# *Progetto Machine Learning*

Progetto per il corso di Machine Learning a.2020/21
Università degli Studi di Milano-Bicocca

---

Artemisia Sarteschi
829677

a.sarteschi@campus.unimib.it

# Indice

# Capitolo 1

# Introduzione

Un asteroide è un piccolo corpo celeste con una composizione simile ad un pianeta terrestre, generalmente non presentano una forma sferica ed hanno solitamente un diametro inferiore al kilometro, esiste però la possibilità di osservarne di grandi dimensioni.

La loro formazione si pensi derivi da dei residui di disco protoplanetario che non sono stati incorporati dai pianeti durante la formazione del Sistema Solare. Gli asteroidi hanno spesso orbite caratterizzate da un'elevata eccentricità.

Il dataset scelto per questa analisi racchiude al suo interno vaste informazioni riguardanti gli asteroidi ed il nostro obbiettivo sarà quello di andare a predire, attraverso le informazioni contenute all'interno di questo dataset, se un asteroide sia pericoloso (*Hazardous*) oppure può essere ritenuto sicuro.

# Design Architetturale

Il dataset scelto presenta 2655 istanze di tipo differente (numeric, integer e factor) e 25 attributi. Di seguito vengono presentati nel dettaglio gli attributi del dataset.

## 1.1 Descrizione degli Attributi

- **Neo.Reference.ID**: ID assegnato ad ogni asteroide. [int]
- **Absolute.Magnitude**: Denota la magnitudine assoluta di un asteroide, ovvero la magnitudine visiva che un osservatore registrerebbe se l'asteroide fosse posizionato a 1 unità astronomica (AU) di distanza e a 1 AU dal Sole e con un angolo di fase zero. [num]
- **Est.Dia.in.KM.min**: Diametro minimo stimato dell'asteroide in Km. [num]
- **Est.Dia.in.KM.max**: Diametro massimo stimato dell'asteroide in Km. [num]
- **Relative.Velocity.km.per.sec**: Velocità relativa in kilometri per secondi. [num]
- **Miles.per.hour**: Miglia orarie [num]
- **Orbit.ID**: Id assegnato alla particolare orbita [int]
- **Orbit.Uncertainity**: Incertezza dell'orbita. [int]

- **Minimum.Orbit.Intersection** : Minima distanza all'intersezione orbitale, utilizzata per valutare il rischio di collisione tra due oggetti astronomici[num]
- **Jupiter.Tisserand.Invariant**: Denota l'invariante Tisserand, esso è utilizato per distinguere le tipologie differenti di orbite. [num]
- **Epoch.Osculation**: Concide con l'orbita gravitazionale di Keplero. [num]
- **Eccentricity**: Valore dell'eccentricità dell'orbita dell'asteroide. [num]
- **Semi.Major.Axis**: Valore del semi asse maggiore dell'orbita dell'asteroide. [num]
- **Inclination**: Inclinazione dell'orbita. [num]
- **Asc.Node.Longitude**: Elemento necessario per per specificarel'orbita di un oggetto nello spazio. [num]
- **Orbital.Period**: Tempo orbitale, ovvero il tempo necessario per compiere un giro completo attorno al suo corpo orbitante. [num]
- **Perihelion.Distance**: Distanza del perielio. [num]
- **Perihelion.Time**: Durata del perielio [num].
- **Aphelion.Dist**: Distanza dell'afelio. [num]
- **Mean.Anomaly**: Anomalia media. [num]
- **Mean.Motion**: scostamento medio. [num]
- **Hazardous**: Denota quando un asteroide è pericoloso o meno. [factor]

## 1.2 Preparazione del Dataset

La prima operazione che andiamo a effettuare sul nostro dataset è la verifica dell'omogeneità della tipologia dei dati per permette il loro confronto. All'interno del nostro dataset però non sono presenti dati ad normalizzare, la variabile target è presente come `string` e viene portata in `factor`. Il secondo controllo che andiamo ad effettuare è la presenza di eventuali valori mancanti, che nel nostro caso non vengono rilevati.

Andiamo quindi ad analizzare il target e successivamente a condurre la PCA (Principal Component Analysis) per rimuovere gli attributi ridondanti.

## 1.3 Analisi del Target

La figura [1.1] mostra la distribuzione dell'attributo target *Hazardous*. L'attributo target rappresenta la possibilitò che un asteroide sia pericoloso o meno (True / False). Dall'istogramma sottostante possiamo vedere come siano stati indicati non pericolosi un numero maggiore di asteroidi: 59% non pericolosi contro 41% pericolosi.

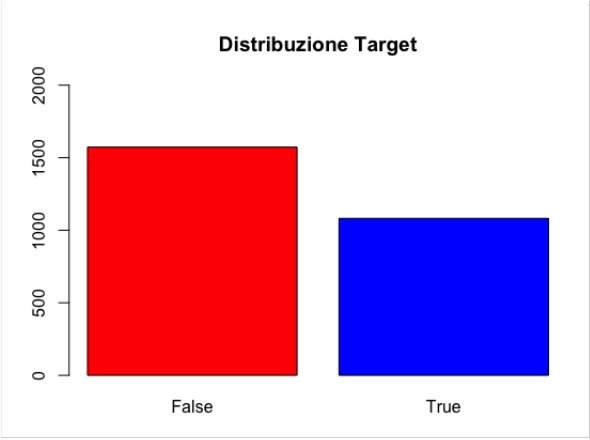



Figura 1.1: Distribuzione del Target

## 1.4 Principal Component Analysis

I dataset molto spesso contengono una o più features ridondanti che vanno ad esprimere informazioni simili rispetto ad altre features. Si rende quindi necessario tenere solamente quelle features in grado di fornire informazioni significative riguardo al dataset. Andiamo dunque ad eseguire una riduzione della dimensionalitò del dataset avvalendoci alla tecnica della PCA (*Principal Component Analysis*). La PCA è una procedura statistica che utilizza una trasformazione ortogonale per con- vertire un insieme di osservazioni di variabili correlate tra loro, pesantemente o leggermente, in un insieme di valori di variabili linearmente non correlate, chiamate componenti principali, mantenendo la varianza presente nel set di dati nella misura massima. La trasformazione è definita cosicchè il primo componente principale abbia la maggiore varianza possibile e ogni componente successivo abbia a sua volta la maggior varianza possibile sotto il vincolo di essere otogonale ai componenti precedenti. I vettori risultanti, chiamati autovettori, sono un insieme di basi ortogonali non correlate. Di seguito l'applicazione della tecnica.

### 1.4.1 Calcolo della Pca

Applicando la tecnica della PCA al nostro dataset otteniamo la tabella riportata in figura [1.2]. In particolare i risultati ottenuti rappresentano l'eigenvalue (autovettore) di ogni dimensione, ovvero la quantità di varianza espressa da ognuna di esse. Le dimensioni principali hanno un valore di eigenvalue maggiore poichè vanno ad esprimere una varianza maggiore dei dati, sono quindi le più importanti.

La scelta di quali dimensioni mantenere durante la riduzione del contenuto informativo ha seguito i seguenti tre criteri:

1. **Criterio dell'autovalore maggiore o regola di Kaiser**: si scelgono i primi componenti con autovalori superiore a 1.

2. **Quota di varianza totale spiegata**: le dimensioni scelte dovrebbero spiegare almeno dal 70% all'80% della varianza totale.

3. **Scree graph**: utilizzando un grafico (Figura [1.3]) chiamato degli autovalori in funzione del numero di componenti principali. Poiché gli autovalori sono decrescenti, il grafico assume la forma di una spezzata con pendenza sempre negativa. Analizzando il grafico, si potrà individuare un punto nel quale si manifesta una brusca variazione di pendenza, in corrispondenza della quale si individua il numero di componenti principali da considerare.

| | eigenvalue | variance.percent | cumulative.variance.percent |
|---|---|---|---|
| Dim.1 | 5.569596e+00 | 2.227838e+01 | 22.27838 |
| Dim.2 | 4.593102e+00 | 1.837241e+01 | 40.65079 |
| Dim.3 | 2.583044e+00 | 1.033218e+01 | 50.98297 |
| Dim.4 | 2.155498e+00 | 8.621993e+00 | 59.60496 |
| Dim.5 | 1.821984e+00 | 7.287936e+00 | 66.89290 |
| Dim.6 | 1.151564e+00 | 4.606256e+00 | 71.49915 |
| Dim.7 | 1.065438e+00 | 4.261753e+00 | 75.76091 |
| Dim.8 | 1.002909e+00 | 4.011638e+00 | 79.77254 |
| Dim.9 | 9.536950e-01 | 3.814780e+00 | 83.58732 |
| Dim.10 | 9.164383e-01 | 3.665753e+00 | 87.25308 |
| Dim.11 | 8.357555e-01 | 3.343022e+00 | 90.59610 |
| Dim.12 | 6.976596e-01 | 2.790638e+00 | 93.38674 |
| Dim.13 | 4.812819e-01 | 1.925128e+00 | 95.31186 |
| Dim.14 | 2.980647e-01 | 1.192259e+00 | 96.50412 |
| Dim.15 | 2.692277e-01 | 1.076911e+00 | 97.58103 |
| Dim.16 | 2.339401e-01 | 9.357605e-01 | 98.51679 |
| Dim.17 | 1.806118e-01 | 7.224472e-01 | 99.23924 |
| Dim.18 | 1.399718e-01 | 5.598873e-01 | 99.79913 |
| Dim.19 | 3.184456e-02 | 1.273783e-01 | 99.92651 |
| Dim.20 | 1.718090e-02 | 6.872360e-02 | 99.99523 |
| Dim.21 | 9.231724e-04 | 3.692690e-03 | 99.99892 |
| Dim.22 | 2.692533e-04 | 1.077013e-03 | 100.00000 |
| Dim.23 | 2.796609e-21 | 1.118644e-20 | 100.00000 |
| Dim.24 | 7.180991e-24 | 2.872396e-23 | 100.00000 |
| Dim.25 | 2.132748e-29 | 8.530992e-29 | 100.00000 |

Figura 1.2: Eigenvalues

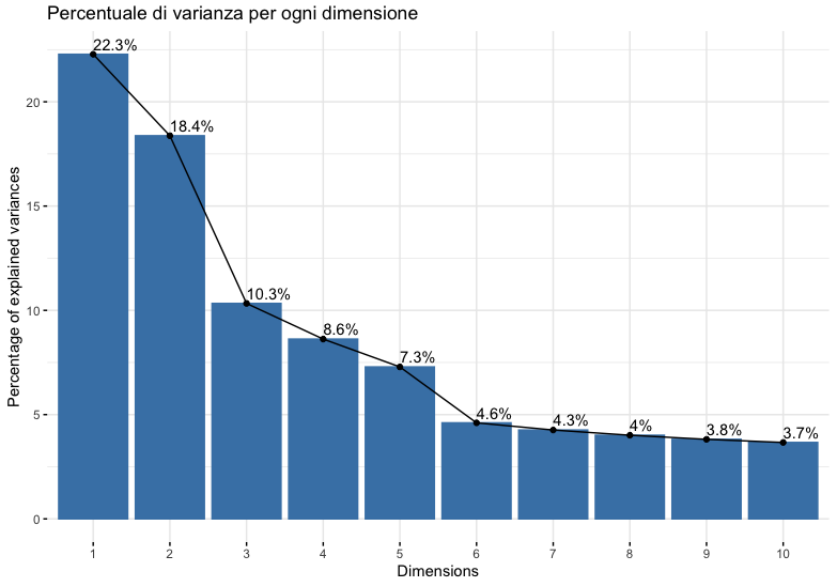



Figura 1.3: Scree graph

Secondo il primo criterio sembrerebbe corretto selezionare le prime otto dimensioni. Considerando invece anche il secondo si considererebbero tra le 6 e le 8 dimensioni. Seguendo invece l'andamento del grafico possiamo vedere una brusca pendenza tra la seconda e terza e la quinta e sesta. Dal momento che, se si scegliesse uno dei due salti evidenziati nel grafico non si raggiungerebbe il 70% di varianza totale spiegata, che per il secondo criterio le dimensioni sarebbero corrette tra la sesta e la ottava, che il valore dell'ottava dimensione è molto vicino ad uno, scegliamo di prendere le dimensioni fino alla settima cosìda avere un valore di varianza totale spiegata del 75%. Scelgo quindi di ridurre il dataset da 25 a 7.

Si scegli quindi di considerare solo le features più espressive studiando il grafico riportato in Figura [1.4]. Si può notare che in tale grafico sono state inserite un numero maggiore di dimensioni di quelle decise per la riduzione, questo si è reso necessario quando facendo il grafico con le esatte sette dimesioni scelte è risultato come varie dimensioni fossero fortemente positivamente correlate, ovvero esprimessero informazioni molto simili. Per questo motivo il grafico è stato prodotto con un numero superiore per andare a scegliere una variabile tra quelle positivamente correlate, in particolare quella con una distanza

maggiore dall'origine che esprimono una qualità maggiore rispetto alle altre.

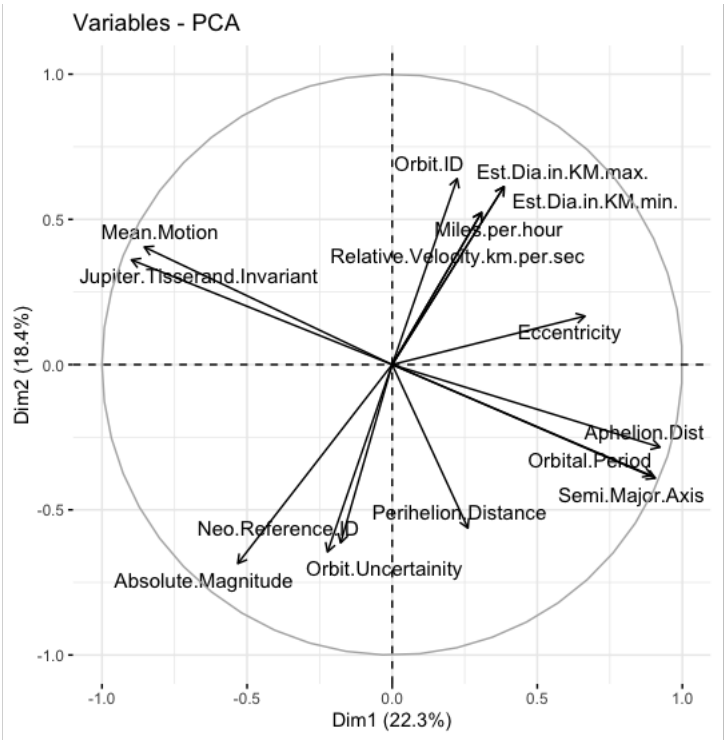



Figura 1.4: Grafico dei Risultati della PCA

Seguendo tutte le motivazione presentate, andiamo ad individuare come pricipali features:

- **Absolute.Magnitude**
- **Est.Dia.in.KM.max.**
- **Orbit.Uncertainity**
- **Jupiter.Tisserand.Invariant**
- **Eccentricity**
- **Semi.Major.Axis**
- **Perihelion.Distance**

### 1.4.2 Confronto tra PCA e Matrice di Correlazione

Nell'ottica di controllare che i risultati ottenuti con la PCA siano corretti andiamo a confrontarli con la matrice di correlazione applicata al dataset.

I risultati ottenuti dalla matrice di correlazione variano tra $-1$ e $+1$; il primo valore indica una proporzionalità inversa tra le variabili del dataset, mentre il secondo una proporzionalità diretta.

Il grafico in Figura [1.5] ci aiuta in questa visualizzazione, poiché rappresenta la correlazione tra una coppia di covariate. In particolare, ogni cerchio va a rappresentare due informazioni.

- La sua *dimensione* rappresenta la correlazione, più la dimensione è elevata più la correlazione tra le variabili è elevata, viceversa la correlazione è minima.
- Il suo *colore* è associato a specifici valori: più il colore si avvicina al blu, più la correlazione è positiva, viceversa se si avvicina al rosso corrisponde ad una correlazione negativa.

Notiamo quindi che l'analisi precedentemente condotta è coerente con i risultati in Figura [1.5].

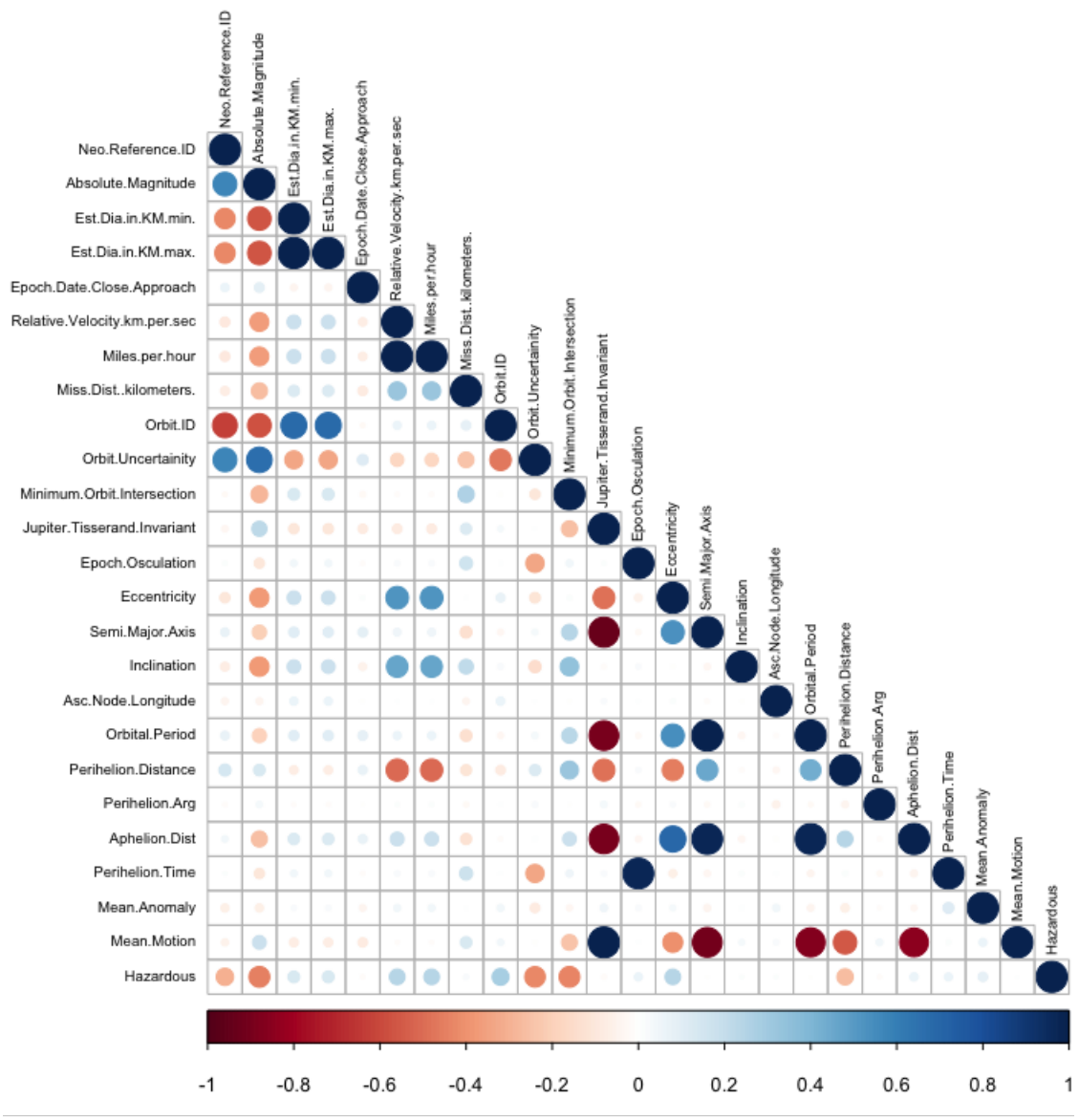



Figura 1.5: Matrice di Correlazione

# Capitolo 2

# Modelli di Machine Learning

Considerando l'impossibilità di avere e costruire un classificatore/modello universale, dobbiamo valutare per ogni problema quale tecnica risulti migliore, valutando la bontà del singolo classificatore applicato ad un dato problema di classificazione. Si noti che per la valutazione di un modello l'unico strumento a nostra disposizione è il modello stesso ed il dataset di addestramento. Nel nostro caso abbiamo condotto gli esperimenti sul dataset ridotto dopo aver eseguito la PCA; quindi solamente con le features selezionate e la colonna target, in modo che ad ogni dato in input posse associata una label. In particolare, l'associazione della label sarà fondamentale dal momento che i modelli scelti sono tutti supervisionati.

I modelli che sono stati presi in considerazione sono i seguenti:

- Naive Bayes
- Neural Network
- Support Vector Machine

Per quanto rigurda SVM abbiamo deciso di effettuare una comparazione tra SVM: *Radiale*, *Polinomiale* e *Lineare*, così da valutare il kernel migliore da utilizzare in questo contesto.

Per l'implementazione in R abbiamo impiegato la libreria *Caret* con metodi specifici per ogni modello.

## 2.1 Naive Bayes

Il Naive Bayes (NB) è un algoritmo di apprendimento supervisionato che costruisce un modello di classificazione che utilizza i dati derivanti dagli eventi precedenti per predire gli eventi futuri. Questo algoritmo assume che l'effetto di un attributo su una data classe è indipendente dai valori degli altri attributi. Tale assunzione, chiamata indipendenza condizionale delle classi, ha lo scopo di semplificare i calcoli e proprio per questo l'algoritmo prende il nome di naive.

La tecnica si basa sul teorema di Bayes che definisce la probabilità condizionata (o a posteriori) di un evento rispetto ad un altro.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

L'algoritmo determina la classe di appartenenza in base alla probabilità condizionali per tutte le calssi in base agli attributi dei vari elementi. La classificaizone corretta si ha quando la probabilità condizionale di una certa classe C rispetto agli attributi $A_n(P\{C|A_1, A_2, ..., A_n\})$ è massima.

## 2.2 Neural Networks

Il modello di rete neurale è un modello superivisionato che crea previsioni basate su dati esistenti. Essa esegue calcoli per rilevare le caratteristiche comuni e decide se un determinato input appartiene o meno ad una specifica classe, nel caso dei nostri dataset se è pericoloso (*Hazardous* = `TRUE`) oppure no (*Hazardous* = `FALSE`).
Una rete neurale si compone di tre parti principali:

- **Livelli di input**: accetta input in base a dati esistenti.
- **Livelli nascosti**: utilizzano la backpropagation per ottimizzare i pesi delle variabili di input al fine di migliorare la potenza predittiva del modello.

- **Livelli di output**: restituiscono le previsioni basate sui dati di input e dei livelli nascosti.

## 2.3 Support Vector Machine

La Support Vecotr Machine è un algoritmo di apprendimento superivisionato che costruisce un modello di classificazione che si basa sull'addestrare cercando l'iperipiano che rende massimo il margine per entrambe le classi di punti considerate all'interno del training set. La SVMrichiede che le loro variabili di input siano numeriche, quindi esprimibili in binario ed il nostro dataset è già ben formato per questo modello. La SVM è stata scelta perchè performa molto bene con problemi di classificazione e con dataset di dimensioni contenute e bilanciati, anche se il nostro dataset non è perfettamente bilanciato.

Per valutare eventuali miglioramenti delle prestazioni abbiamo sperimentato con tre tipologie di kernel differenti: *radiale*, *polinomiale* e *lineare*. Anche per questo modello ci siamo basati sulla libreria `caret` con i metodi: `svmRadial`, `svmPoly` e `svmLinear`.

## 2.4 K-fold Cross Validation

Per ottenere risultati scarsamente dipendenti dalla divisione in train set e test set, all'interno del trainset abbiamo effettuato una *k-fold cross validation*. Questo metodo consiste nella suddivisione del dataset totale in k parti di uguale numerosità e, ad ogni passo, la k-esima parte del dataset viene considerata come test set, mentre la restante parte costituisce il training set. Questo permette al nostro modello di fare il train sui k-1 campioni non esclusi e con essi cercare di predire il k-esimo escluso, evitando quindi problemi di overfitting.

Nel nostro caso abbiamo eseguito una 10-fold croos validation, s empre implementata con la libreria *caret*.

# Capitolo 3

# Esperimenti

Una volta individuati i modelli sono stati eseguiti diversi esperimenti per andare ad individuare il migliore. Per i vari esperiemnti prendiamo in considerazione il dataset risultante dopo aver applicato la PCA e viene effettuata una divisione del 70% per il trainset e 30% per il test set. Per tutti i modelli viene effettuato l'addestramento tramite 10-Fold Cross Validation. Le metriche di valutazione prese in considerazione sono le matrici di confusione così raprpesentate:

| Reference | | |
|---|---|---|
| **Prediction** | Absence | Presence |
| Absence | TN | FP |
| Presence | FN | TP |

1. True negative (TN): La predizione è corretta, l'asteroide non è pericoloso;
2. True positive (TP): La predizione è corretta, l'asteroide è pericoloso;
3. False negative (FN): La predizione è sbagliata, l'asteroide è pericoloso ma viene classificato sicuro.
4. False positive (FP): La predizione è sbagliata, l'asteroide non è pericoloso ma viene classificato come tale;

In questo contesto, i falsi positivi sono solo un falso allarme che a successivi controlli potrebbero essere declassati come non pericolosi. Ma al contrario se non ci si accorgesse in tempo di un asteroide pericoloso mal classificato potrebbero verificarsi dei problemi gravi. Oltre alle matrici di confusioni sono state considerate altre metriche altrettanto importanti:

*Accuracy*: È il rapporto tra gli asteroidi etichettati correttamente e l'intero gruppo di asteroidi.
Accuracy = $(TP + TN)/(TP + FP + FN + TN)$

*Precision*: È il rapporto tra gli asteroidi pericolosi etichettati correttamente dal modello e tutti quelli etichettati come pericolosi, anche in modo errato.
Precision = $TP/(TP + FP)$

textitRecall (detta anche Sensitivity): È il rapporto tra gli asteroidi che sono stati correttamente etichettati pericolosi dal modello e tutti coloro che sono pericolosi effettivamente nella realtà.
Recall = $TP/(TP + FN)$

*F-Measure*: È la media armonica della Precision e del Recall. La F-Measure è buona se c'è una sorta di equilibrio tra Precision e Racall nel modello.
F-Measure = $2 * (Recall * Precision)/(Recall + Precision)$

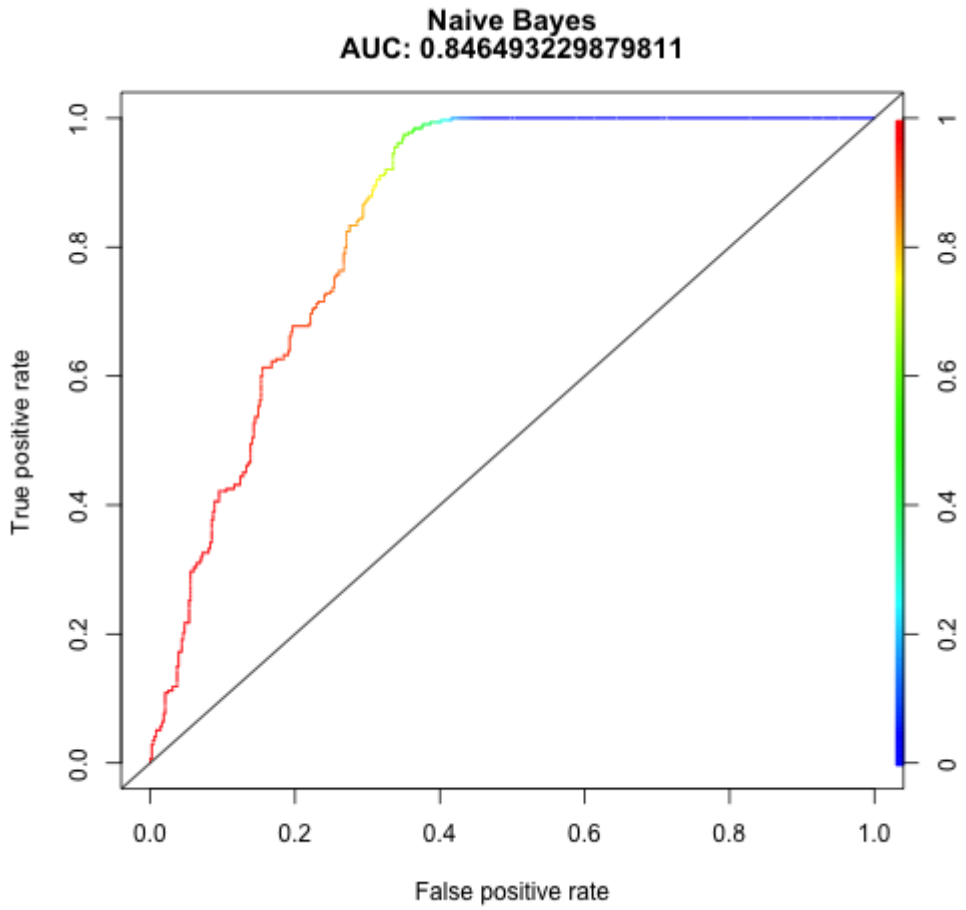
**Tutti i risultati ottenuti dei modelli sono riportati all'ultima esecuzione.**

## 3.1 Naive Bayes

I risultati ottenuti per il primo modello *Naive Bayes*, ottenuti utilizzando la libreria `caret` con metodo `naive_bayes`, sono i seguenti:

| | |
|---|---|
| **Accuracy** | 0.7726 |
| **Precision** | 0.9840 |
| **Recall** | 0.6356 |
| **F-Measure** | 0.7723 |
| **AUC** | 0.8465 |
| **CI 95%** | (0.7419, 0.8013) |

| | **False** | **True** |
|---|---|---|
| **False** | 307 (39%) | 5 (1%) |
| **True** | 176 (22%) | 308 (39%) |



## 3.2 NN

Il secondo modello supervisionato che abbiamo utilizzato è stato *Neural Network*. La libreria `caret`, impiegata per tutti i modelli di questo report, si appoggia al package *nnet* per questo modello. Questa package ci permette di creare una rete neurale a singolo layer nascosto di cui abbiamo eseguito due train differenti, per andare a valutare quale ci permettesse di avere la migliore accuratezza: il primo con il tuning fornito di default da caret e il successivo fornito da noi con un numero di neuroni compresi tra 1 e 7 con pesi variaibli tra 0.1 e 0.5.

Abbiamo quindi ottenuto i seguenti valori ottimali di *neuroni nascosti* e *accuratezza*.

| | **Accuracy** | **Size** | **AUC** |
|---|---|---|---|
| **NN (caret)** | 0.7876884 | 5 | 0.8677065 |
| **NN (tuned)** | 0.7814070 | 7 | 0.8639228 |

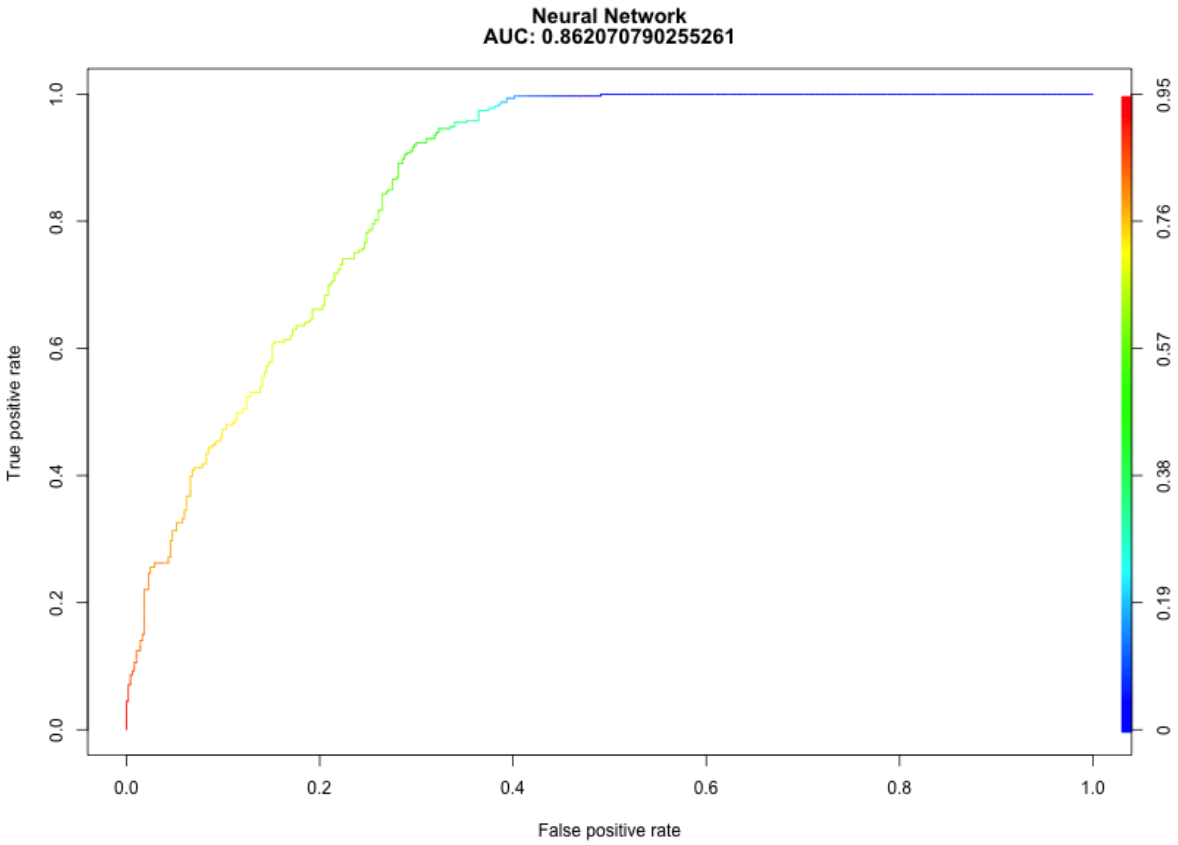
Avendo ottenuto un accuratezza superiore e un $AUC$ (area sottostante la curva $ROC$) maggiore nel caso venga lasciato il tuning di default del package, decidiamo di utilizzare per la comparazione con gli altri modelli i risultati ottenuti dal primo dei train.

### 3.2.1 Risultati ottenuti per la Neural Network

I risultati per il secondo modello *Neural Network* sono i seguenti:

| | |
|---|---|
| **Accuracy** | 0.7877 |
| **Precision** | 0.9313 |
| **Recall** | 0.7019 |
| **F-Measure** | 0.8005 |
| **AUC** | 0.8677 |
| **CI 95%** | (0.7576, 0.8156) |

| | **False** | **True** |
|---|---|---|
| **False** | 339 (43%) | 25 (3%) |
| **True** | 144 (18%) | 288 (36%) |



## 3.3 SVM, con comparazioni

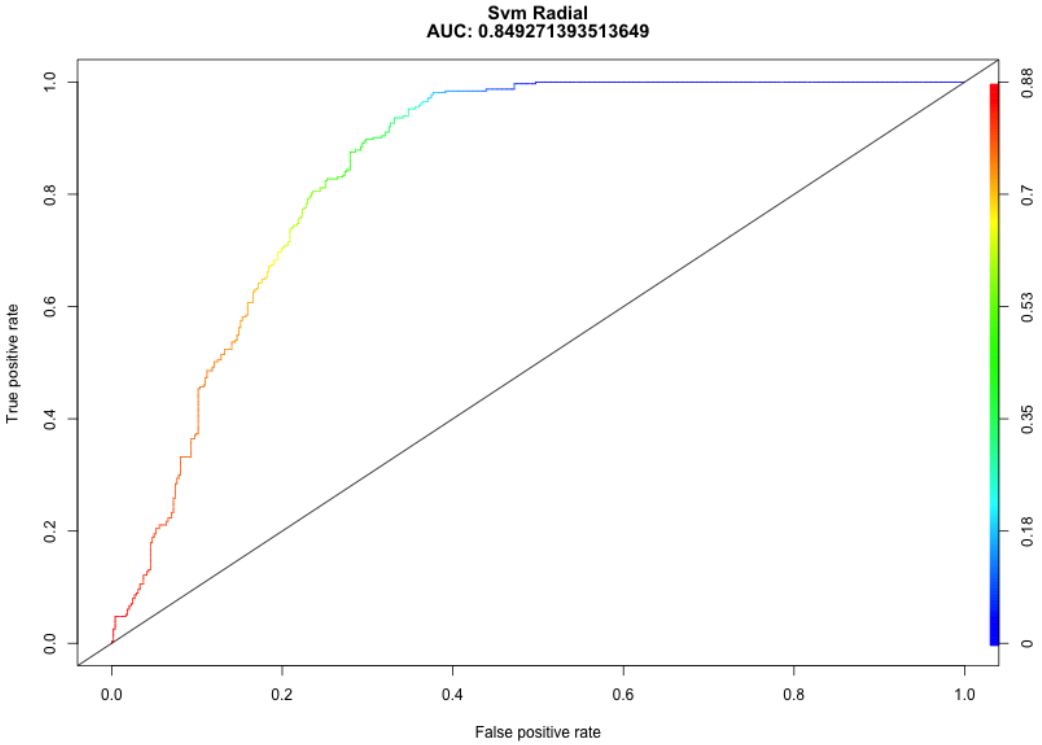
### 3.3.1 Kernel Radiale

Il primo kernel utilizzato è stato quello *radiale* e dopo la valutazione dei risultati ottenuti dal tuning fornito libreria, abbiamo deciso di avvalerci di questo.

I risultati ottenuti per il kernel sono i seguenti.

| | |
|---|---|
| **Accuracy** | 0.7751 |
| **Precision** | 0.8689 |
| **Recall** | 0.7412 |
| **F-Measure** | 0.8000 |
| **AUC** | 0.8493 |
| **CI 95%** | (0.7445, 0.8037) |

| | **False** | **True** |
|---|---|---|
| **False** | 358 (45%) | 54 (7%) |
| **True** | 125 (14%) | 259 (34%) |



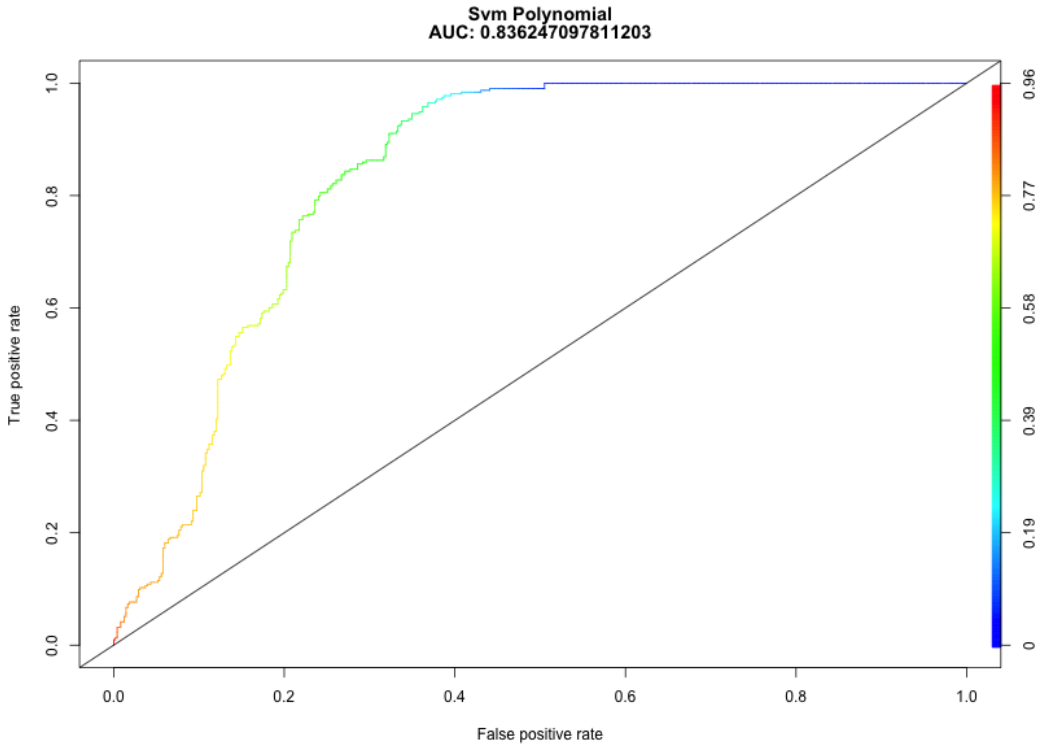
### 3.3.2 Kernel Polinomiale

Il secondo kernel utilizzato è stato quello *polinomiale* e dopo la valutazione dei risultati ottenuti dal tuning fornito libreria, abbiamo deciso di avvalerci di questo.

I risultati ottenuti per il kernel sono i seguenti.

| | |
|---|---|
| **Accuracy** | 0.7739 |
| **Precision** | 0.8741 |
| **Recall** | 0.7329 |
| **F-Measure** | 0.7973 |
| **AUC** | 0.8362 |
| **CI 95%** | (0.7432, 0.8025) |

| | **False** | **True** |
|---|---|---|
| **False** | 354 (44%) | 51 (7%) |
| **True** | 129 (16%) | 262 (33%) |



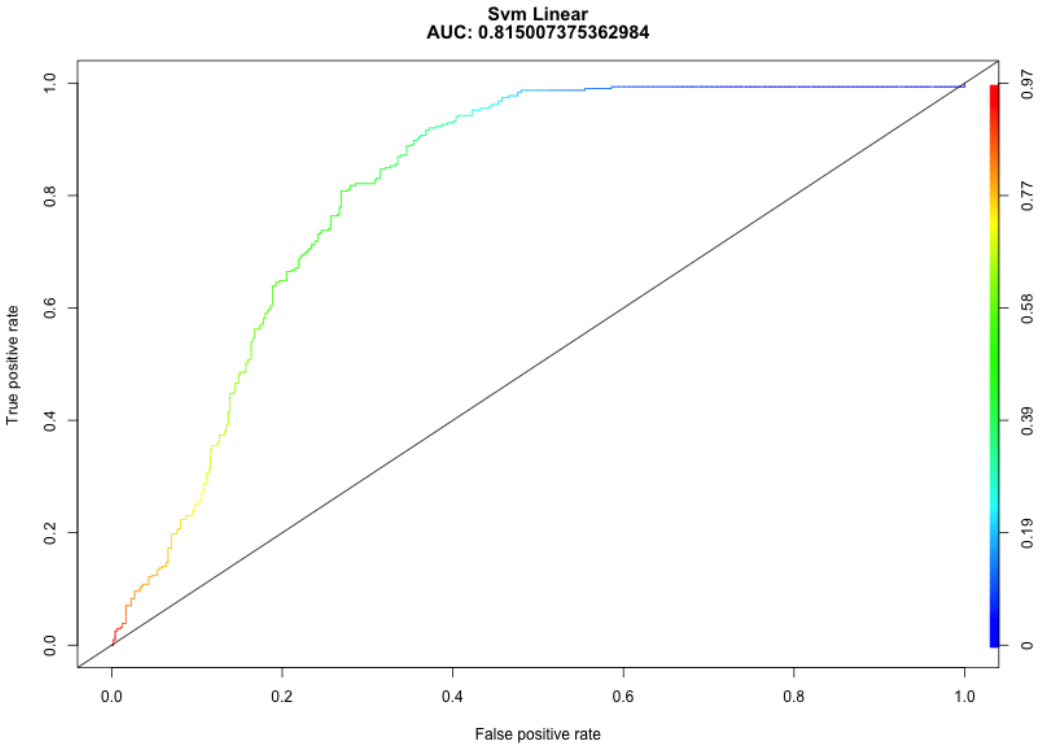
### 3.3.3 Kernel Lineare

Il terzo kernel utilizzato è stato quello *Lineare* anche in questo caso dopo la valutazione dei risultati ottenuti dal tuning fornito libreria, abbiamo deciso di avvalerci di questo.

I risultati ottenuti per il kernel sono i seguenti.

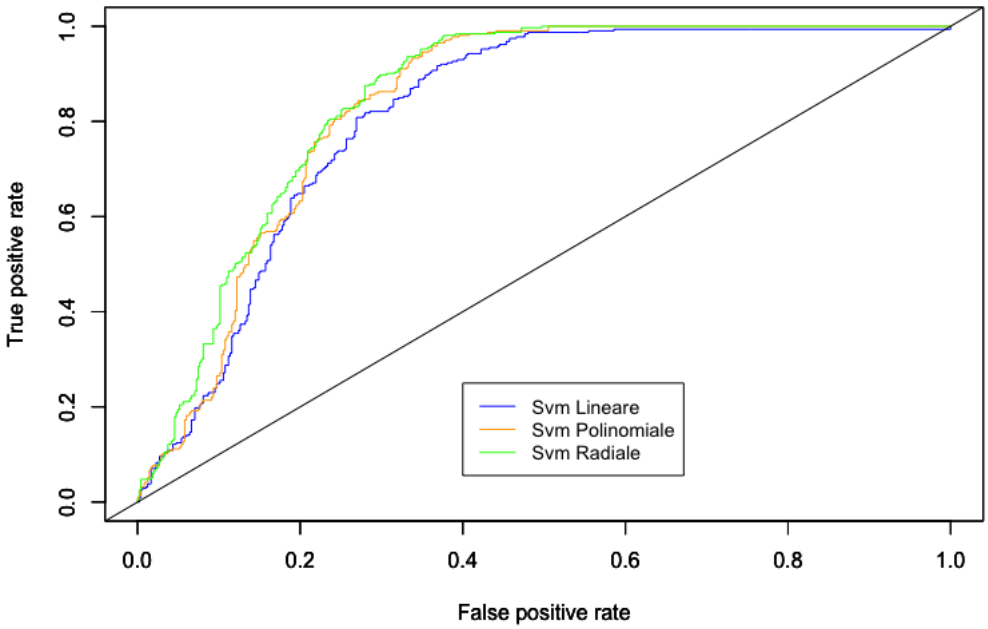
| | |
|---|---|
| **Accuracy** | 0.7462 |
| **Precision** | 0.8129 |
| **Recall** | 0.7557 |
| **F-Measure** | 0.7833 |
| **AUC** | 0.8150 |
| **CI 95%** | (0.7145, 0.7761) |

| | **False** | **True** |
|---|---|---|
| **False** | 365 (46%) | 84 (11%) |
| **True** | 118 (15%) | 229 (28%) |



### 3.3.4 Comparazione dei Kernel

Nell'ottica di comparare i kernel precedentemente utilizzati abbiamo creato un ulteriore grafico delle curve *ROC* e riportiamo in tabella i valori relativi alle *AUC* per ogni *kernel*.




In seguito alla valutazione dei risultati ottenuti scegliamo il *kernel radiale* come migliore, confronteremo quindi questo con gli altri modelli.

## 3.4 Comparazione

Compariamo quindi i risultati ottenuti dai modelli, nella Figura [3.1] possiamo vedere che per un livello di confidenza pari a 0.95 la Neural Network è quella con un intervallo di confidenza maggiore.

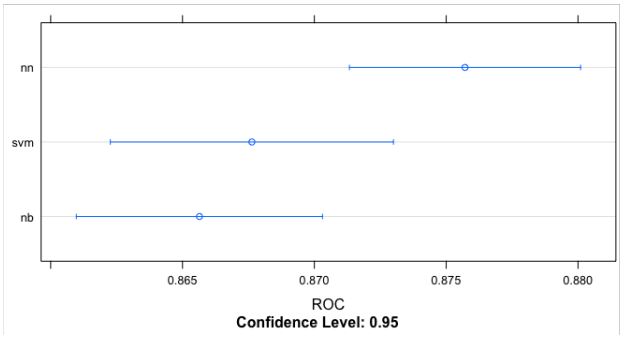



Figura 3.1: Interavalli di Confidenza

In figura [3.2] possiamo osservare diversi boxplot per i valori che contribuiscono alla creazione delle curve ROC. Per quanto riguarda il valore di Specificity possiamo osservare che le performance di NN e SVM sono simili, in particolare si va a distaccare Naive Bayes. Per il valore di ROC sembra essere lievemente migliore NN rispetto alle altre, per la Sensitivity sembra invece migliore il comportamento della SVM anche se molto vicina a Neural Network.

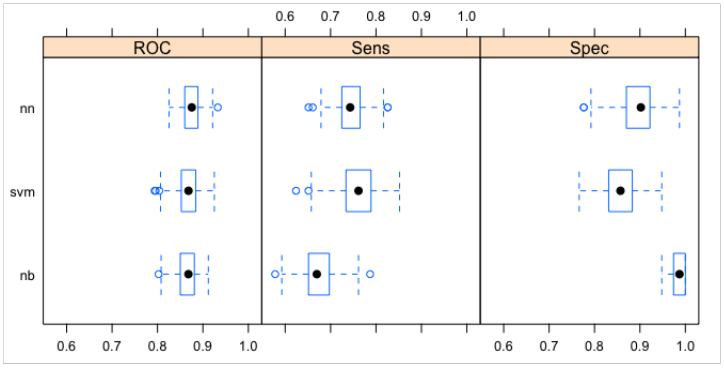
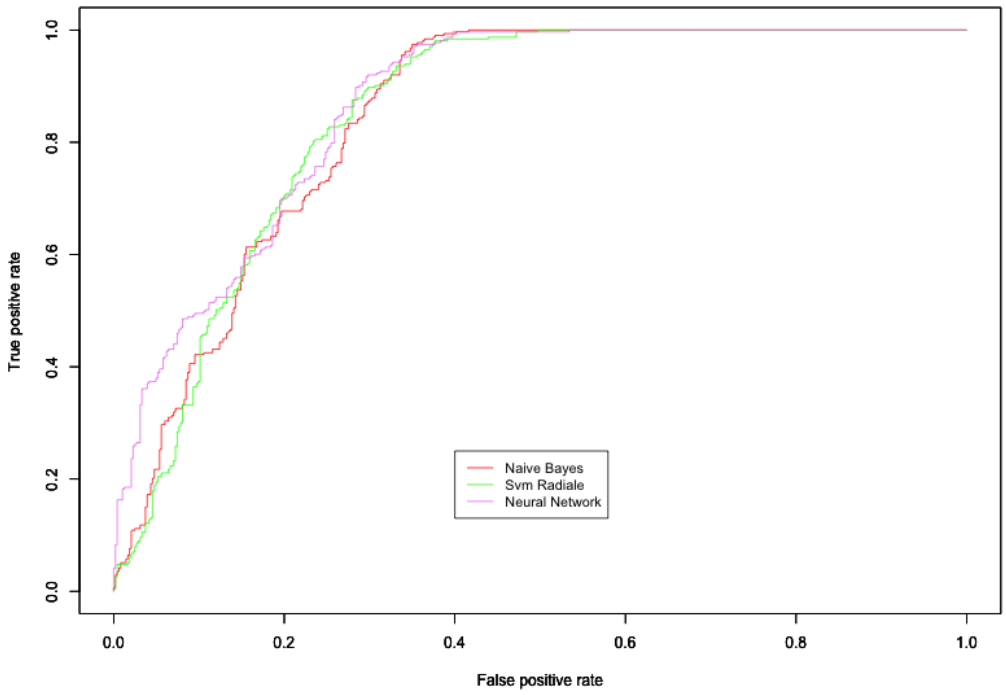



Figura 3.2: Boxplot

Confrontiamo infine le curve ROC dei modelli e le relative tempistiche di addestramento. Notamo come Naive Bayes sia quello con l'esecuzione nettamente più veloce rispetto agli altri due modelli, notiamo però anche che il modello NN sia quello ad avere una più alta $AUC$. Vediamo in fine dal grafico che non c'è un modello che chiaramente, come nel caso precedente dei kernel della $SVM$ abbiamo una curva $ROC$ che si mantiene sempre superiore alla altre.




| | AUC | Time | Final Model |
|---|---|---|---|
| **Naive Bayes** | 0.84649323 | 3.570 | 0.020 |
| **Neural Network** | 0.86207079 | 80.975 | 0.250 |
| **SVM Radiale** | 0.84927139 | 75.064 | 0.327 |

# Capitolo 4

# Conclusioni

Dopo aver confrontato i vari modelli implementati la conclusione è stata che, sebbene tutti i modelli fossero tutti molto vicini come prestazioni, la Neural Network performi meglio dei precedenti. La scelta ricade su questo modello perchè ,benchè la SVM con kernel radiale abbiamo dei risultati lievemente migliori per quanto riguarda la recall (ovvero la riduzione dei falsi negativi), tutti gli altri parametri considerati risultano superiori. Le prestazioni però della SVM con kernel radiale non sono comunque da considerarsi negative. Sarebbe comunque consigliabile confermare queste considerazioni con un dataset più numeroso.