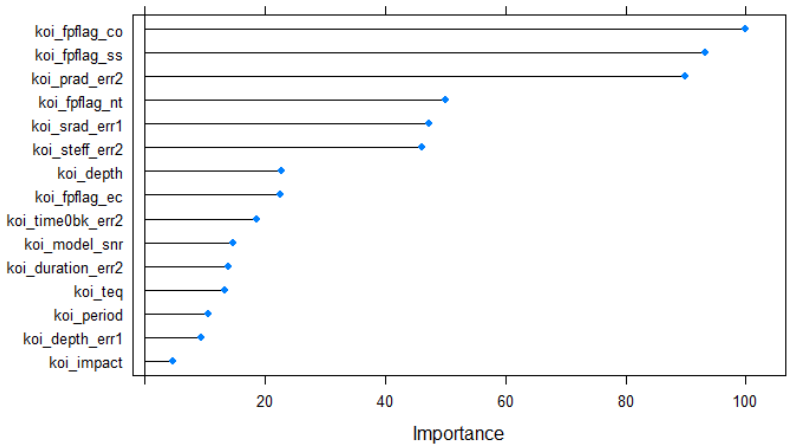

Figura 23: Importanza delle variabili nel modello Decision Tree

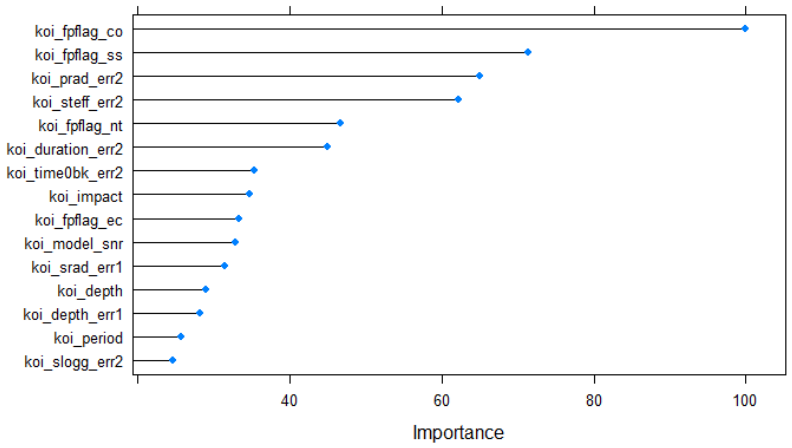

Figura 24: Importanza delle variabili nel modello Random Forest

Come è possibile notare sia dai grafici precedenti e soprattutto dalle distribuzioni riportate nelle Figure 25 e 26, tra le variabili con importanza maggiore ricoprono un ruolo decisivo soprattutto le quattro variabili *flag*, risultando fortemente sbilanciate verso le istanze con label "FALSE POSITIVE". Questo è dovuto al significato stesso di queste variabili.

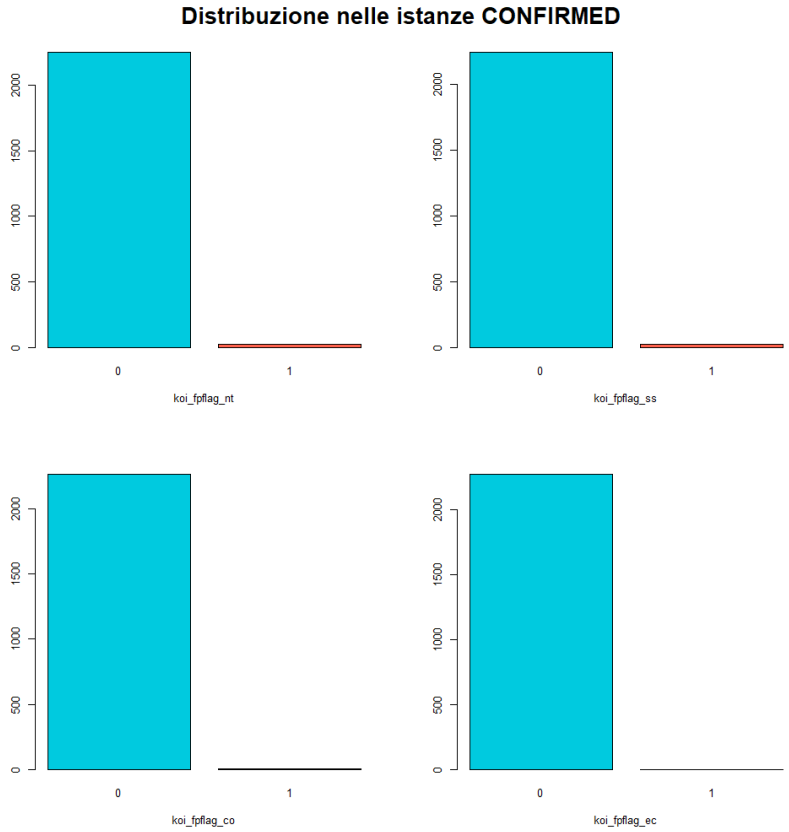

Figura 25: Distribuzioni delle variabili *flag* nelle istanze con label CONFIRMED

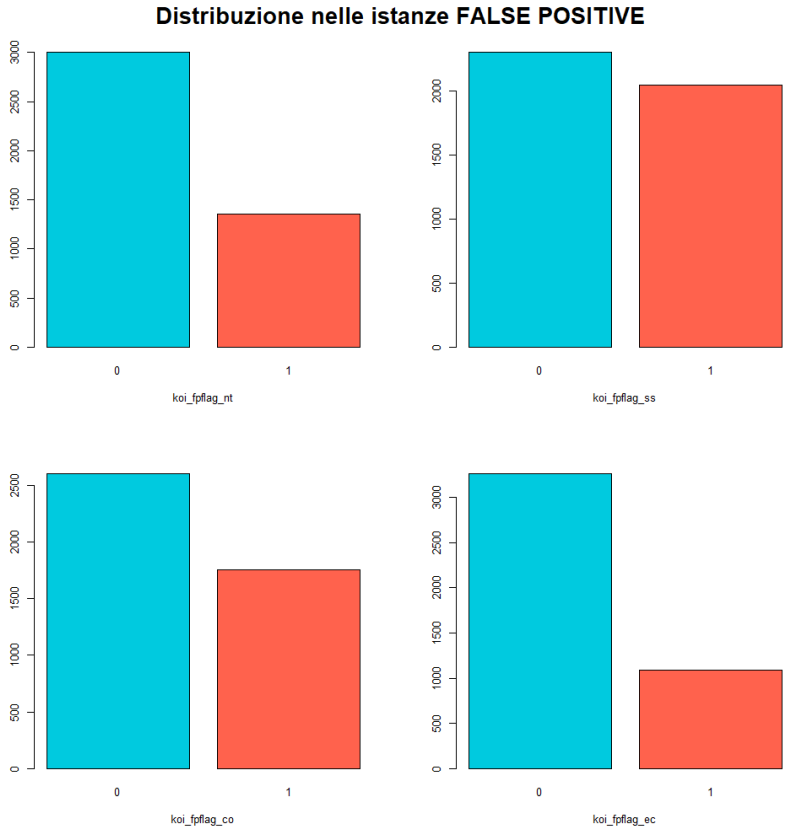

Figura 26: Distribuzioni delle variabili *flag* nelle istanze con label FALSE POSITIVE

Lo sbilanciamento della distribuzione delle variabili *flag* è reso ancora più incisivo dallo sbilanciamento delle label stesse, così come mostrato in Figura 1.