Vicidomini Filippo 884022

Università Ca' Foscari di Venezia

2º Assignment – Informatica II, Modulo 1

Lo scopo di questo assignment è lo studio dei classificatori discriminativi e generativi e imparare le varie differenze tra questi.

Per comodità, i file con il codice sono in formato *Jupiter Notebook.*

Term Frequency – Inverse Document Frequency (TF – IDF)

TF-IDF è una tecnica di analisi del testo utilizzata per valutare l'importanza di una parola all'interno di un documento rispetto ad altri documenti in un determinato insieme.

TF-IDF si basa su due concetti: il *term frequency* (TF) e *l'inverse document frequency* (IDF). Il TF misura la frequenza di una parola all'interno di un singolo documento, mentre l'IDF valuta l'importanza di una parola all'interno di un intero insieme di documenti.

Per calcolare il valore TF-IDF di una parola, si moltiplica il suo valore TF per il suo valore IDF. Parole con un alto valore TF-IDF sono considerate più importanti perché sono utilizzate frequentemente all'interno di un singolo documento, ma non sono comuni in tutti i documenti.

TF-IDF viene spesso utilizzato nell'analisi del testo per identificare le parole chiave in un documento, per il riassunto automatico di testi e per il Clustering di documenti simili.

Matematicamente si può definire come:

$$\mathrm{W}_{ij} = tf_{ij} * \log(\mathrm{N}/\mathrm{ni})$$

dove:

- W_{ij} è il peso della parola i all'interno del documento j
- tf_{ij} è il numero di volte che la parola i compare nel documento j
- N è il numero di documenti nel corpus
- ni è il numero di documenti nel corpus che contengono la parola i
- log è il logaritmo in base e

La base del logaritmo è *e* perché più bassa è la base, più alto è il risultato, il che può influire sul troncamento dei risultati di ricerca impostati per punteggio. Dal punto di vista matematico, la base del logaritmo non ha importanza in quanto può essere cambiata con la formula:

$$\log_b x = \ln x \,/\, \ln b$$

Dal punto di vista pratico, la base *e* è più adatta per il calcolo di tf-idf.

Ad occuparsi di trasformare il dataset in un formato utilizzabile per gli algoritmi di Machine Learning è il file *convert_data_to_tfidf.py*. Che prende il file *csv* di partenza *spambase.csv* e crea il file *spam_tfidf.csv* che contiene il dataset in formato tf-idf.

Classificatori Discriminativi e Generativi

I classificatori discriminativi sono modelli di classificazione che prendono in input un'istanza di dati e producono una previsione sulla classe a cui appartiene l'istanza. In altre parole, questi modelli basano la loro previsione sulla base di ciò che hanno già visto in passato, utilizzando le informazioni raccolte durante il processo di addestramento per determinare la classe a cui un'istanza di dati appartiene.

I classificatori generativi, d'altra parte, sono modelli che cercano di generare una rappresentazione probabilistica delle diverse classi presenti nei dati. In altre parole, questi modelli non solo prevedono la classe a cui un'istanza di dati appartiene, ma cercano anche di generare una rappresentazione probabilistica della distribuzione dei dati all'interno di ogni classe. In questo modo, i classificatori generativi possono essere utilizzati per generare nuovi dati che appartengano a una determinata classe.

Support Vector Machine (SVM)

Il *support vector machine* è un algoritmo di machine learning che permette di classificare dati in una categoria o nell'altra in base ad un insieme di dati di training. Inizialmente era un algoritmo utilizzato per problemi di classificazione binaria. L'obiettivo che si pone è di massimizzare il margine presente tra due classi di dati e l'iperpiano trovato. La distanza, definita come:

$$d_i = \frac{|w^T x_i + w_0|}{||w||^2}$$

Può essere massimizzata minimizzando il denominatore attraverso il metodo dei moltiplicatori di Lagrange, ovvero una tecnica per studiare i massimi e i minimi vincolati di una funzione a più variabili. Si ottiene dunque un espressione che coinvolge i parametri Alpha, e sostituiti questi valori nella lagrangiana possiamo effettivamente ottimizzare il tutto.

Si definisce algoritmo di apprendimento supervisionato e parametrico, in quanto per poter funzionare necessita di un insieme di dati di training e non necessita di conoscere a priori la forma della funzione di apprendimento, è inoltre discriminativo, in quanto cerca di trovare una funzione di apprendimento che massimizza la distanza tra le due classi.

Per il *teorema di Mercer* possiamo sostituire il prodotto scalare con una funzione *K* detta *Kernel* per ottenere una mappatura non lineare ad alta dimensione dei dati. Scelto il Kernel adeguato l'iperpiano restituito dalla SVM tenderà a seguire in maniera appropriata i dati.

SVM Lineare

Una SVM lineare è un modello di classificazione utilizzato per prevedere a quale di due classi appartiene un dato esempio. A differenza di altri modelli di classificazione una SVM lineare cerca di trovare la linea (nota come "iperpiano" in uno spazio ad alta dimensione) che meglio separa i dati nelle due classi. Una volta trovato questo iperpiano, la SVM può essere utilizzata per fare previsioni su nuovi dati.

funzione di apprendimento:

$$f(x) = w * x + b$$

dove:

- w è il vettore dei pesi
- x è il vettore delle features
- b è il bias
- f(x) è il valore predetto

La funzione di apprendimento è definita come il prodotto scalare tra il vettore dei pesi e il vettore delle features più il bias. Il bias è un valore che viene aggiunto alla funzione di apprendimento per poter spostare la funzione di apprendimento rispetto all'origine

---

SVM Polinomiale

Una SVM polinomiale è un tipo di macchina vettoriale di supporto che può essere utilizzata per classificare i dati in due o più classi. È simile a un SVM lineare, ma invece di trovare una linea per separare i dati, trova una funzione polinomiale per farlo. Ciò consente a SVM di acquisire modelli più complessi nei dati e fare previsioni più accurate.

funzione di apprendimento:

$$f(x) = (w * x + b)^d$$

dove:

- w è il vettore dei pesi
- x è il vettore delle features
- b è il bias
- d è il grado del polinomio
- f(x) è il valore predetto

La funzione di apprendimento è definita come il prodotto scalare tra il vettore dei pesi e il vettore delle features più il bias, elevato al grado del polinomio

---

SVM Radial Basis Function (RBF)

il support vector machine RBF ha come caratteristica di lavorare su infinite dimensioni. Corrisponde quindi a una trasformazione non lineare particolarmente complicata, tanto da renderla impossibile da utilizzare direttamente. In questo modo consente di apprendere limiti decisionali non lineari. Ma la funzione di apprendimento è particolarmente facile da calcolare, ed è possibile cambiare il parametro *gamma* per rendere il *Support vector classifier* più o meno complesso.

funzione di apprendimento:

$$f(x) = e^{-gamma*||x-c||^2}$$

dove:

- x è il vettore delle features
- c è il vettore dei centroidi
- gamma è il parametro gamma
- f(x) è il valore predetto
- ||x - c||^2 è la distanza euclidea tra il vettore delle features e il vettore dei centroidi

La funzione di apprendimento RBF è definita come la funzione esponenziale della distanza euclidea tra il vettore delle features e il vettore dei centroidi. Il vettore dei centroidi è un vettore che contiene i valori dei centroidi delle features del documento che si sta analizzando.

SVM Kernel Angolare

Il Support Vector Machine (SVM) con kernel angolare è un tipo di SVM che utilizza un kernel angolare per classificare i dati. Il kernel angolare è una funzione matematica che trasforma gli input di dati in uno spazio ad alta dimensione in modo che possano essere facilmente separati utilizzando un piano di separazione. Questo tipo di SVM è particolarmente utile quando i dati non possono essere facilmente separati usando un piano di separazione nello spazio originale a causa della loro complessa struttura.

Ma la funzione di apprendimento si può calcolare come

funzione di apprendimento:

$$f(x) = \cos^{-1}\left(\frac{(x * c)}{||x|| * ||c||}\right)$$

dove:

- x è il vettore delle features
- c è il vettore dei centroidi
- f(x) è il valore predetto
- ||x|| è la norma del vettore delle features
- ||c|| è la norma del vettore dei centroidi
- (x * c) è il prodotto scalare tra il vettore delle features e il vettore dei centroidi

Il kernel è definito positivo se il prodotto scalare tra il vettore delle features e il vettore dei centroidi è maggiore di zero. Altrimenti è definito negativo. Il valore predetto è quindi l'arco della funzione coseno del prodotto scalare tra il vettore delle features e il vettore dei centroidi, diviso per la norma del vettore delle features e la norma del vettore dei centroidi. Il valore predetto è compreso tra 0 e 1.

Random Forest Classifier

Random Forest Classifier è un algoritmo di apprendimento supervisionato utilizzato per la classificazione di dati. Si basa sulla costruzione di un insieme di alberi decisionali, noti come "foresta", utilizzando una tecnica di selezione casuale dei campioni dei dati e delle caratteristiche per ogni albero. Ogni albero nella foresta produce una propria previsione che viene poi combinata insieme per prendere la decisione finale.

Questo metodo di aggregazione di previsioni multiple consente di migliorare la precisione rispetto a quella ottenuta utilizzando un singolo albero decisionale.

Random Forest Classifier è uno dei metodi di classificazione più utilizzati in campo scientifico e industriale per la sua capacità di gestire grandi quantità di dati e di gestire con successo le situazioni in cui ci sono molte variabili o caratteristiche. Inoltre, è uno dei metodi più veloci ed efficienti per la classificazione di dati.

Struttura dell'albero

L'albero di decisione è una struttura dati che permette di classificare dati in base ad un insieme di regole. L'albero è composto da nodi e archi. Ogni nodo contiene una regola, mentre ogni arco rappresenta la risposta alla regola.

Il nodo radice è il primo nodo dell'albero, mentre i nodi foglia sono i nodi che non hanno archi uscenti.

Costruzione dell'ensemble

L'ensemble è composto da più alberi di decisione. Ogni albero è costruito in modo indipendente dagli altri alberi.

Ogni albero viene costruito in questo modo:

1. si prendono le features del dataset e si prendono *n* feature casuali
2. si prendono le istanze del dataset e si prendono *n* istanze casuali
3. si costruisce l'albero di decisione

Classificazione

Per classificare una nuova istanza, si passa per tutti gli alberi dell'ensemble, e si prende il valore predetto da ogni albero. Si prende la moda tra i valori predetti dagli alberi, e questa è la classe predetta.

K – Nearest Neighbors (KNN)

Il KNN (K Nearest Neighbors) è un algoritmo di classificazione non parametrico utilizzato per la previsione di una variabile di destinazione in base alle variabili d'input. L'algoritmo si basa sulla presunzione che gli elementi simili appartengano alla stessa classe. Nello specifico, l'algoritmo KNN

prende in considerazione i *k* elementi più vicini (neighbors) all'elemento di cui si vuole prevedere la classe e assegna la classe più presente tra i *k* elementi più vicini. Il valore di *k* viene scelto dall'utente e può influire sulla precisione delle previsioni.

L'algoritmo KNN è spesso utilizzato per problemi di classificazione binaria, ma può essere esteso anche a problemi di classificazione multi-classe.

È anche chiamato algoritmo di apprendimento *lazy* perché non apprende immediatamente dal set di addestramento, ma memorizza il set di dati e al momento della classificazione esegue un'azione sul set di dati.

Classificazione

Per classificare una nuova istanza, si calcola la distanza euclidea tra la nuova istanza e tutte le istanze del dataset. Si prendono le prime *k* istanze più vicine, e si prende la moda tra le classi di queste istanze, e questa è la classe predetta.

Naive Bayes Gaussiano

Teorema di bayes

Il teorema di bayes è una formula che permette di calcolare la probabilità di appartenenza di un punto ad una classe basandosi sulle probabilità di appartenenza di ogni punto del dataset.

Naive Bayes Gaussiano è un modello di classificazione generativo basato su una teoria che utilizza le distribuzioni gaussiane per rappresentare le caratteristiche di una classe. Questo modello parte dall'ipotesi che tutte le caratteristiche siano indipendenti tra loro, il che significa che l'influenza di una caratteristica sulla probabilità di appartenenza ad una classe non dipende dalle altre caratteristiche.

Il modello *Naive Bayes Gaussiano* si basa sulla formula di probabilità di Gauss per calcolare la probabilità che un dato esempio appartenga a una classe specifica. Questa formula prende in considerazione la media e la deviazione standard delle caratteristiche per una classe specifica. Una volta calcolate le probabilità per ogni classe, l'esempio viene assegnato alla classe con la probabilità più alta.

Questo modello è efficace per i dati. Tuttavia, l'ipotesi d'indipendenza delle caratteristiche può essere limitante in alcuni casi e può portare a risultati non accurati.

La probabilità di appartenenza di una feature della classe viene calcolata come segue:

$$p(x|y) = \frac{1}{\sqrt{2\pi\sigma}} * e^{-\frac{1}{2}*\frac{(x-mu)^2}{\sigma^2}}$$

dove:

- x è il valore della feature
- y è la classe
- mu è la media della feature

- sigma è la deviazione standard della features

Classificazione

Per classificare una nuova istanza, si calcola la probabilità di appartenenza ad ogni classe, e si prende la classe con la probabilità più alta.

ANALISI DATI & CONCLUSIONI

Per fare la cross-validation dei vari classificatori ci affidiamo ad una funzione presente in *sklearn.modelselection* ovvero *cross_val_score* in cui il numero di campioni ogni volta studiato è dieci, il risultato dell'accuratezza medie è riportato sui grafici.
cross_val_score è un metodo di sklearn che permette di valutare il modello utilizzando la cross validation. Questo metodo prende in input il modello, il dataset, il numero di fold e la metrica da utilizzare per valutare il modello. Questo metodo restituisce un array con i risultati di ogni fold.

Per la visualizzazione veloce e chiara dei risultati, è stata utilizzata la *confusion matrix* ovvero uno strumento per poter analizzare gli errori commessi dai vari classificatori. Ci permette di vedere in quali istanze risponde meglio o peggio il modello.
Nelle righe vengono riportati i dati reali, nelle colonne i dati predetti; di conseguenza nella diagonale principale avremo il numero di dati predetti correttamente, nelle altre celle, il numero di errori effettuati dalla macchina. Nel nostro caso, si verificheranno veri e falsi spam, e veri e falsi non-spam. Nella prima cella in alto a sinistra e in quella in basso a destra vengono riportati rispettivamente, i veri non-spam e spam. Nella cella in alto a destra vengono riportati i falsi negativi, ovvero i documenti spam che vengono classificati come non spam; nell'ultima cella vengono invece riportati i dati dei falsi positivi, quei file non-spam che vengono classificati come spam.

SVM Lineare

Sono stati effettuati 5 test con 3 valori differenti del parametro *C*, ovvero il parametro di regolarizzazione. Teoricamente scegliendo un valore elevato di C è come se forzassimo la SVM a classificare tutti i punti, mentre un basso valore di C ci permetterà di rinunciare più facilmente a molti dei punti in modo da ottenere un margine migliore.
All'aumentare di *C* si nota un aumento del tempo di esecuzione, ma non un corrispondente miglioramento della precisione.
Questa infatti rimane costante in un intorno di *0.898*.
Dalla confusion matrix si vede bene che non vi è una sensibile differenza tra i vari parametri

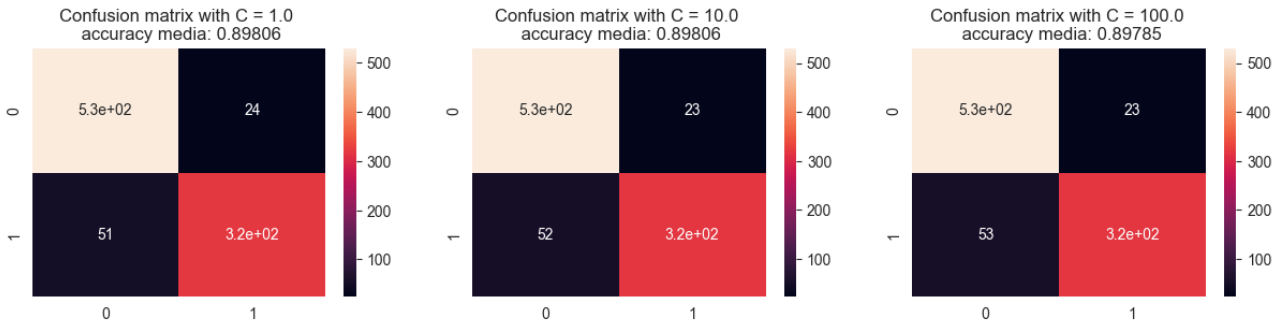

Figura 1: confusion matrix per SVM lineare.

La support vector machine lineare può essere meno efficace in caso di dati molto complessi o non lineari, in cui potrebbe essere necessario utilizzare tecniche di classificazione più sofisticate. Nel nostro caso però, i dati ottenuti sono buoni, per cui l'utilizzo di SVM lineare, con *C=1* può essere una scelta consona.
Il tempo medio di training in questo caso è di *1.0s* mentre per il tempo di predizione, è costante per i valori di *C*, si muove in un intorno di *0,04s*.
Ci siamo fermati a studiare il caso in cui C vale 100, in quanto con valori più grandi di cento, il tempo di training aumenta notevolmente, e la cross validation non viene completata in tempi ragionevoli.

SVM Polinomiale

Visti i tempi di training della SVM polinomiale, in particolare all'aumentare del paramento *C*, che risultano essere per nulla brevi, lo studio e la cross validation è stata effettuata su 6 parametri del valore *C,* in quanto i tempi di training per valori più grandi di quest'ultimo sono sopra i centoventi (nel caso migliore) secondi.

La SVM polinomiale è un modello di classificazione che si dimostra efficace nel riconoscimento di pattern complessi e non lineari nei dati. La scelta del grado del polinomio può influire significativamente sulla performance del modello; pertanto, è importante selezionare il grado ottimale attraverso metodi di validazione accurati.

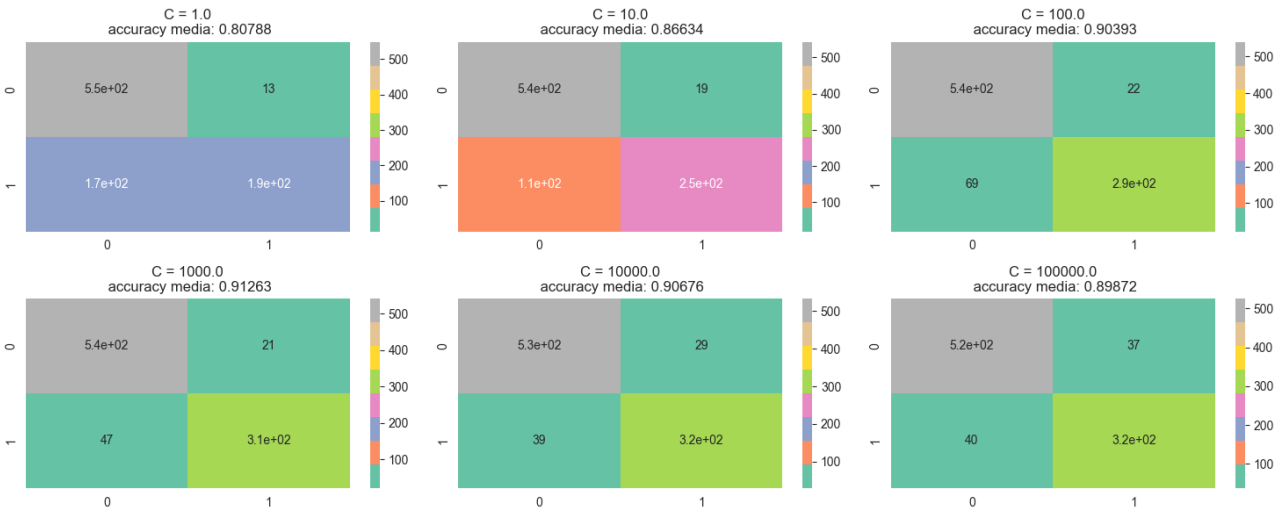

Figura 2: confusion matrix per SVM polinomiale.

Si nota chiaramente che all'aumentare del parametro *C* l'accuratezza del classificatore in questione cambia significativamente, dai dati nel notebook corrispondente vediamo che quando *C* vale *1*, l'accuratezza è appena a *0.808*, mentre con *C* pari a *1000*, l'accuratezza sale a *0.913*.
Tuttavia, la SVM polinomiale può anche presentare alcuni svantaggi, come il tempo di elaborazione più elevato rispetto ad altri modelli e la necessità di una quantità adeguata di dati per ottenere risultati accurati. Come si può vedere dai dati, un ottimo compromesso lo troviamo con *C* pari a mille, che ci consente di avere un tempo di training medio pari a *0.65s*.

SVM RBF

Per il caso del RBF abbiamo due parametri da testare, gamma e *C*, in quanto facciamo quindi un doppio studio dei parametri. Come prima cosa lasciamo settare gamma in automatico, e vediamo il miglior *C* che possiamo scegliere.

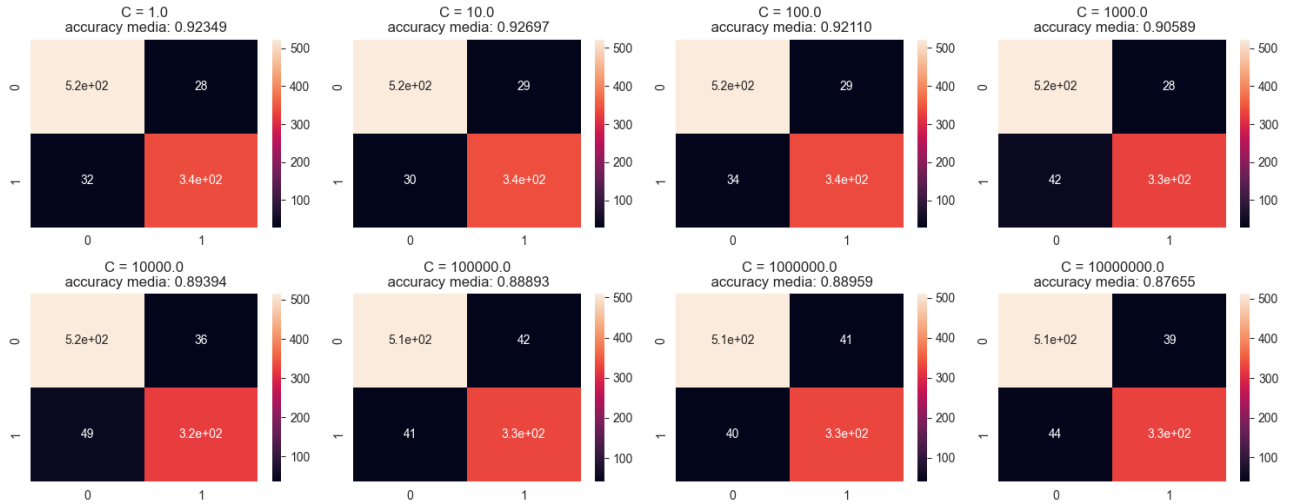

Figura 3: confusion matrix per SVM rbf con gamma automatico.

Si vede che con *C* pari a *10* otteniamo un'accuratezza pari *a 0.927*. Per cui scegliamo questo valore per lo studio di gamma.

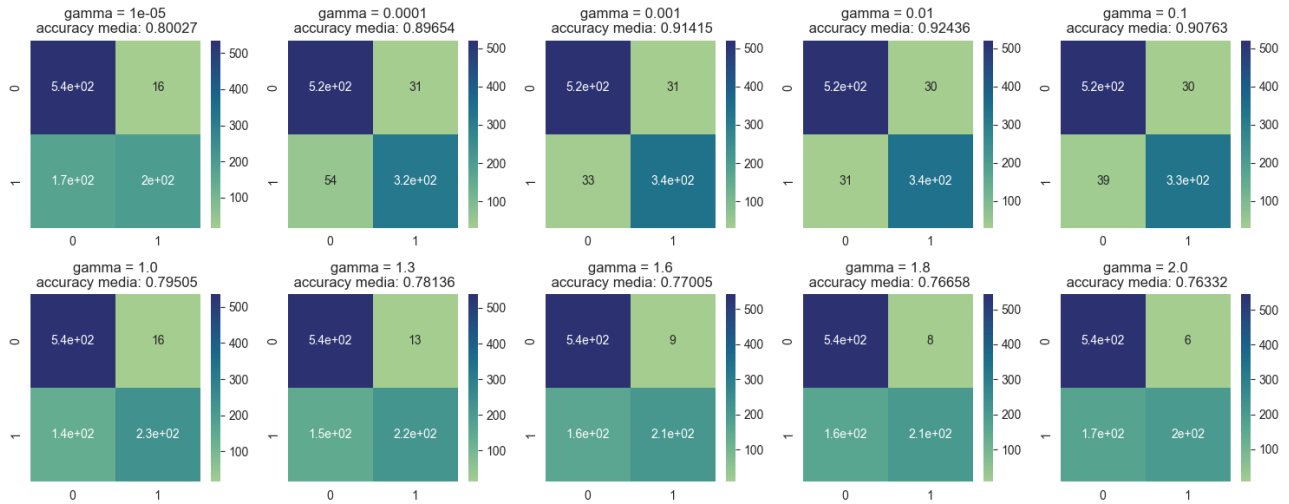

Figura 4: confusion matrix per SVM rbf in funzione di gamma, c fissato a 10.

L’accuratezza massima in questo caso (*0.925*), la si raggiunge quando gamma è settato a *0.01*.

È un potente algoritmo che può funzionare bene su una vasta gamma di problemi, ma non è sempre la scelta migliore per ogni situazione. Alcuni dei vantaggi dell'utilizzo di RBF SVM includono la sua capacità di gestire dati non lineari e la sua capacità di funzionare bene su dati ad alta dimensione.
Tuttavia, può essere costoso in senso computazionale e può richiedere un'attenta messa a punto dei parametri per ottenere buone prestazioni.
In conclusione, RBF SVM è uno strumento utile per l'apprendimento automatico, ma è importante considerare attentamente i suoi punti di forza e i suoi limiti.

SVM con Kernel Angolare

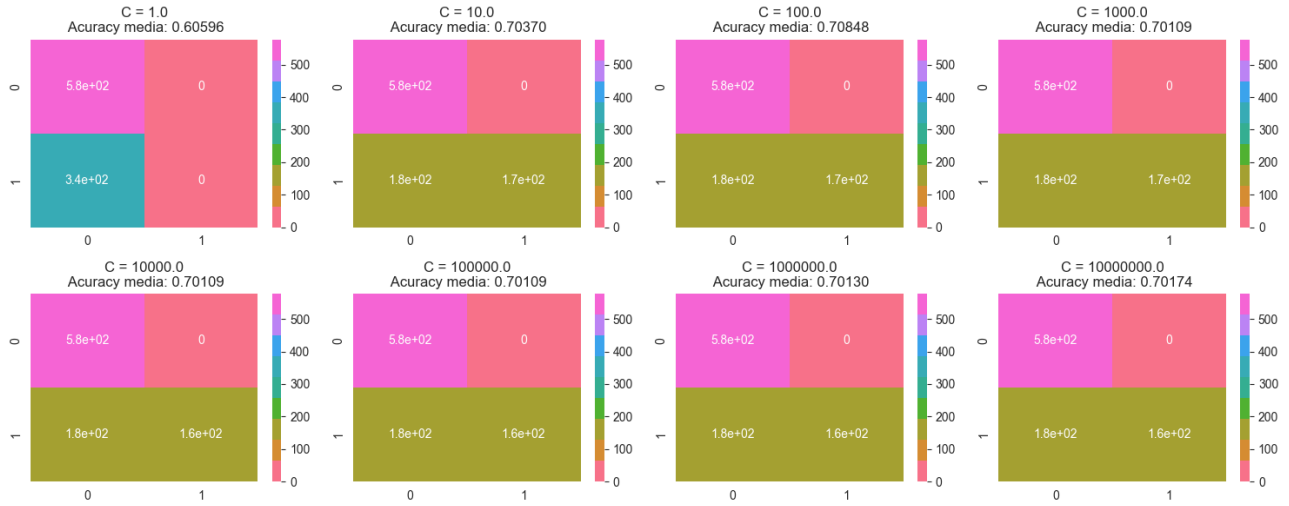

Figura 5: confusion matrix per SVM con kernel angolare.

I dati di accuratezza ottenuti utilizzando SVM con kernel angolare non sono particolarmente buoni, ma non sono nemmeno troppo cattivi. La media dell'accuratezza è in un intorno di *0.700*, che è un valore non troppo alto. Il tempo di esecuzioni non è particolarmente alto, siamo sempre in un intorno del mezzo secondo.
Questo modello è quindi adatto a problemi di classificazione non troppo complessi, ma non adatti a problemi di classificazione molto complessi.

Random Forest Classifier

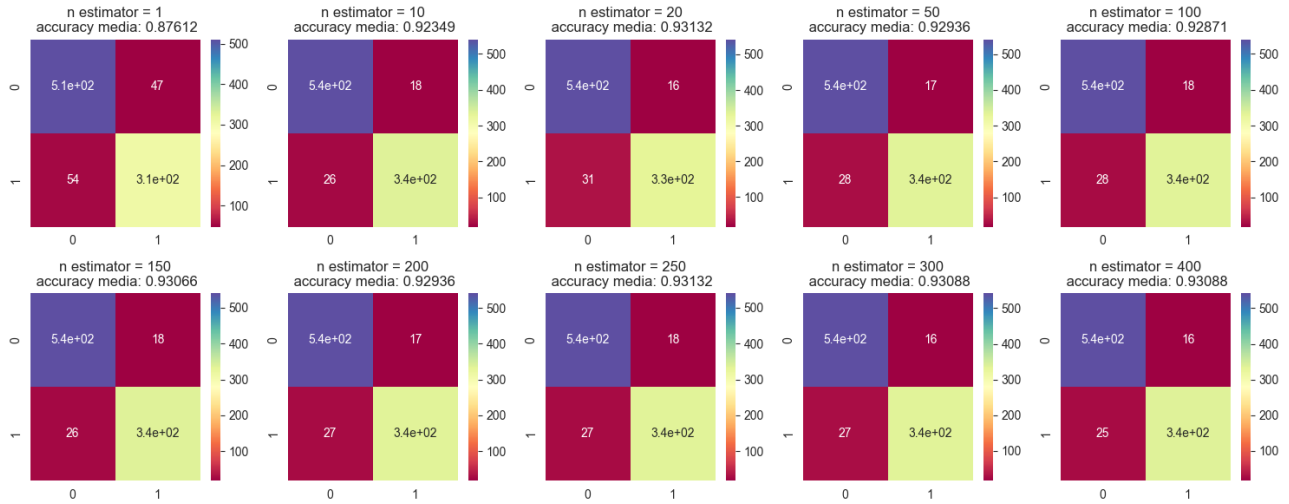

Figura 6: confusion matrix per Random Forest Classifier.

Le performance del random forest classifier sono molto buone, con un'accuratezza media di *0.930*. Il tempo di esecuzione è relativamente basso, nel caso peggiore siamo a tre secondi di tempo di training. L'accuratezza massima la troviamo quando settiamo il numero di stimatori a *250*, e quest'ultima risulta essere in media pari a *0.931*, un tempo di training pari a *2.4* secondi. Possiamo concludere che questo sia un ottimo classificatore.

Naive Bayes Gaussiano

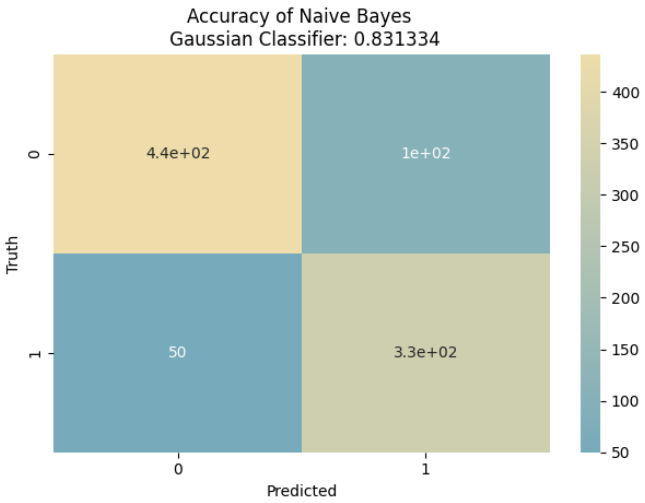

Figura 7: confusion matrix per Naive Bayes Gaussiano.

In quanto non ci sono parametri da testare per questo classificatore, procediamo direttamente con la cross validation eseguita, come negli altri casi, dieci volte.

L'accuratezza media del naive bayes gaussian è di 0.831, un valore non troppo alto, ma non troppo basso.
Il tempo di esecuzione è molto basso, nel caso peggiore siamo a 0.02 secondi di tempo di training. Al contrario il tempo di predizione è relativamente più alto, intorno ai due secondi.
Possiamo quindi dire che questo sia un buon classificatore.

K – Nearest Neightbors

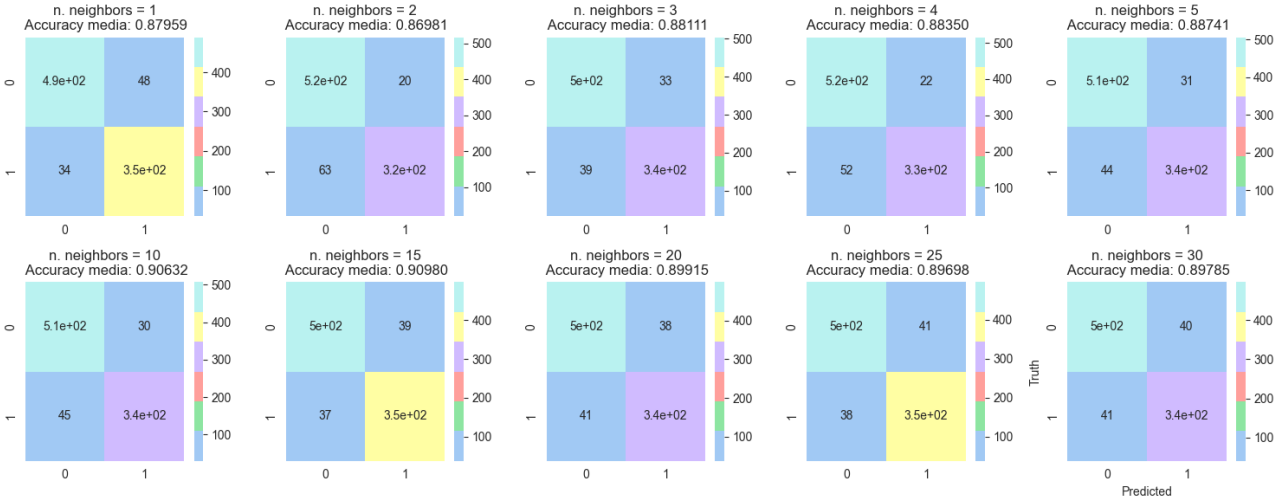

Figura 8: confusion matrix per KNN.

Le performance del K nearest neighbors sono molto buone, con un'accuratezza media di *0.900*. Il tempo di esecuzione è basso, nel caso peggiore siamo a *0.04* secondi di tempo di training.

L'accuratezza massima la troviamo quando settiamo il numero di vicini a *10* vicini, e quest'ultima risulta essere pari a *0.91*. Possiamo concludere che questo sia un buon classificatore.