Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Fisica

Segmentazione semantica automatica di immagini WSI per applicazioni in Patomica

Relatore:
Dott. Enrico Giampieri

Presentata da:
Annamaria D'Ambrosio

Anno Accademico 2019/2020

Alla mia famiglia

Sommario

Nella patologia tradizionale, i vetrini vengono preparati con un campione di tessuto del paziente e poi rivisti dal patologo con un microscopio ad alto ingrandimento. Successivamente viene delineata manualmente sull'immagine la Region of Interest (ROI). Obiettivo di questa fase è generare correttamente bordi di strutture anatomiche, in quanto una segmentazione sbagliata potrebbe condurre ad un'errata pianificazione della cura successiva.

La segmentazione è un processo che richiede molto tempo. Per ovviare a questo si ricorre alla cosidetta Patomica, ovvero la patologia digitale che ha lo scopo, mediante algoritmi computazionali, di estrarre automaticamente parametri quantitativi delle immagini istologiche, al fine di migliorare la velocità e l'accuratezza delle diagnosi. La Patomica ha anche lo scopo di individuare precocemente importanti malattie quali il tumore della pelle.

Il lavoro di questa tesi, dunque, nasce proprio con lo scopo di superare questi problemi e rendere più facile il lavoro del patologo dal punto di vista del tempo e dell'accuratezza nella determinazione di eventuali cure.

In particolare, oggetto di questo lavoro è lo sviluppo di modelli di segmentazione semantica automatica di immagini WSI di dermatologia prese da tessuti di melanoma raccolti in uno studio clinico del Policlinico Sant'Orsola di Bologna. La segmentazione è basata sull'algoritmo KNN con particolare attenzione alla ricerca e selezione di features che consentano di ottenere i migliori risultati.

Viene in oltre presentato un confronto con altri due modelli supervisionati (Support Vector Machine e Random Forest), allenati sulle stesse features del migliore modello KNN, per paragonarne le performance con quelle ottenute con il modello KNN.

L'utilizzo di modelli supervisionati per il task di segmentazione semantica di immagini WSI con risultati soddisfacenti potrebbe aprire la possibilità futura di utilizzare anche modelli non supervisionati permettendo così un ulteriore aumento dell'efficienza.

Indice

Introduzione

L'istopatologia moderna si basa fondamentalmente sull'accurata valutazione delle immagini microscopiche, con lo scopo di diagnosticare correttamente i pazienti e di trovare ad essi le giuste terapie. Negli ultimi anni, grazie allo svilupparsi delle tecniche di scansione e di elaborazione delle immagini, la disciplina dell'istologia ha guadagnato dei miglioramenti: in primis il passaggio dagli oculari del microscopio allo schermo del computer. Grazie a questi sviluppi ci sono stati diversi vantaggi, come la telepatologia, la condivisione in remoto per una rapida collaborazione, l'integrazione con flusso di lavoro clinici digitalizzati, la storia clinica del paziente e, soprattutto, la disponibilità all'utilizzo di applicazioni dell'intelligenza artificiale.

Whole slide imaging (WSI), che si basa sulla scansione di vetrini convenzionali per produrre diapositive digitali, è la tecnica di imaging più recente e utilizzata dai dipartimenti di patologia di tutto il mondo che continua a svilupparsi tra i patologi per scopi diagnostici, formativi e di ricerca.[1]

Questo processo include quattro fasi sequenziali: l'acquisizione dell'immagine (scansione), l'archiviazione, l'editing e la visualizzazione dell'immagine sullo schermo.

La scansione avviene catturando molte strisce successive di tessuto, la cui ampiezza dipende dall'obiettivo dello strumento; queste vengono poi assemblate per ricreare l'immagine digitale dell'intero vetrino.

Nel campo della patologia digitale, il concetto di ingrandimento e risoluzione va rivalutato nel contesto in cui le immagini sono acquisite e visualizzate. Questo fattore è solitamente indicato come il potere di ingrandimento delle lenti del microscopio utilizzate durante l'analisi. Dopo il processo di digitalizzazione, questo fattore di ingrandimento originale tende a cambiare, a seconda della risoluzione dello schermo di visualizzazione. Pertanto, la risoluzione, misurata in micrometri per pixel, è definita dal sensore ottico e dall'obiettivo usato per scansionare il vetrino.

La maggior parte degli scanner istologici sono generalmente dotati di obiettivi da 20x e 40x; con risoluzione di 0,5 mm per pixel e 0,25 mm per pixel. Quello più usato nelle valutazioni istopatologiche è l'obiettivo da 20x, considerato, tra l'altro, lo standard per le immagini scansionate, poiché c'è una buona conciliazione tra tempi di scansione e qualità delle immagini. La scansione dei vetrini con l'obiettivo da 40x può portare alla quadruplazione dei tempi di acquisizione e della dimensione del file finale, incrementando

i tempi di lavorazione di ogni vetrino e dei costi di archivio. Da tenere in considerazione che l'immagine di un singolo vetrino acquisito con WSI a 20x occupa circa 600 MB di memoria.

Nonostante la tecnica di WSI sia abbastanza sviluppata, fa ancora difficoltà a inserirsi nei laboratori istopatologici per la diagnosi primitiva. Ciò è dovuto principalmente ad alcuni svantaggi sia reali che percepiti. La diffidenza da parte dei patologi è rivolta soprattutto alla qualità delle immagini digitali: infatti, vi sono molti fattori, come la risoluzione, la compressione dell'immagine e soprattutto l'algoritmo di messa a fuoco, che giocano un ruolo chiave nell'interpretazione del preparato.

Un altro punto a sfavore riguarda la scansione dei vetrini che richiede molto tempo. Nonostante si hanno avuto miglioramenti tecnologici, il tempo medio per l'acquisizione di un vetrino è di circa 5/10 minuti, a seconda del numero di slice presenti nella slide, per un solo livello di ingrandimento. Nell'istologia tradizionale, invece, il patologo ha accesso a tutti i livelli di ingrandimento contemporaneamente. Il vero vantaggio, infatti, è che una volta acquisite, le immagini possono essere archiviate e consultate da remoto quasi subito.

In aggiunta, le immagini ora possono essere elaborate da algoritmi di intelligenza artificiale, favorendo l'applicazione come il Machine Learning che potrebbero migliorare il campo di ricerca, come è già avvenuto in diversi settori scientifici. Per poter elaborare automaticamente immagini di dimensioni così grandi, bisogna avere cura di ridurle in patch più piccole, la cui dimensione dovrebbe essere abbastanza grande da consentire l'interpretazione e preservare un certo grado di rappresentabilità dell'immagine originale.

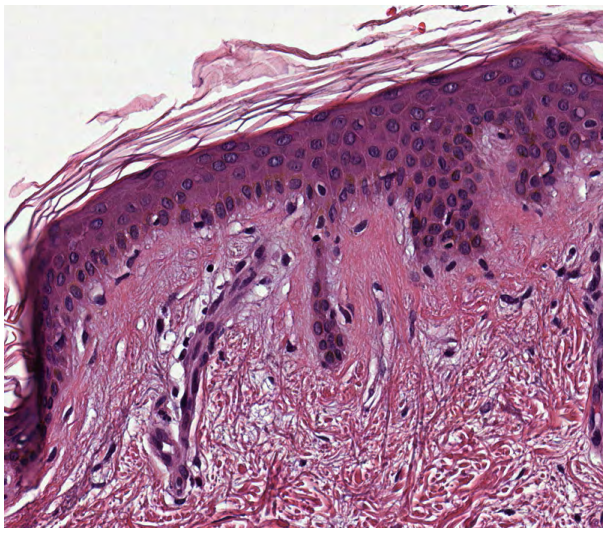


Figura 1: Pach di un campione di pelle

Come citato pocanzi per far si che la macchina sia in grado di elaborare immagini di dimensione abbastanza grande, come quelle ottenute con WSI, si ha la necessita di suddividerle in parti più piccole, le cosi dette pach. Queste dimensioni però devono contenere al contempo una quantità sufficiente di informazione, e devono mantenere un grado di rappresentatività dell'immagine.

Al contrario invece, se le pach sono molto piccole e si è concentrati più su un'unica regione, si perde l'informazione, e quindi nell'allenare la macchina il rischio è che l'algoritmo classifichi male e di conseguenza il patologo non sia in grado di dare una giusta diagnosi.

Esistono diverse tecniche che possono essere utilizzate nel riconoscimento degli oggetti.

La *Classificazione* consiste nell'attribuire ad ogni pixel dell'immagine sottoposta alla macchina, una classe che ha precedentemente imparato a conoscere. Per fare ciò si richiedono un sacco di dati da cui poter apprendere.

La *Segmentazione* invece è il processo in cui l'immagine è divisa in regioni distinte che hanno caratteristiche comuni.

Il risultato del processo di segmentazione è un'immagine con le stesse dimensioni di quella iniziale composta da regioni di colore pieno, che rappresentano gli oggetti rilevati. Questa immagine è chiamata maschera di segmentazione. Un esempio è riportato in Fig.2.

Figura 2: Maschera di segmentazione.

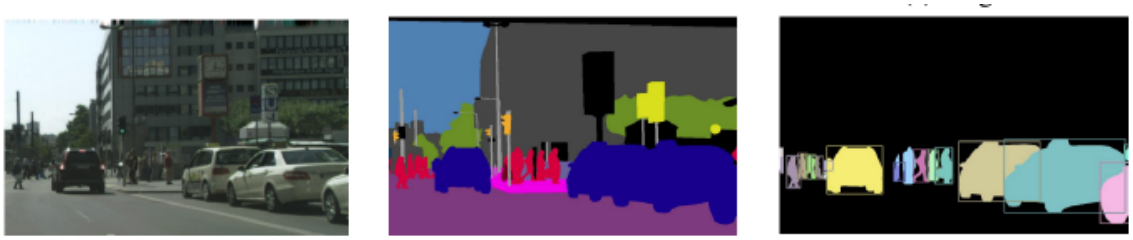
Queste segmentazioni possono essere utilizzati per eseguire l'estrazione delle caratteristiche, che fornisce informazioni fondamentali sugli organi o sui volumi delle lesioni, sul

conteggio delle cellule, ecc.

La segmentazione delle immagini gioca un ruolo cruciale in molte applicazioni di imaging medico automatizzando o facilitando la delineazione di strutture anatomiche e altre regioni di interesse [2].

La segmentazione manuale è possibile, ma richiede tempo ed è soggetta alla variabilità dell'operatore, rendendo i risultati difficili da riprodurre [3]. Pertanto sono preferibili metodi automatici o semiautomatici.

Il processo di segmentazione si divide in *segmentazione semantica* dove la macchina attribuisce ad ogni pixel dell'immagine una classe ben specifica e *instance segmentation* che riconosce i vari elementi che compongono un'immagine, e quindi i contorni dei singoli oggetti, attribuendone poi una la classe di appartenenza.



(a) *Image* (b) *Semantic segmentation* (c) *Instance segmentation*

Figura 3: a) Per una data immagine(a), si mostra:b) Semantic segmentation, c) Instance segmentation

Dunque l'analisi delle immagini istologiche consiste in genere nel rilevare le diverse componenti nei campioni e nell'individuarne eventuali cellule anomale e stabilirne il tipo. Inoltre, i campioni prelevati da diverse parti del corpo umano presentano caratteristiche completamente differenti, e questo aumenta la complessità dell'analisi. Un esame accurato di un campione richiede quindi un'attenta ispezione effettuata da un esperto altamente qualificato. L'automatizzazione, in questo caso risulta di ottimo aiuto, dando un vantaggio sia nei tempi che nell'accessibilità.

Capitolo 1

Background Teorico

1.1 Il machine learning

L'apprendimento automatico, noto più comunemente come machine learning, è una sottocategoria dell'intelligenza artificiale. Esso viene applicato ad una varietà di campi: robotica, riconoscimento di modelli, diagnosi medica ecc.

Si usa per modellare dei fenomeni al fine di fare classificazioni o regressioni a partire da dati raccolti da istanze osservate in tali fenomeni.

Il vantaggio del machine learning rispetto ai sistemi tradizionali basati su regole predeterminate sta nel fatto che quante più volte viene "allenato" con maggiore quantità e diversità di dati, tanto più viene incrementata la sua performance.[4]

Un aspetto molto positivo è che non bisogna creare algoritmi specifici per ogni task, ma si sfruttano algoritmi generici che imparano automaticamente ad adattarsi allo svolgimento di un task specifico.

Il machine learning viene preferito rispetto ai sistemi "tradizionali" in quanto, pur non dando un output certo, consente di migliorarsi nel tempo garantendo un risultato più vicino alla realtà.

L'implementazione di un processo di machine learning si può dividere in tre fasi principali: una fase in cui la macchina viene allenata, sottoponendola a una serie di dati (training), una fase di validazione dove un set di dati (validation set) serve a fare delle misure per verificare l'efficacia dei modelli, e una fase di test in cui i risultati vengono valutati in termini di efficacia nello svolgimento del task su istanze indipendenti da quelle usate per il training.

L'algoritmo allenato, unitamente ai dati e parametri che si andranno ad utilizzare nella ricerca, viene definito modello. È possibile suddividere gli algoritmi di machine learning in *supervisionati* e *non-supervisionati*. Nel machine learning supervisionato

l'algoritmo apprende da un set di dati già classificato e corredato da esempi dell'output desiderato.

Le tecniche di apprendimento supervisionato possono essere due:

-la *classificazione* dove lo scopo è quello di riuscire a prevedere l'etichettatura delle classi di dati futuri. Nella fig.1.1 si ha un dataset bidimensionale , utilizzando un algoritmo supervisionato è possibile separare le due classi da un confine decisionale e associare dunque ad ogni campione la rispettiva classe.

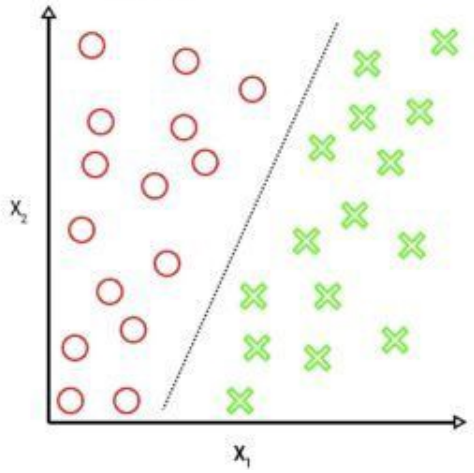


Figura 1.1: Classificazione binaria.

- la *regressione* invece il risultato è un valore continuo, alla macchina spetta il compito di trovare una relazione tra valore di input e output.

Mentre in quello non-supervisionato abbiamo dati senza etichetta e quindi l'algoritmo non cerca di apprendere una regola da un esempio pratico ma cerca di elaborarla da zero.

Le principali tecniche dell'apprendimento non-supervisionato sono:

-il *clustering* cioè implica una formazione basata su dati non classificati e con i quali l'algoritmo genera autonomamente un modello di classificazione in base alle caratteristiche comuni.

-la *riduzione* della dimensionalità dei dati in cui l'algoritmo elimina i dati di poca importanza e combina le informazioni emergenti (correlate) per concentrare l'analisi su quelli in cui si può estrarre uno schema.[5]

C'è un ulterione apprendimento dove volgere una particolare attenzione che è quello *semi-supervisionato*. Tipicamente utilizzato nelle applicazioni biomediche. Questa tecnica

di apprendimento è una combinazione tra l'allenamaneto supervisionato e quello non-supervisionato, nel quale durante l'addestramento combina una piccola quantità di dati etichettati con una grande quantità di dati non etichettati. In questo modo si possono ridurre i costi necessari alla preparazione dei dati iniziali e velocizzare il processo.

La fusione tra i dati può produrre un notevole miglioramento nell'accuratezza dell'apprendimento e portare a risultati migliori rispetto alle tecniche pure non supervisionate.

La situazione tipica di utilizzo di questa tecnica è quando l'acquisizione di dati etichettati richiede l'ausilio di un essere umano altamente qualificato (come anatomopatologo). Il costo effettivo per costruire interi e idonei set di formazione completamente etichettati in queste situazioni sarebbe insopportabile e l'apprendimento semi-supervisionato è di grande aiuto pratico.

Nei paragrafi successivi andiamo a descrivere nel dettaglio gli algoritmi supervisionati cha abbiamo utilizzato in questo lavoro di tesi. E sono rispettivamente il K-Nearest Neighbours (KNN), il Support Vector Machine (SVM) e il Random Forest (RF).

1.1.1 K-Nearest Neighbours

La logica alla base del KNN consiste nel riconoscimento dei pattern basato sulla vicinanza dei dati. Ad esempio se prendiamo un oggetto A che ha caratteristiche simile ad un oggetto B, ed è posto nelle sue immediate vicinanze, possiamo supporre che A appartenga alla stessa classe di B. Un esempio è riportato nella figura 1.2 qui di seguito.

L'elemento fondamentale in questo modello è la definizione del parametro K ovvero il numero di "vicini" all'oggetto che si sta studiando.

Un K troppo ridotto (es.=1) rischia di rendere l'algoritmo "cieco" in quanto stiamo limitando la sua regione di osservazione. Aumentando progressivamente il valore di K il modello diventa più stabile e le previsioni più accurate in quanto si considerano più "vicini". È importante comunque che il valore K non sia eccessivamente grande pena l'invalidità del modello, in effetti se k è molto grande si corre il rischio di ignorare piccoli dettagli che potrebbero essere rilevanti.

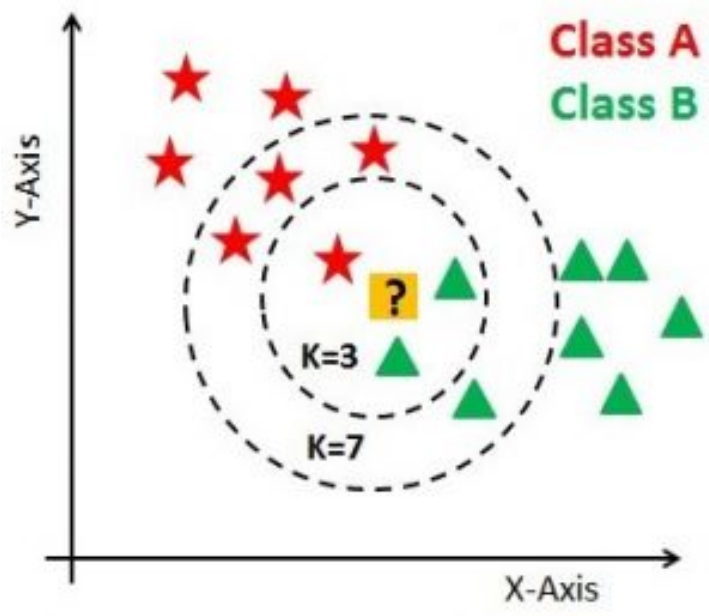


Figura 1.2: In figura vediamo come la scelta del valore k influisca sulla classificazione.

Si noti nell'immagine sopra che la maggior parte delle volte, punti simili sono vicini l'uno all'altro. L'algoritmo KNN dipende da questo presupposto, e riesce ad unire il concetto di somiglianza (a volte chiamata distanza, prossimità o vicinanza) degli elementi più vicini e, a maggioranza, classificare l'elemento. calcolando la distanza tra i punti su un grafico.

La somiglianza viene definita in base a una metrica di distanza tra due punti di dati. Una scelta popolare è la distanza euclidea.

Per selezionare la K giusta per i dati, eseguiamo l'algoritmo KNN più volte con diversi valori di K e scegliamo la K che riduce il numero di errori, mantenendo allo stesso tempo la capacità dell'algoritmo di fare previsioni accurate quando gli vengono forniti dati che non ha mai visto prima. Ecco le cose da tenere a mente:

Man mano che riduciamo il valore di K a 1, le previsioni diventano meno stabili.

Al contrario, aumentando il valore di K, le previsioni diventano più stabili a causa del voto a maggioranza / media e, quindi, più propense a fare previsioni più accurate (fino a un certo punto). Solo alla fine, quando si iniziano ad osservare un numero importante di errori al quel punto sappiamo di aver spinto troppo oltre il valore di K.

1.1.2 Support Vector Machine

Il SVM (Support Vector Machine) è un altro tipo di classificatore supervisionato. In principio esso nasce come classificatore di tipo binario.

L'idea che sta alla base di questo modello è che l'algoritmo crea un iperpiano che divide lo spazio tra le due classi. Dunque date due classi linearmente separabili e un training set che contiene n campioni, dati x_i pattern multidimensionali e y_i le etichette delle due classi che possono assumere valore +1 e -1.

L'iperpiano è rappresentato dalla funzione:

$$D(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + b \tag{1.1}$$

dove $\mathbf{W}$ è il vettore normale all'iperpiano e indica il peso, $\mathbf{x}$ è il vettore di caratteristiche in input e b l'intercetta all'origine che indica il bias.

Ma ci possono essere una moltitudine di iperpiani che possono eseguire la separazione. Gli iperpiani che separano i pattern nel training set con distanza minima $1/\ ||\mathbf{W}||$ su ogni lato soddisfano, per i= 1...n, le equazioni :

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 & se \quad y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & se \quad y_i = -1 \end{cases} \tag{1.2}$$

Chiamato margine, la minima distanza tra l'iperpiano di separazione e un pattern del training set, il valore che esso assume è $\tau = 2/\ ||\mathbf{W}||$.

I pattern che giacciono sul margine sono detti support vector.

Qui di seguito vediamo la figura 1.3 che ci illustra meglio questo metodo:

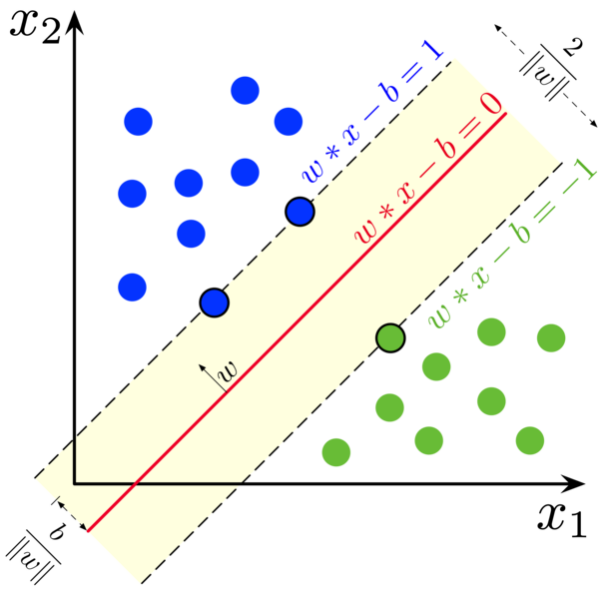


Figura 1.3: SVM addestrato con campioni di due classi, si noti l'iperpiano e i margini a margine massimo.

L'iperpiano ottimo secondo SVM è quello soddisfa i vincoli di separazione dei pattern e massimizza il margine τ.

Nel caso multiclasse invece ci sono alcuni metodi per applicare il SVM al caso multiclasse. Tra questi c'è il metodo *one-against-all* (uno contro tutti), ovvero si costruiscono n classificatori dove n corrisponde al numero delle classi totali che abbiamo. Supponiamo di prendere l'i-esimo classificatore binario che separa i vettori della classe i-esima a tutte gli altri vettori delle altre classi. Quindi si addestra considerando gli elementi della classe i-esima positivi e tutti gli altri negativi. Si procede in questo modo su ogni i-esimo classificatore ($i = 1,\ldots,$n), calcolando di volta in volta D($\mathbf{x}$). Al termine del training si assegna il pattern $\mathbf{x}$ alla classe n-esima per cui è massima la distanza dalla superficie decisionale.[6]

1.1.3 Random Forest

Nell'intelligenza artificiale il Random Forest è un classificatore supervisionato che fa un "ensemble" di alberi decisionali, essendo questi ultimi dei classificatori a se che individualmente potrebbero essere non accurati nella previsione, ma che combinati assieme, possono produrre previsioni più accurate.

L'albero decisionale è composto da nodi interni che rappresentano le caratteristiche, Da questo nodo dipartono tanti rami quanti sono i possibili valori che la caratteristica può assumere, fino a raggiungere le foglie che indicano la categoria associata alla decisione. Il vantaggio nell'utilizzare l'albero di decisione è che è molto semplice da leggere.

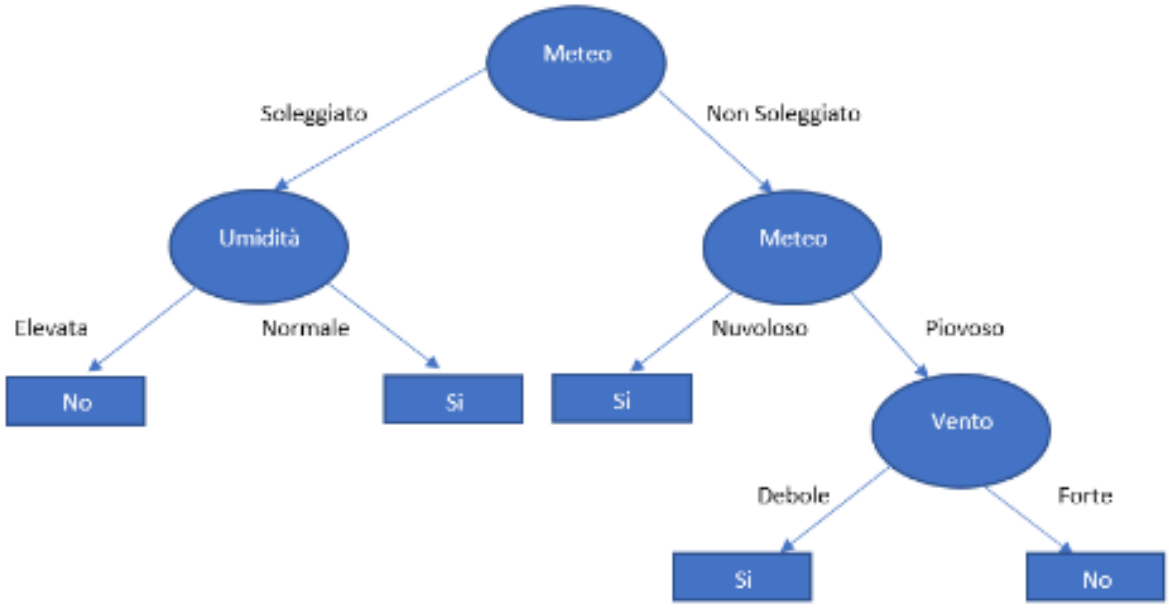


Figura 1.4: Questa in figura è un esempio di albero decisionale. Ad.es. se lo scopo è quello di uscire a fare una passeggiata o no, l'obbiettivo del problema è prevedere se si farà in base a determinate caratteristiche.

Dopo aver stabilito il numero di alberi decisionali da utilizzare si selezionano casualmente un insieme di esempi dal data set che genererà un sottoinsieme che si andrà ad utilizzare come training per un albero decisionale. Successivamente si fa un ulteriore scelta casuale del sottoinsieme di esempi (Fig.1.3) e questo porta alla generazione di un

nuovo albero, fino a quando non si raggiunge il numero di alberi che compongono la foresta.

La predizione finale si otterrà tenendo in considerazione il voto di ogni singolo albero decisionale.

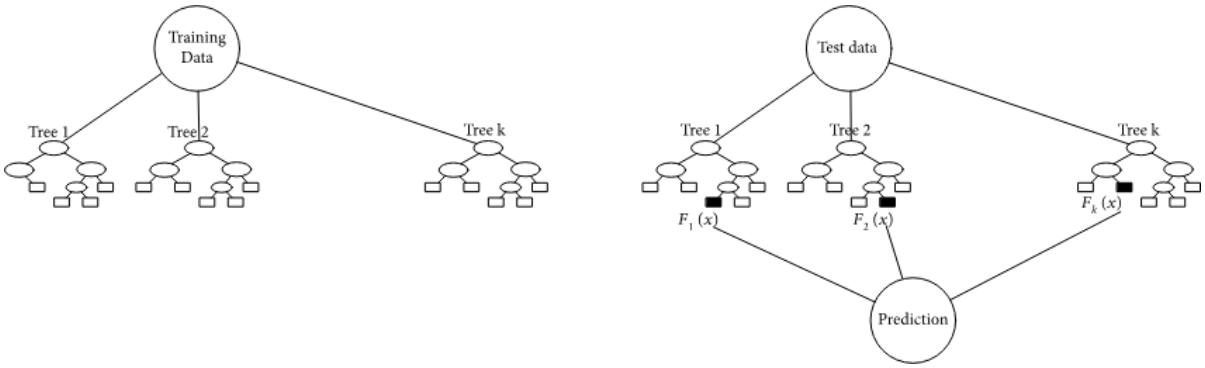


Figura 1.5: Un tipico esempio di Random Forest.

Dunque maggiore è il numero di alberi, migliore sarà il modello.

Il limite di questo classificatore è che più si aumenta l'inserimento di alberi decisionali e maggiore sarà lo sforzo computazionale.[7]

1.2 Filtri immagine

In questo lavoro abbiamo usato vari filtri per generare le features da utilizzare nel training dei modelli di machine learning.

Per ottenere un modello per la segmentazione automatica delle immagini eseguiamo delle operazioni di pre-processing alle immagini del nostro dataset che consiste in operare ridimensionamenti, conversioni in diversi spazi colore e nell'applicazione di filtri. I filtri di immagine possono essere utilizzati per ridurre la quantità di rumore nell'immagine oggetto di studio, migliorando così la precisione dei modelli di machine learning.

L'applicazione di un filtro consiste nell'applicare una particolare matrice (kernel) all'immagine. Nello specifico un kernel è una matrice quadrata nxn dove "n" è un numero dispari in quanto nella convoluzione (sovrapposizione di matrici) è fondamentale identificare il centro della matrice kernel. Andiamo a descrivere brevemente i filtri che sono stati utilizzati in questo lavoro di tesi. [8]

1.2.1 Filtro mediano

Il filtro mediano è un filtro non lineare che viene spesso utilizzato per rimuovere il rumore da un'immagine preservandone però i contorni. Il filtro mediano calcola la mediana delle intensità dei pixel che circondano il pixel centrale in un kernel nxn. La mediana risultante va a sostituire il valore originale del pixel centrale. Un esempio di calcolo è riportato nella figura seguente:

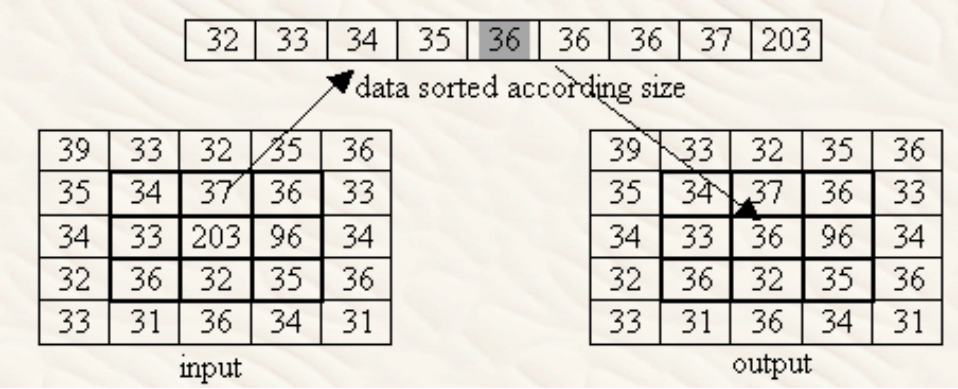


Figura 1.6: Calcolo della mediana[9]

Sotto è riportato un esempio di una immagine dopo che è stato applicato il filtro mediano.

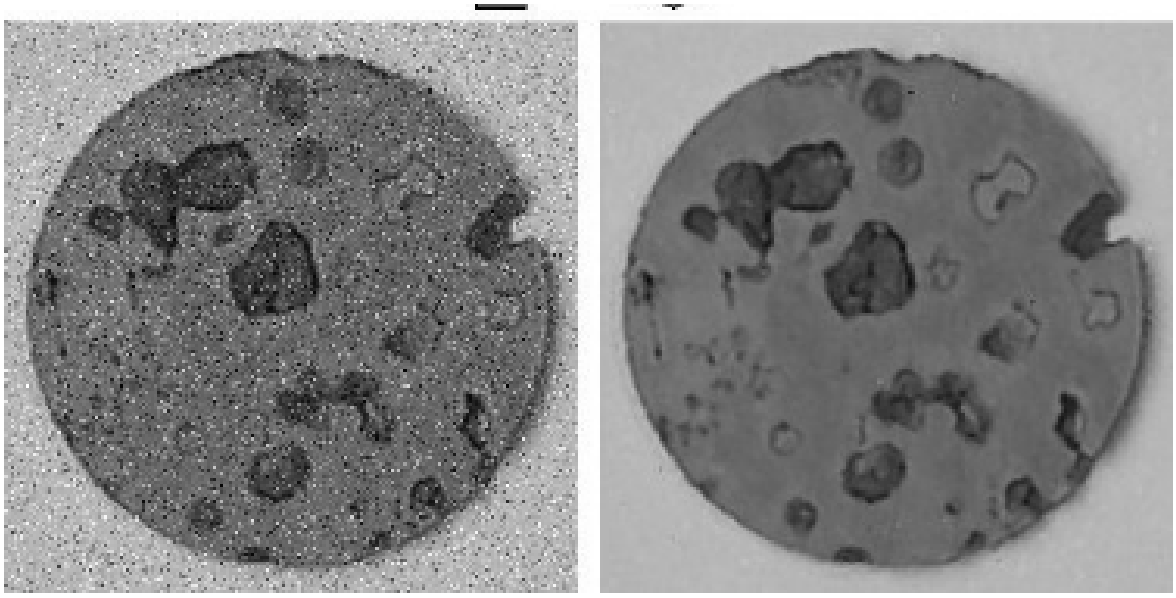


Figura 1.7: Il risultato dell'applicazione di un filtro mediano 3x3.[9]

1.2.2 Filtro di varianza

Il filtro di varianza è un filtro che serve ad evidenziare zone di colore omogeneo o con noise.

Come calcolo il filtro di varianza è simile al filtro mediano, con l'alternativa che calcola la varianza delle intensità dei pixel che circondano il pixel centrale in un kernel nxn. La varianza risultante va a sostituire il valore originale del pixel centrale.

Nella figura sottostante è riportato un esempio di immagine dopo avergli applicato il filtro di varianza.

Figura 1.8: a) immagine originale; (b) risultato dopo l'applicazione del filtro di varianza.[19]

1.2.3 Filtro edge detection (Operatore di Sobel)

Generalmente la prima cosa che si cerca di riconoscere su un immagine sono i bordi. Un bordo infatti segna il confine tra un oggetto e un altro. Un bordo è costituito da pixel con le variazioni di intensità dei toni di grigio che sono diversi dai pixel vicini.

Per far ciò ricorre di aiuto il cosi detto filtro di Edge Detection che utilizza l'operatore gradiente.

I filtri di edge di base utilizzano una sola direzione per rilevare i bordi (orizzontalmente o verticalmente), se invece si vogliono rilevare i contorni in tutte le direzioni, si combinano più operatori insime. Qui entro in gioco l'operatore di Sobel dove calcola pixel per pixel un valore approssimato del gradiente di una funzione trovando la direzione lungo la quale si ha il massimo incremento possibile dal chiaro allo scuro, e la velocità con cui avviene il cambiamento lungo quella direzione.[10]

Il risultato ottenuto fornisce una misura di quanto "bruscamente" l'immagine varia in quel punto, e quindi della probabilità che quella parte di immagine rappresenti un contorno.

L'algoritmo utilizzato dall'operatore di Sobel applica due kernel 3x3 all'immagine originaria, una in direzione orizzontale e l'altra in direzione verticale. Se chiamiamo ripettivamente G_x e G_y le immagini in cui i punti rappresentano i valori approssimati delle derivate in orizzontale ed in verticale. Il valore totale del gradiente risulta essere:

$$G = \sqrt{(G_x^2) + (G_y^2)} \tag{1.3}$$

Da questi risultati si ottiene anche la direzione del gradiente:

$$\Theta = arctang(G_y/G_x) \tag{1.4}$$

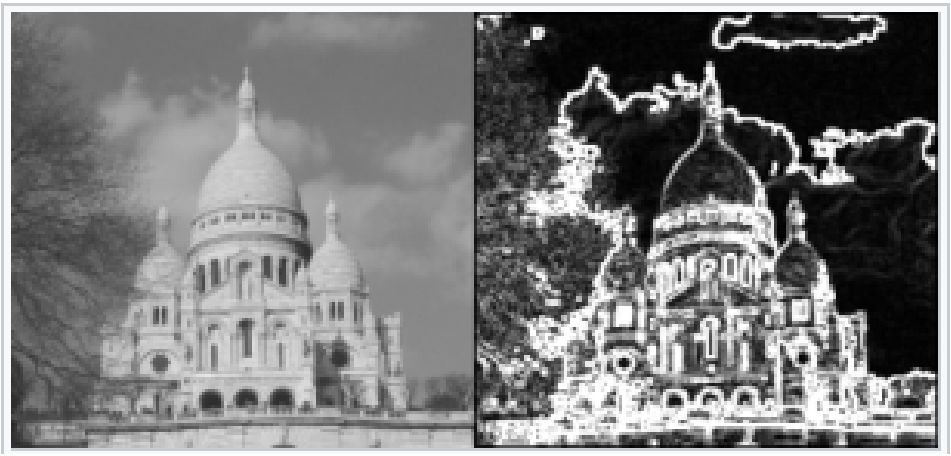


Figura 1.9: Applicazione di un filtro di Sobel[10]

1.2.4 Filtro di entropia

Il principio del filtro entropico è quello di rilevare difetti o danneggiamento di una immagine.

Spesso una immagine presenta dei difetti, e spesso il rumore presente nelle immagine può ingannare la presenza di essi. Per ridurre dunque questo rumone si ricorre al filtro entropico.

Questa funzione riceve in ingresso un'immagine in scala di grigi e restituisce un'immagine in cui le intensità luminose dei pixel sono i corrispondenti valori di entropia.[11]

Il filtro calcola l'entropia per ciascuna area dell'immagine in ingresso che scorre sull'intera immagine.

L'espressione utilizzata per il calcolo dell'entropia S è quella classica:

$$S = -K_b \sum P_i ln P_i \tag{1.5}$$

dove k_b è la costante di Blotzman e P_i è la probabilità associata alla configurazione di un sistema in uno specifico microstato.[12].

Nell'analisi dell'immagine P_i è la distribuzione di un i-esimo colore all'interno di una sezione di una immagine e quindi associa le variazioni delle "configurazioni di colore" all'interno di un "sistema immagine" come caoticità di quest'ultimo.[11]

Qui di seguito si riporta una immagine con filtro di entropia:

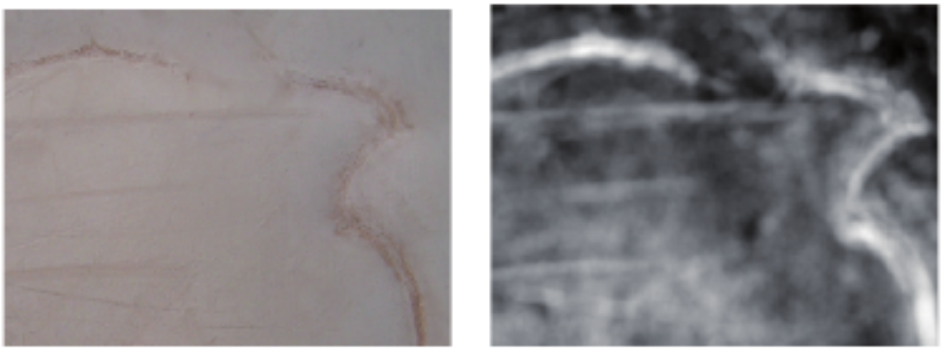


Figura 1.10: A destra immagine di sezione di cuoio con presenza di marchio a fuoco, a sinistra immagine in scala di grigi a seguito del filtro entropico.

1.3 Spazio di colore

Uno spazio colore è un'organizzazione specifica di colori sulla base di variabili numeriche. Tramite queste variabili si vuole ottenere la rappresentazione spaziale dei colori su assi cartesiani. Qui di seguito riportiamo gli spazi dai quali abbiamo estratto le features.

1.3.1 RGB

Lo spazio RGB è uno spazio colore in 3D tra i più utilizzati. Può essere visualizzato come un cubo ai cui angoli troviamo il nero, i tre colori primari (verde, rosso e blu), i tre colori secondari (ciano, magenta e giallo) ed il bianco.[12]I colori all'interno del cubo sono ottenuti sommando l'intensità di ogni canale rgb. In ambito digitale, i componenti rosso, verde e blu sono solitamente definiti con numeri compresi tra 0 e 255. Questo spazio è il più usato perchè è stato il primo modello che ha dato una descrizione adeguata di come funziona l'occhio umano, ed è in grado di replicare in maniera estremamente facile una grandissima varietà di colori visibili all'occhio umano.

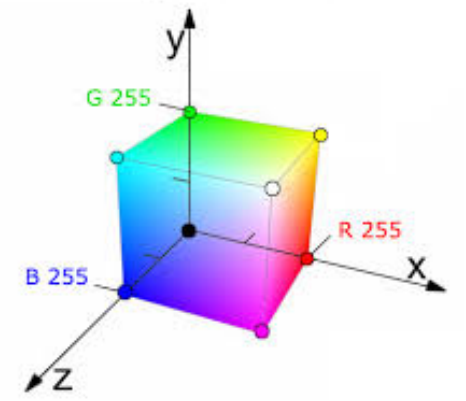


Figura 1.11: Illustrazione dello spazio colore RGB

1.3.2 HSV

Il modello HSV (Hue, Saturation, Value) è uno spazio di colore solitamente rappresentato da un cono esagonale.

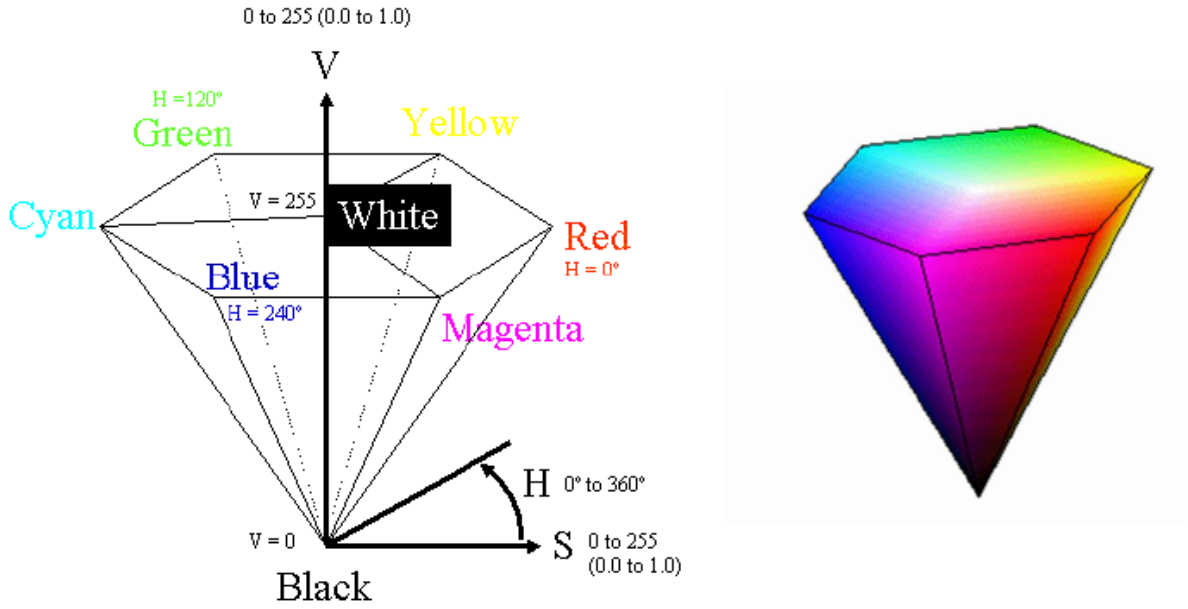


Figura 1.12: Illustrazione dello spazio colore hsv

Qui il colore è rappresentato come combinazione di:

-La *saturazione* rappresenta la vividezza del colore: ad esempio se il colore ha una saturazione massima allora significa che è estremamente intenso, mentre un colore con saturazione nulla è tendente alla scala di grigi.

-La *luminosità* è intesa come la quantità di nero nel colore. Una luminosità massima comporta un colore tendente al bianco mentre una luminosità nulla comporta un colore totalmente nero.

-La *tonalità* è descritta da un numero compreso tra 0° e 360° ed è misurata da un angolo intorno all'asse verticale dell'ipotetico cono esagonale (come si vede in figura) e indica il colore così come definito nello spettro dei colori: rosso, giallo, verde, ecc.

1.3.3 LAB

Un altro spazio colore è il modello Lab (CIE L*a*b*) che, rispetto ai modelli trattati precedentemente, è un modello basato più sulla percezione umana del colore. Nello specifico, abbiamo la Luminosità (Luminance) ed i valori a e b che identificano le componenti cromatiche che rappresentano il rosso/verde e il blu/giallo. I valori della luminosità vanno da una scala da 0 (scuro) a 100 (chiaro), la componente a assume valori tra -120 (rosso) a +120 (verde) mentre la componente b varia da +120 (giallo) a -120 (blu).

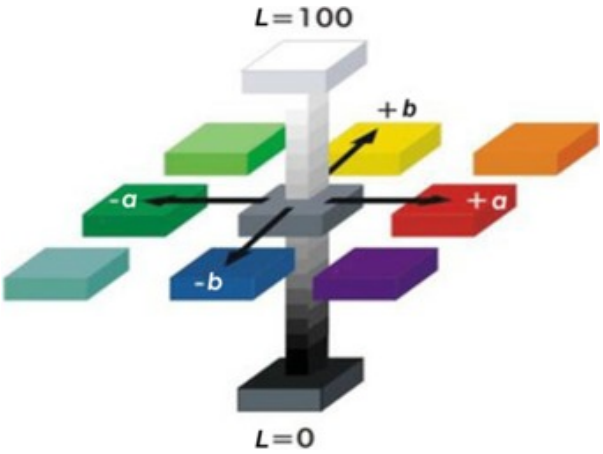


Figura 1.13: Illustrazione dello spazio colore LAB

1.3.4 HED

Questo spazio di colore verrà un po più approfondito dato che è quello che in questo lavoro di tesi risulta l'aver generato features migliori. HED come acronimo sta per Haematoxylin-Eosin-DAB che è la colorazione usata dai medici per analizzare tessuti, in questo particolar modo, i campioni di pelle.

L'ematossilina (blu) colora principalmente i nuclei delle cellule e l'eosina (rosso magenta) è utile nella colorazione del citoplasma. Un'altra tinta molto usata è l'uso dell'immunoistochimica con colorazione perossidasi di rafano sviluppata con DiAminoBenzidina (DAB, marrone), che va ad evidenziare la melatonina.

L'obiettivo nell' usare queste differenti colorazioni sta nel fornire la distribuzione di queste sostanze e quindi individuare le strutture a cui la colorazione va ad attaccarsi, dato che ognuna di essa ha il compito di evidenziare una determinata zona di tessuto. La quantità di colorazione depositata determinerà quindi la densità ottica a lunghezze d'onda specifiche della colorazione secondo la legge di Lambert-Beer (1), dove la densità ottica è direttamente proporzionale alla concentrazione del colore.

Queste tinte però hanno complessi spettri di assorbimento sovrapposti, e per esaminare contemporaneamente la fotometria e le caratteristiche morfometriche di una o più strutture, i relativi contributi dei coloranti allo spettro di assorbimento risultante, devono essere separati. Per ovviare a ciò ci sono vari metodi, ma in questo lavoro di tesi non ce ne occuperemo. Accenniamo solo un piccolo esempio su come funziona in uno spazio di colore RGB.

Ad esempio, supponendo che si utilizza un sistema di microscopia video con una telecamera RGB, ogni colorazione pura sarà caratterizzata da uno specifico fattore di assorbimento c per la luce in ciascuno dei tre canali RGB.

Le intensità rilevate della luce trasmessa attraverso un campione e la quantità (A) di colorazione con fattore di assorbimento c sono descritte dalla legge li Lambert-Beer:

$$I_C = I_{O,C} e^{-Ac_c} \tag{1.6}$$

con $I_{o,C}$ intensità della luce che entra nel campione, I_C l'intensità della luce rilevata dopo il passaggio del campione e il pedice c che indica il canale di rilevamento. Nel modello RGB, le intensità I_R, I_G, I_B sono ottenute dalla telecamera per ogni pixel. Perché l'intensità relativa in ciascuno dei canali dipende dalla concentrazione di macchia in modo non lineare[11], i valori di intensità dell'immagine non possono essere utilizzati direttamente per separare e misurare ciascuna macchia. Tuttavia la densità ottica per ogni canale può essere definita come:

$$OD_C = -log_{10}(I_C/I_{O,C}) = A * C_C \tag{1.7}$$

Come si può vedere, l'OD per ogni canale è lineare con la concentrazione di materiale assorbente, e può quindi essere utilizzato per la separazione del contributo di più coloranti in un campione.

Ogni colorante puro sarà caratterizzato da una densità ottica specifica per la luce in ciascuno dei tre canali RGB, che può essere rappresentata da un vettore OD 3 per 1 che descrive la macchia nello spazio colore RGB convertito in OD.

Ad esempio, misurazioni di un campione colorato con solo ematossilina ha prodotto valori di OD pari a 0,18, 0,20 e 0,08 per i canali R, G e B rispettivamente.

La lunghezza del vettore sarà proporzionale alla quantità di colorante, mentre i valori relativi dal vettore descrivono l'effettiva OD per i canali di rilevamento. Nel caso di tre canali, il sistema di colore può essere descritto come una matrice nella forma:

$$\begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix} \tag{1.8}$$

Con ogni riga che rappresenta un colore specifico e ogni colonna che rappresenta la densità ottica rilevata dal canale rosso, verde e blu per ogni colorazione.

I valori specifici del colore per l'OD in ciascuno dei tre canali possono essere facilmente determinati misurando l'assorbimento relativo di rosso, verde e blu sui vetrini colorati con una singola colorazione.

La matrice OD per la combinazione di ematossilina, eosina e DAB[13] è:

R	**G**	**B**	
0.18	0.20	0.08	**Hematoxylin**
0.01	0.13	0.01	**Eosin**
0.10	0.21	0.29	**DAB**

1.4 Metodi di valutazione

1.4.1 La matrice di confusione

Per valutare la performance di un modello esistono delle tecniche di valutazione che ci consentono di analizzare i risultati. Queste funzioni richiedono in input le etichette reali e in output quelle derivanti dalla deduzione del modello. Restituiscono un valore numerico che è indice della qualità dei risultati.

Ci sono alcune metriche di classificazione che ci consentono di valutare l'efficacia del modello, quello che in questa tesi è stata utilizzata è la cosiddetta matrice di confusione.

Ma prima di descriverla bisogna precisare i seguenti concetti:

° True Positive: il data set è costituito da campioni etichettati e classificati allo stesso modo.

° False Positive: il data set è classificato dal modello come appartenente ad una classe ma in realtà ne appartiene ad un'altra.

° False Negative: il data set è classificato dal modello come non appartenente ad una classe ma in realtà appartiene proprio a quella classe.

Come citato poc'anzi la matrice di confusione è uno strumento efficace e utile nella valutazione del modello.

Questa matrice può essere calcolata come un array NxN (righe x colonne), dove N corrisponde alla totalità delle classi che abbiamo.

Essa è strutturata in modo che le colonne j rappresentano le previsione mentre le righe i rappresentano lo stato effettivo. Gli elementi sulla diagonale corrispondono ai campioni correttamente classificati.

La matrice di confusione più semplice da studiare è il caso binario, dove supponendo di avere due etichette -1 e +1, la matrice di confusione corrisponde alla figura 1.8 dove N_{11} corrisponde al vero-positivo, elemento N_{12} falso negativo, N_{21} falso-positivo, N_{22} vero negativo.

		Previsione	
		1	**-1**
Etichetta	**1**	Vero positivo	Falso negativo
	-1	Falso positivo	Vero negativo

Figura 1.14: Matrice di confusione, caso binario.[14]

Se ci troviamo in un problema di classificazione multi-classe la matrice di confusione sarà come quella rappresentata nella figura 1.9, dove come nel caso precedente nella diagonale sono presenti i campioni correttamente classificati e nel resto della matrice sono presenti i falsi-positivi e i falsi-negativi.

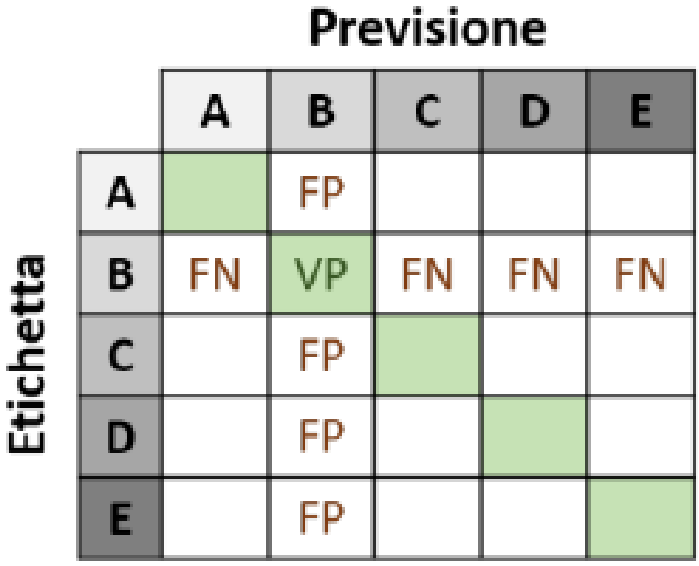


Figura 1.15: Matrice di confusione, con 5 classi.[14]

1.4.2 Coefficiente di Matthews

Strettamente connesso alla matrice di confusione è il cosi detto coefficiente di Matthews. In questo lavoro di tesi ci interessa utilizzarlo perchè vogliamo che le classi siano ben rappresentate.

Nel caso bidimensionale il coefficiente di Matthwes è una metrica a valore singolo che tiene conto di tutte e quattro le voci della matrice di confusione (Vero Positivo, Vero Negativo, Falso Positivo, Falso Negativo)[14]. Di seguito la formula:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{1.9}$$

Esso può avere valori compresi tra -1, che implica una previsione inversa, 0 rappresenta una scelta casuale pesata sulla numerosità delle classi e +1 che implica una previsione perfetta.

Nel caso multiclasse invece la formula generale diventa:

$$MCC = \frac{\Sigma_k \Sigma_l \Sigma_m C_{kk} C_{lm} - C_{kl} C_{mk}}{\sqrt{\Sigma_k (\Sigma_l C_{kl})(\Sigma_{k'|k' \neq k} \Sigma_{l'} C_{k'l'})} \sqrt{\Sigma_k (\Sigma_l C_{lk})(\Sigma_{k'|k' \neq k} \Sigma_{l'} C_{l'k'}}} \tag{1.10}$$

Definendo variabili intermedie:

- $t_k = \sum_i C_{ik}$ numero di volte in cui si è verificata veramente la classe k;
- $p_k = \sum_i C_{ki}$ numero di volte in cui è stata prevista la classe k;
- $c = \sum_k C_{kk}$ numero totale di campioni correttamente previsto;
- $s = \sum_i \sum_j C_{ij}$ il numero totale di campioni.

Dunque la formula può essere semplificata in questo modo:

$$MCC = \frac{cs - \vec{t} \cdot \vec{p}}{\sqrt{s^2 - \vec{p} \cdot \vec{p}} \sqrt{s^2 - \vec{t} \cdot \vec{t}}} \tag{1.11}$$

Capitolo 2

Pipeline

In questa parte vengono descritte tutte le procedure svolte in questo lavoro di tesi, a partire dalla segmentazione manuale delle immagini fino alla segmentazione automatica operata dal modello knn. Sono infine riportate delle immagini rappresentative dei risultati ottenuti.

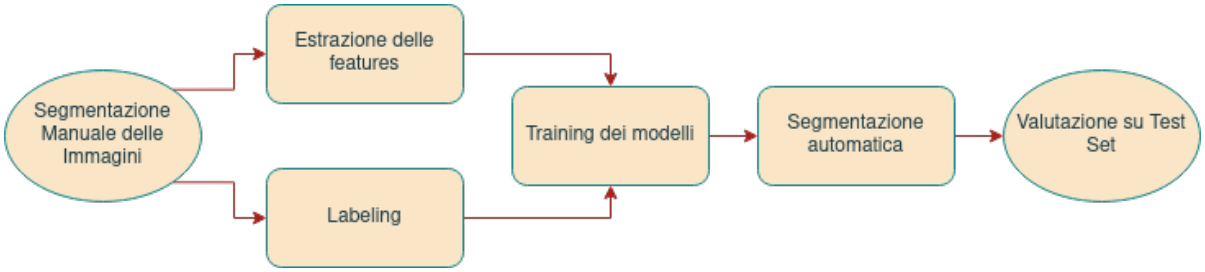


Figura 2.1: Diagramma di flusso

2.1 Segmentazione manuale delle immagini

Il lavoro di questa tesi è iniziato con la segmentazione manuale delle immagini di patch di derma per evidenziare i differenti strati di pelle. Sono stati individuati segmenti corrispondenti alle seguenti classi con colorazioni diverse:

- bianco ([255,255,255]): vetrino;

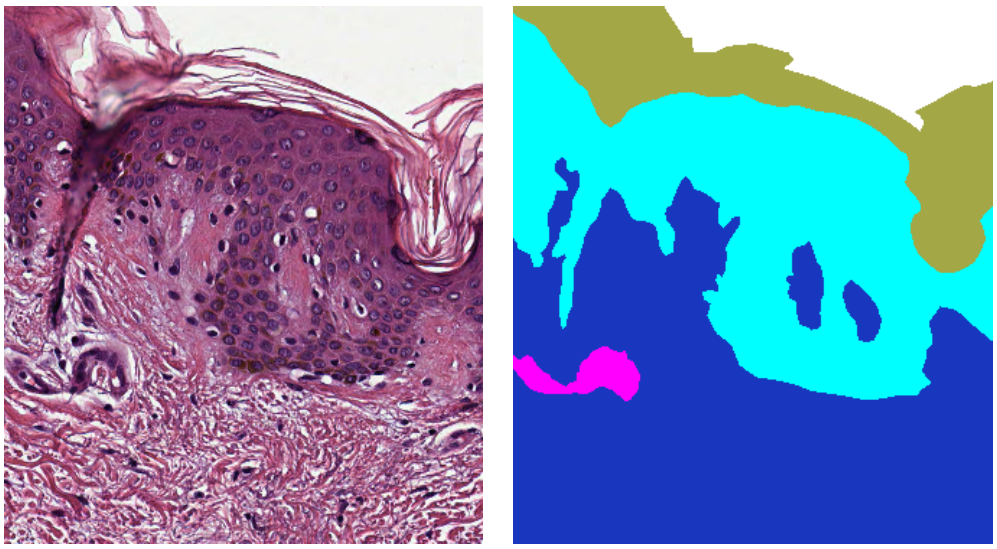
- verde([163,167,69]): strato corneo;

- azzurro([0,255,255]): derma;

- blu([25,55,190]): epidermide;

- viola([255,0,255]): vasi.

Per fare ciò abbiamo utilizzato il software GIMP. Di seguito si riporta un esempio di immagine originale e la corrispondente segmentazione manuale (Figura 2.2)



(a) *302 immagine originale* (b) *302 immagine colorata*

Figura 2.2: a) Immagine originale di un patch di derma. b) Stessa immagine segmentata a mano così suddivisa: bianco per il vetrino, verde per lo strato corneo, azzurro per il derma, blu per l'epidermide e fucsia per i vasi.

2.2 Estrazione delle features a partire dalle immagini originali

Attraverso l'utilizzo di diversi filtri sono state generate ed associate ad ogni pixel delle feature a partire dalle immagini originali. A seguire viene descritto il processo in dettaglio.

I filtri sono stati calcolati sugli spazi di colore RGB, HED, hsv, Lab, una cui presentazione è possibile consultare nella sezione 1.4 di questa tesi.

Innanzi tutto abbiamo aperto l'immagine come matrice di valori RGB, separando le diverse componenti e normalizzando i valori per restare in un intorno dell'intervallo -0,5 e 0,5. La normalizzazione dei valori per le diverse features è stata effettuata scalando ogni variabile per il massimo valore ottenibile, ottenendo così valori nell'intervallo [0, 1] e successivamente sottraendo il valore medio 0.5, portando le variabili nell'intervallo [-0.5, 0.5].

A questo punto abbiamo calcolato rispettivamente il filtro mediano, l'operatore di Sobel (edge detection), e il filtro varianza.

Gli stessi filtri sono stati calclati sugli spazi colore HED, hsv e Lab. Nello spazio colore hsv sono state calcolate altre due feature corrispondenti a rappresentazioni alternative degli stessi valori secondo le seguenti formule:

- $sen(h * 2 * \pi) * s * v$

- $cos(h * 2 * \pi) * s * v$

Le due rappresentazioni così ottenute risultano più efficaci per l'utilizzo nel training dei nostri modelli.

Nello spazio di colore Lab, oltre a calcolare i filtri comuni agli altri spazi di colore, abbiamo sfruttato la componente L (che rappresentando la luminosità e fornisce dunque una versione in scala di grigi dell'immagine) per calcolare un filtro di entropia.

Fatto ciò il programma restituisce una pila di livelli per ogni immagine in cui ogni livello rappresenta una feature.

I dati così creati sono stati gestiti in Python come un dizionario chiave-valore in cui ogni chiave è il nome di una feature ed ogni valore un array bidimensionale Numpy.

Le features generate sono rispettivamente: R-mf(filtro mediano su canale rosso), G-mf(filtro mediano su canale verde), B-mf(filtro mediano su canale blu), R-edge(filtro di edge su canale rosso), G-edge(filtro di edge su canale verde), B-edge(filtro di edge su canale blu), R-variance(filtro varianza su canale rosso), G-variance(filtro varianza su canale verde), B-variance(filtro varianza su canale blu), H-mf(filtro mediano su canale ematossilina), E-mf(filtro mediano su canale eosina), D-mf(filtro mediano su canale Dab), H-variance (filtro di varianza su canale ematossilina), E-variance(filtro di varianza su canale eosina), D-variance(filtro di varianza sul canale Dab), H-edge(filtro di edge su

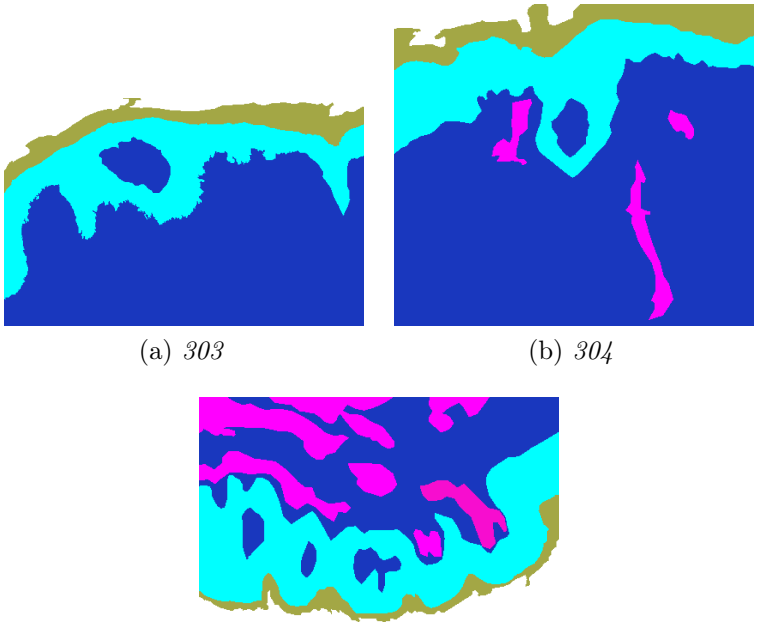
canale ematossilina), E-edge(filtro di edge su canale eosina), D-edge(filtro di edge su canale Dab), L-mf(filtro madiano su canale della luminosità), a-mf(filtro mediano sul canale lungo la direzione a), b-mf(filtro mediano sul canale lungo la direzione b), L-edge(filtro edge sul canale della luminosità), a-edge(filtro edge sul canale lungo la direzione a), b-edge(filtro edge sul canale lungo la direzione b), L-variance(filtro varianza sul canale della luminosità), a-variance(filtro varianza sul canale lungo la direzione a), b-variance(filtro varianza sul canale lungo la direzione b), L-entropy(filtro di entropia sul canale della luminosità), h-mf(filtro mediano sul canale della tonalità), s-mf(filtro mediano sul canale della saturazione), v-mf(filtro mediano sul canale del value), h-edge(filtro di edge sul canale della tonalità), s-edge(filtro di edge sul canale di saturazione), v-edge(filtro di edge sul canale del value), h-variance(filtro varianza sul canale di tonalità), s-variance(filtro varianza sul canale di saturazione), v-variance(filtro varianza sul canale di value), hsv-sin, hsv-cos.

2.3 Labeling delle diverse sezioni a partire dalle segmentazioni manuali

I valori target per ogni pixel delle immagini sono stati ottenuti a partire dalle immagini segmentate manualmente. Ad ogni pixel è stata associata una classe in base al proprio colore secondo la suddivisione descritta nel paragrafo (2.1). In questa fase ogni classe target è rappresentata da un valore numerico intero.

2.4 Training dei modelli KNN

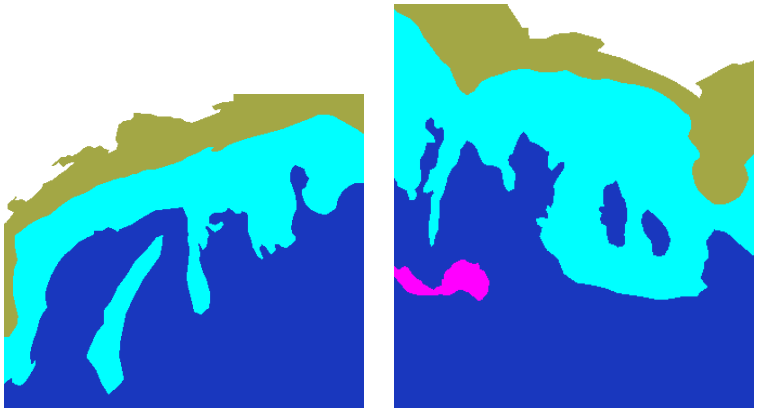
I dati così ottenuti sono stati separati in un development set (training e validation) composto dalle immagini 303, 304, 601 (training) e 301, 302, 305 (validation) ed un test set composto dalle immagini (101, 102, 103, 104, 105, 1001, 1002). Qui di seguito sono state riportate le immagini utilizzate nel traing set, rispettivamente le 303, 304, 601, le immagini utilizzate per il validation set che sono rispettivamente le 301, 302, 305 ed in fine le immagini utilizzate come test set.



(a) *303* (b) *304*

(c) *601*

Figura 2.3: Figure segmentate a mano utilizzate per il training set



(a) *301* (b) *302*

(c) *305*

Figura 2.4: Figure segmentate a mano utilizzate per il validation set

Essendo le immagini di dimensioni differenti qui di seguito sono riportate le dimensioni dei differenti insiemi di dati in termini di pixel (che sono gli effettivi datapoint):

- training set: 421776 pixel;

- validation set: 428302 pixel;

- test set: 1007130 pixel.

Il training dei modelli si è svolto in fasi successive durante le quali sono state selezionate le features più significative attraverso un processo di ricerca (grid search) sullo spazio dei seguenti iperparametri:

• numero di vicini da considerare,

• dimensione del kernel per i filtri,

• sottoinsieme delle features da includere.

Per ogni fase della ricerca abbiamo ottenuto un numero variabile di modelli a seconda delle combinazioni di features utilizzate (da 2400 a circa 14000). I modelli ottenuti sono stati ordinati per coefficiente di Matthews raggiunto sul validation set ed in base a questo ranking è stata operata la selezione degli iperparametri.

Qui di seguito riportiamo una tabella che mostra l'ultima grid search che abbimo eseguito, si puo vedere i vari iperparamentri che sono stati presi in considerazione:

	Neighbors	Kernel_Size	Features
GreadSearch			
1	[3,5,7,10,15,20,25,50]	[20, 25, 30, 35, 40]	*[R, G, B, R_mf, G_mf, B_mf, R_variance, G_variance, B_variance, L, a, b, L_mf, a_mf, b_mf, L_variance, a_variance, b_variance, h, s, v, h_mf, s_mf, v_mf, h_variance, s_variance, v_variance, hsv_sin, hsv_cos, H, E, D, H_mf, E_mf, D_mf, H_variace, E_variance, D_variance, L_ entropy]*

Figura 2.5: Tabella che mostra i numeri di vicini considerati, dimensioni del kernel e sottoinsieme delle features considerate.

Il coefficiente di correlazione di Matthews è stato scelto come metrica perché tiene conto della numerosità delle varie classi che abbiamo e ci da informazioni di quanto sbaglia il modello tra le varie classi.

2.5 Segmentazione automatica

I modelli ottenuti nella fase di training possono essere utilizzati per operare una segmentazione automatica delle immagini originali. Alle immagini di segmentazioni generate dai modelli knn selezionati è stato successivamente applicato un filtro di denoising (attraverso l'applicazione di una moda scorrevole) per ridurre il rumore ed ottenere segmentazioni più precise.

L'applicazione del filtro moda consiste nel calcolare la moda dei valori assunti da una feature dei pixel che circondano il pixel centrale nella matrice kernel nxn. La moda risultante va a sostituire il valore originale del pixel centrale. Nel nostro caso il kernel ha dimensioni 6x6.

Come risultato di questo processo abbiamo dunque ottenuto per ognuna delle immagini una versione "spuria" della segmentazione ed una seconda versione con rumore ridotto che ha generalmente raggiunto uno score migliore.

Nell'immagine seguente è possibile vedere un esempio di confronto tra immagine originale, segmentazione manuale, segmentazione automatica "spuria" e segmentazione automatica con denoising.

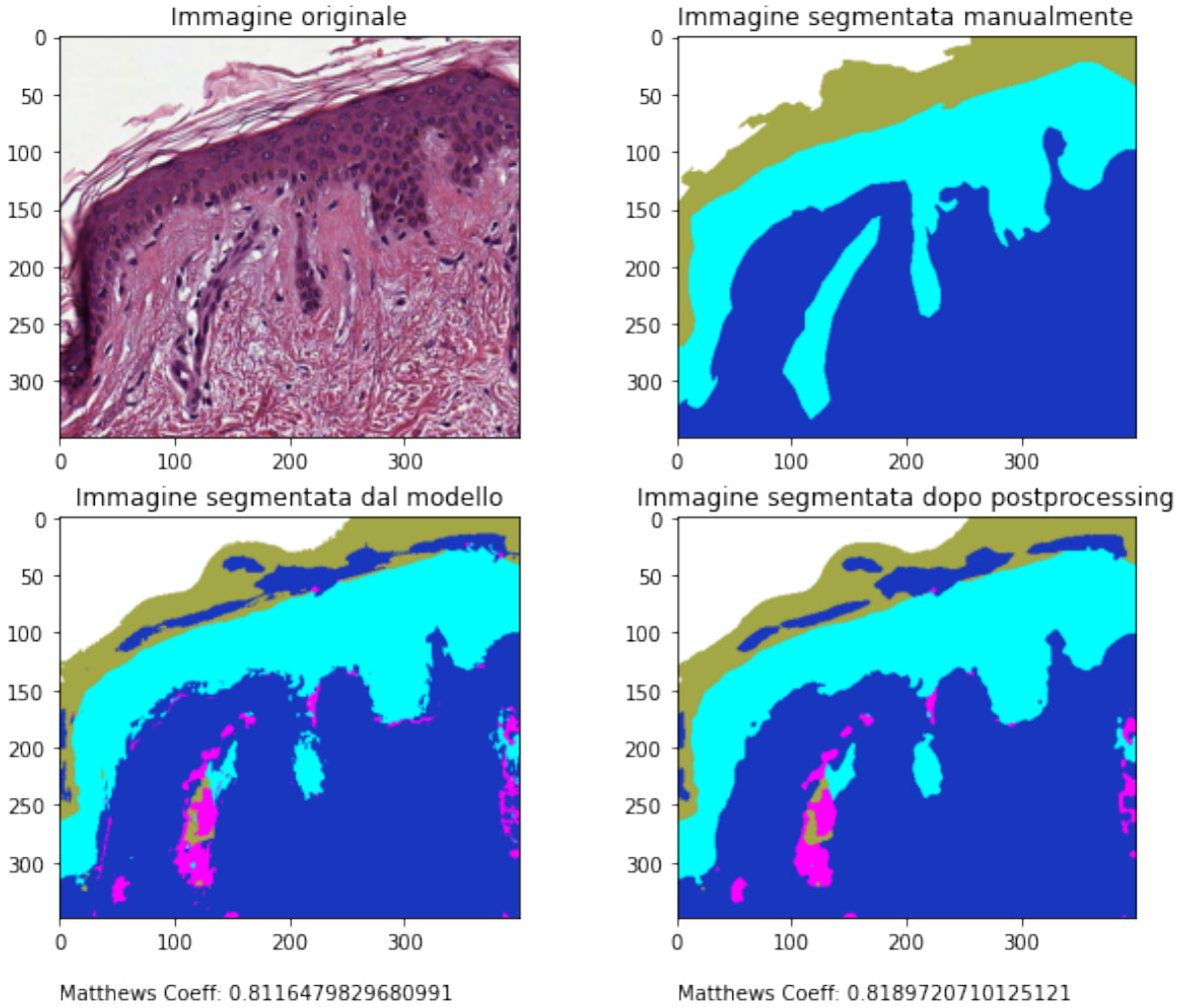


Figura 2.6: Una delle immagini utilizzate per il validation set. Partendo dall'alto osserviamo rispettivamente: l'immagine originale, la segmentazione manuale, la segmentazione automatica "spuria" e la segmentazione automatica con denoising

2.6 Valutazione su test set

Completata la fase di grid search è stato selezionato il modello con i risultati migliori che è stato successivamente applicato al test set per la valutazione della capacità di generalizzazione su immagini mai incontrate. I risultati sono presentati in maniera dettagliata nel proseguo di questa tesi.

Di seguito si riporta la tabella con all'interno i primi dieci modelli migliori dell'ultima grid search.

MIGLIOR MODELLO			
score	window_size	features	neighbors
0.8121581491663432	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.8106640599940792	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.8089454882926896	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.8082705659179271	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.8078245168180355	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	20
0.8074563890866494	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	20
0.8068601185167039	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.805733352830825	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	15
0.805331366029398	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.8052702892564052	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	15

Figura 2.7: Tabella con i primi dieci modelli migliori ottenuti dall'ultima grid searh eseguita.

2.7 Ambiente di lavoro

In questa sezione descriverò la macchina che ho utilizzato per sviluppare il progetto e l'ambiente di lavoro che è stato realizzato.

Il lavoro di computazione è stato eseguito per le parti interattive e meno computazionalmente intense su un laptop con sistema operativo MX/Linux, e su un server del Dipartimento di Ingegneria Biomedica con sistema operativo Debian Linux, per quanto riguarda l'esecuzione in batch delle operazioni più pesanti.

Tutto il codice prodotto durante lo sviluppo, le immagini ed i dati prodotti sono stati raccolti in un apposito repository su GitHub, che è liberamente disponibile.[15,16]

Tutto il software è stato scritto in Python 3.8 e 3.9 e sono state installate ed utilizzate varie librerie compatibili con queste versioni illustrate di seguito:

NumPy : è una libreria open source di funzioni del linguaggio di programmazione Pyhon. Il modulo contiene diverse funzioni e metodi utili per il calcolo scientifico. In particolar modo per il calcolo vettoriale e matriciale. [17]

SciPy. La libreria SciPy è uno dei pacchetti principali che compongono lo stack SciPy. Fornisce molte implementazioni numeriche facili ed efficienti da usare, come integrazione numerica, interpolazione, ottimizzazione, algebra lineare e statistica. [18]

Due moduli in particolare hanno ricoperto un ruolo essenziale in questo progetto:

SciPy.ndimage che fornisce una serie di funzioni generali di elaborazione e analisi delle immagini progettate per operare con array di dimensionalità arbitraria;[19]

SciPy.stats è un modulo che contiene un gran numero di distribuzioni di probabilità e una crescente libreria di funzioni statistiche.

ImageIO: è una libreria Python che fornisce un'interfaccia semplice per leggere e scrivere un'ampia gamma di dati di immagini.

Sklearn: è un modulo del linguaggio Python che fornisce implementazioni di numerosi algoritmi di machine learning e varie funzionalità di supporto. Sono stati utilizzati tre moduli in particolare:

Sklearn.metrics: è un modulo che implementa funzioni che valutano l'errore di previsione per scopi specifici.

Abbiamo utilizzato come funzione:

sklearn.metrics.confusion_ matrix. Calcola la matrice di confusione per valutare l'accuratezza di una classificazione. [20]

sklearn.metrics.matthwes_ corrcoef. Calcola il coefficiente di correlazione di Matthwes (MCC).[21]

Sklearn.neighbors: è un modulo che implementa l'algoritmo k-nearest-neighbors.

In particolare abbiamo utilizzato:

sklearn.neighbors.kNeighborsClassifier: che è il classificatore che implementa il voto di pluralità dei k vicini più vicini.

Sklearn.preprocessing: è un pacchetto che fornisce diverse funzioni di utilità comuni e classi di trasformatori per modificare i vettori di elementi grezzi in una rappresentazione più adatta per gli stimatori a valle.

È stato usato in particolare:

sklearn.preprocessing.StandardScaler: è una funzione che standardizza le caratteristiche rimuovendo la media e ridimensionando la varianza all'unità.

Matplotlib: è una libreria completa per la creazione di visualizzazioni statistiche, animate e interattive in Python.[22]

Nello specifico è stato usato:

matplotlib.pyplot: è un interfaccia basata sullo stato per matplotlib.

Scikit-image: è un pacchetto Python dedicato all'elaborazione delle immagini e utilizza nativamente array NumPy come oggetti immagine.[23]

Scikit-image.color: modulo che permette di convertire da uno spazio di colore all'altro.

Capitolo 3

Risultati

Nelle fasi finali della grid search abbiamo utilizzato il modello con il miglior Matthews coefficient per generare esplicitamente le segmentazioni di alcune immagini di test.

Il modello migliore operato sul Knn restituisce queste features:
R-variance, G-variance, B-variance, H-mf, E-mf, D-mf, L-entropy; con neighbors=50 e window size= 30.

Sul modello migliore del Knn abbiamo voluto confrontare cosa succede se operiamo con il Random Forest e il Support Vector Machine. Qui di seguito si visualizzano le stesse immagini del test set utilizzate per il Knn, segmentate altresì con il modello RF e con il modello SVM, visualizzandone anche il coefficiente di Matthews. Discutiamo di volta in volta il risultato confrontando le stesse immagini con algoritmi diversi. Riportiamo anche le rispettive matrici di confusione per avere un quadro completo sull' accuratezza della classificazione, tenendo quindi in considerazione tutte le molteciplità delle classi.

3.1 Discussione immagine 104 con i modelli Knn, RF e SVM

Cominciamo a prendere in considerazione l'immagine 104 secondo i modelli KNN, RF e SVM (figure da pag.43 a pag.48). Come si nota dall'immagine il Knn in generale riesce a segmentare in modo corretto quasi tutte le classi, fa un po più fatica a riconoscere i vasi.

Il RF invece confonde gran parte del derma in strato corneo, ma anche la parte segmentata a mano come vaso, la riconosce per la maggior parte come derma.

L'algoritmo SVM la zona segmentata a mano come vaso sanguigno viene riconosciuta per la maggior parte come strato corneo e solo ai bordi viene riconosciuta come vaso. Lo strato corneo spesso viene confuso come vetrino e derma.

In generale però come si nota tutti e tre i metodi, che normalmente fanno errori diversi in diverse aree, tendenzialmente stanno tutti e tre commettendo lo stesso errore, cioè quello di riconoscere i vasi.

Questo potrebbe dipendere dal fatto che in principio è stata selezionata a mano, in modo errato, una parte che non è vaso sanguigno; pertanto gli algoritmi, non riconoscendo quella zona come vaso (perchè è una zona un po mista di chiaro/scuro), tendono a classificare quella zona in modo errato.

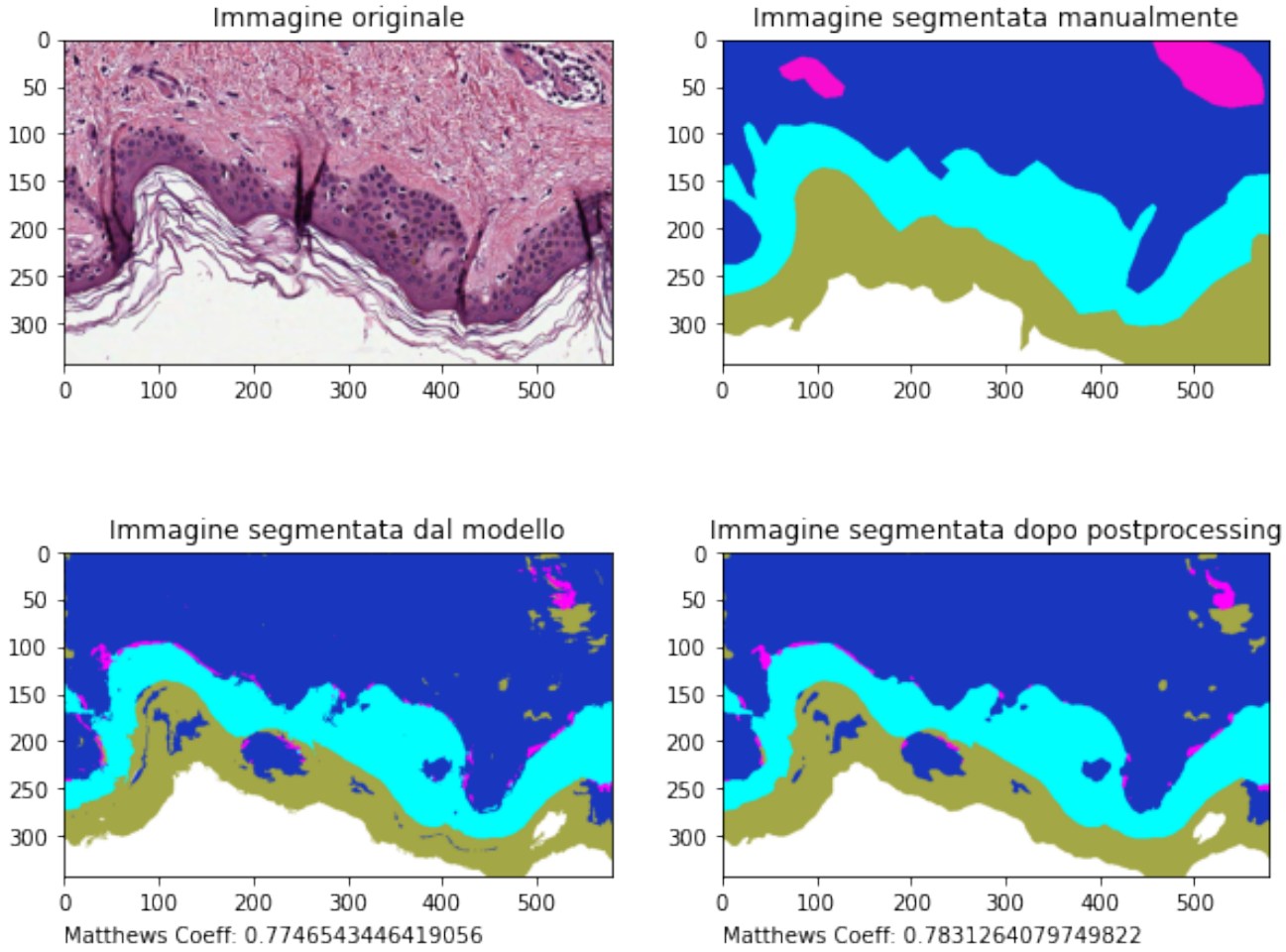


Figura 3.1: Immagine 104 utilizzata per il test set sull'algoritmo KNN

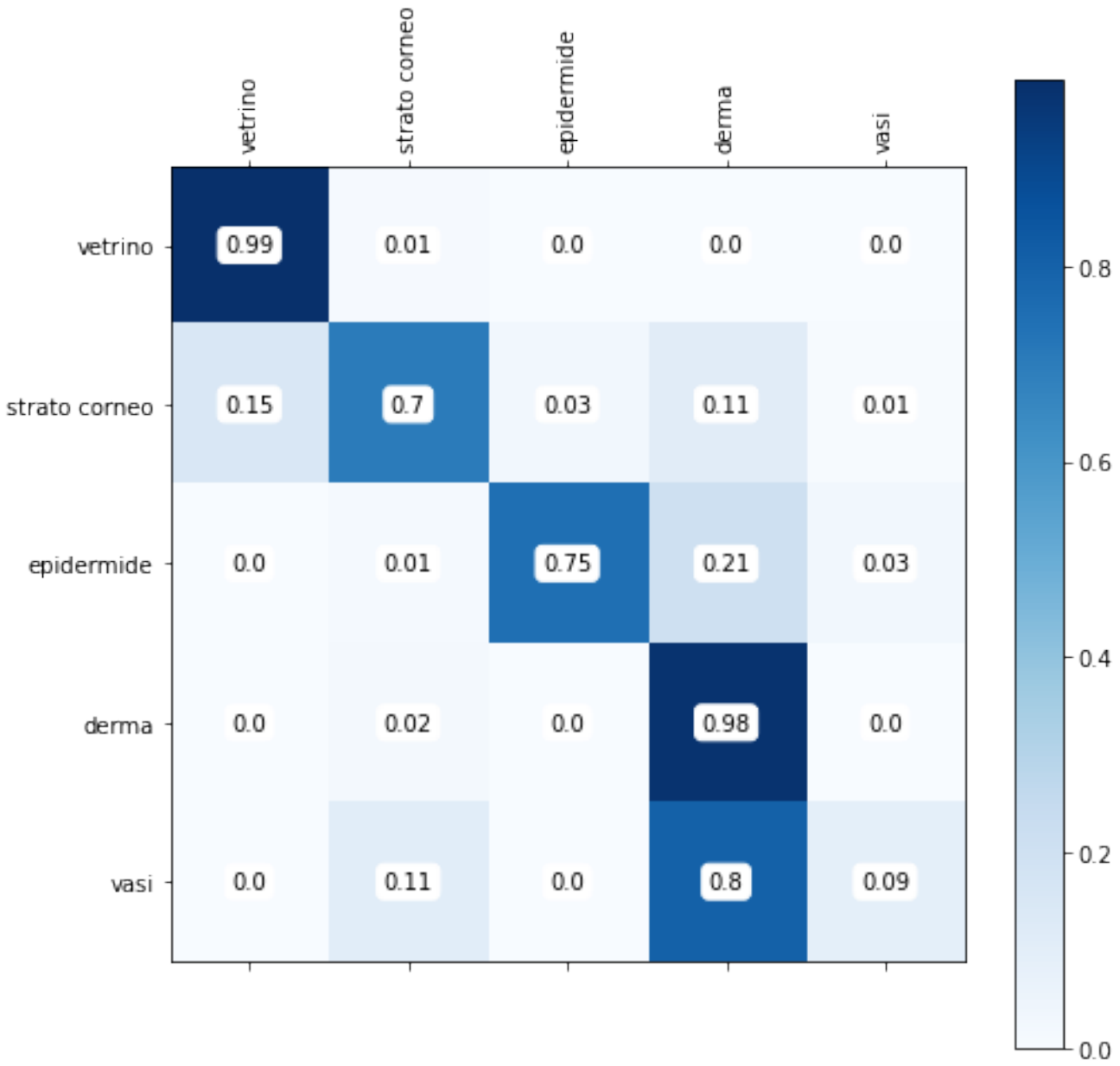


Figura 3.2: Matrice di confusione dell'immagine 104 sull'algoritmo KNN

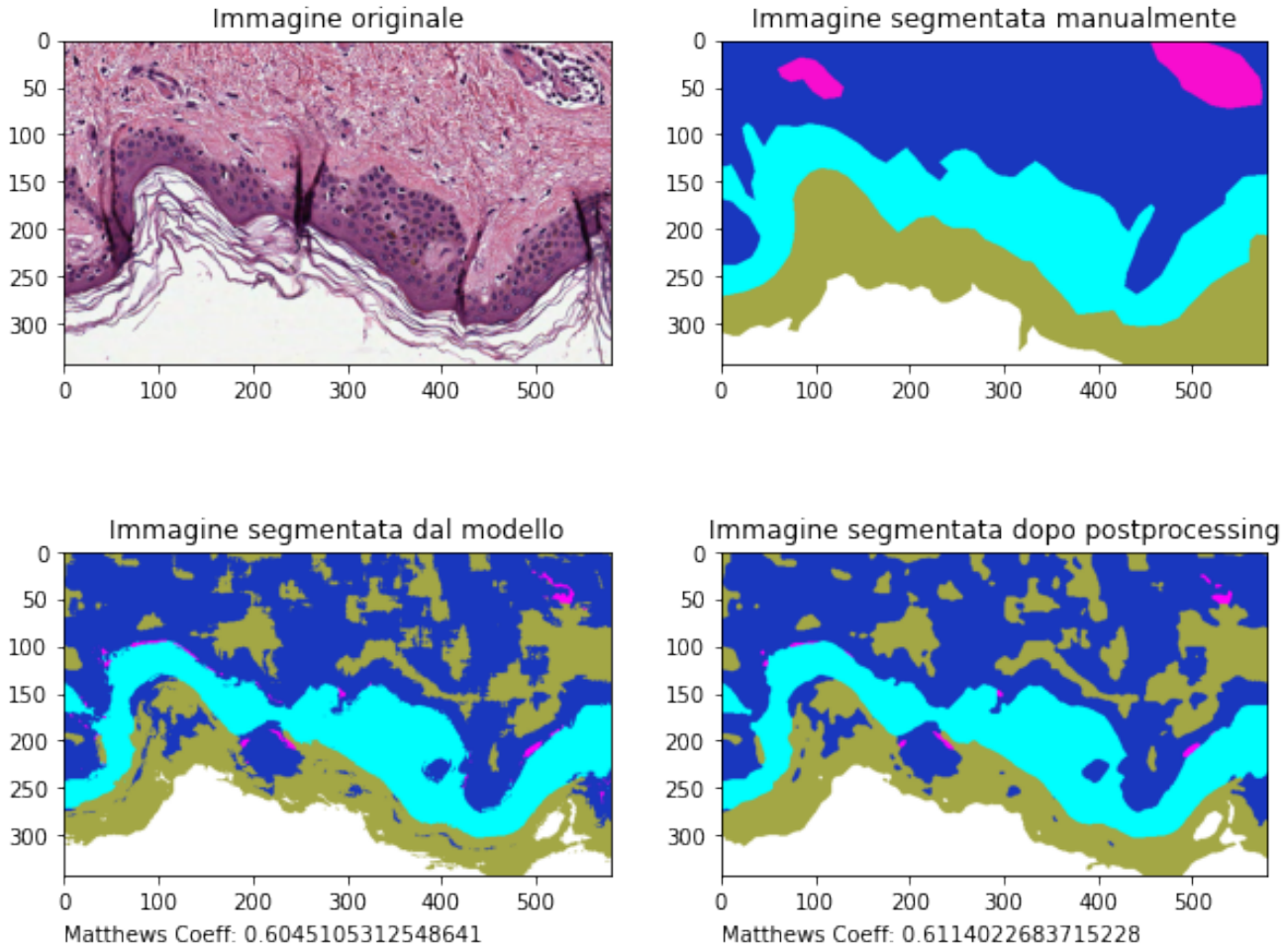


Figura 3.3: Immagine 104 utilizzata per il test set sull'algoritmo RF

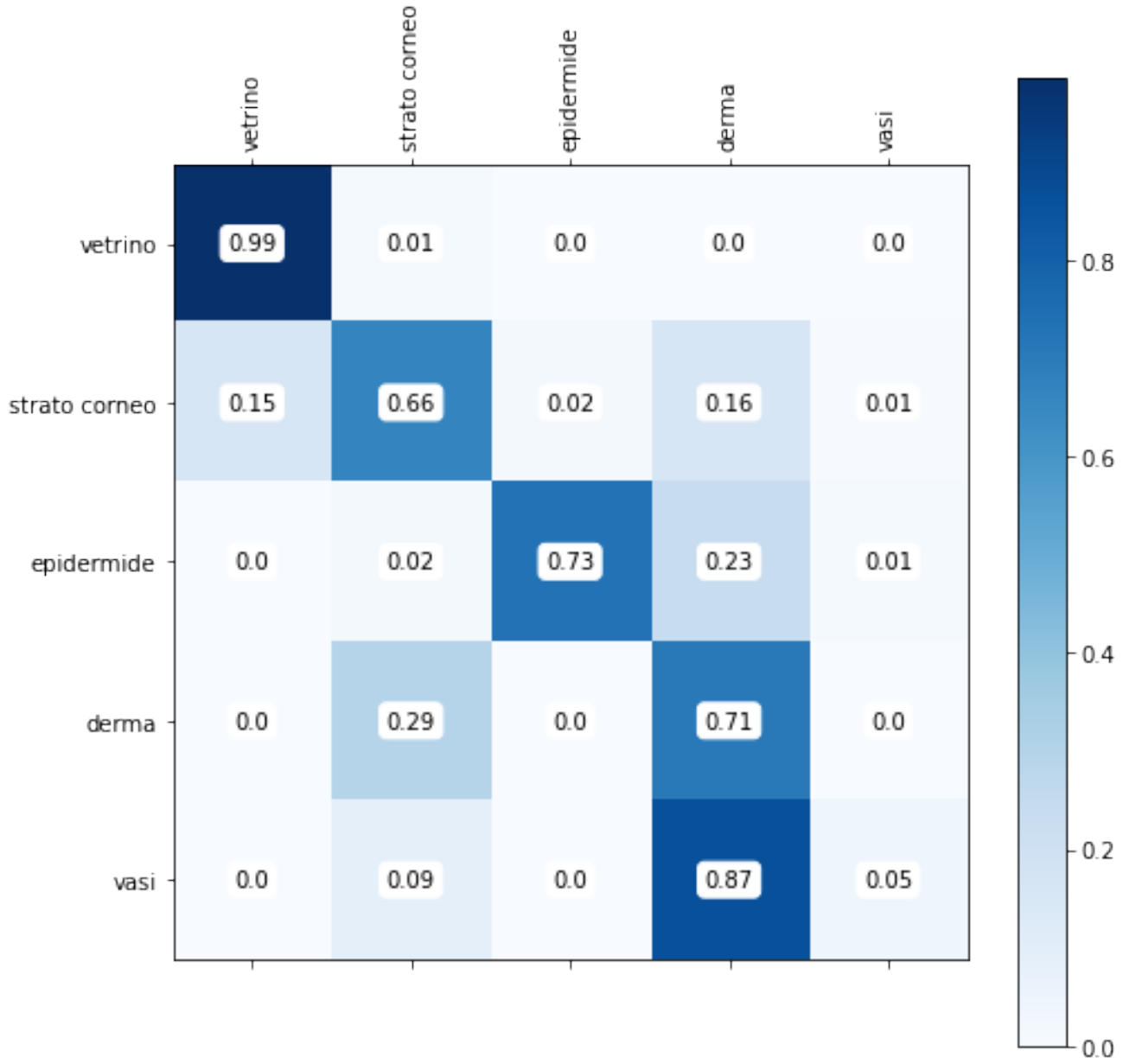


Figura 3.4: Matrice di confusione dell'immagine 104 sull'algoritmo RF

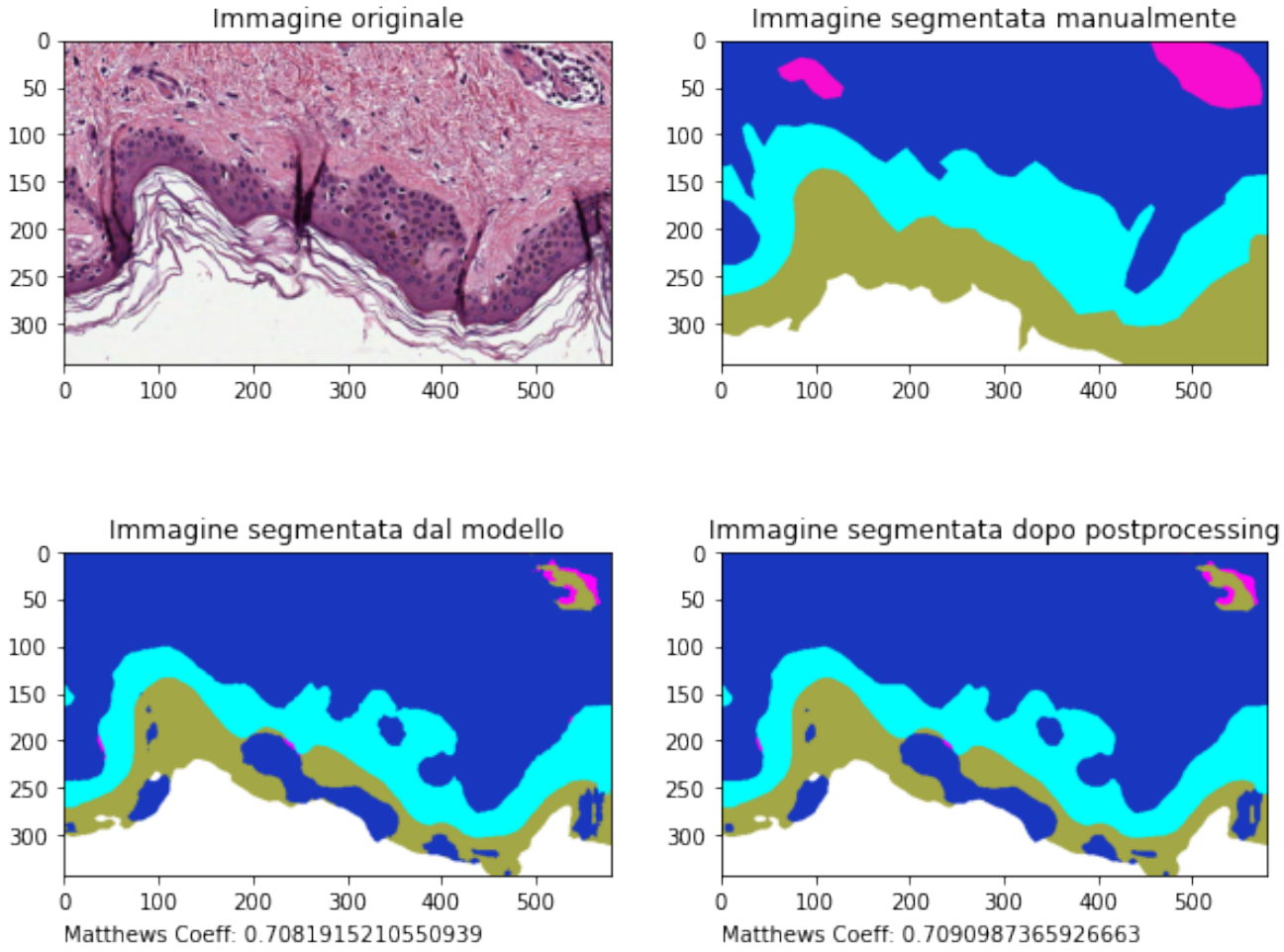


Figura 3.5: Immagine 104 utilizzata per il test set sull'algoritmo SVM

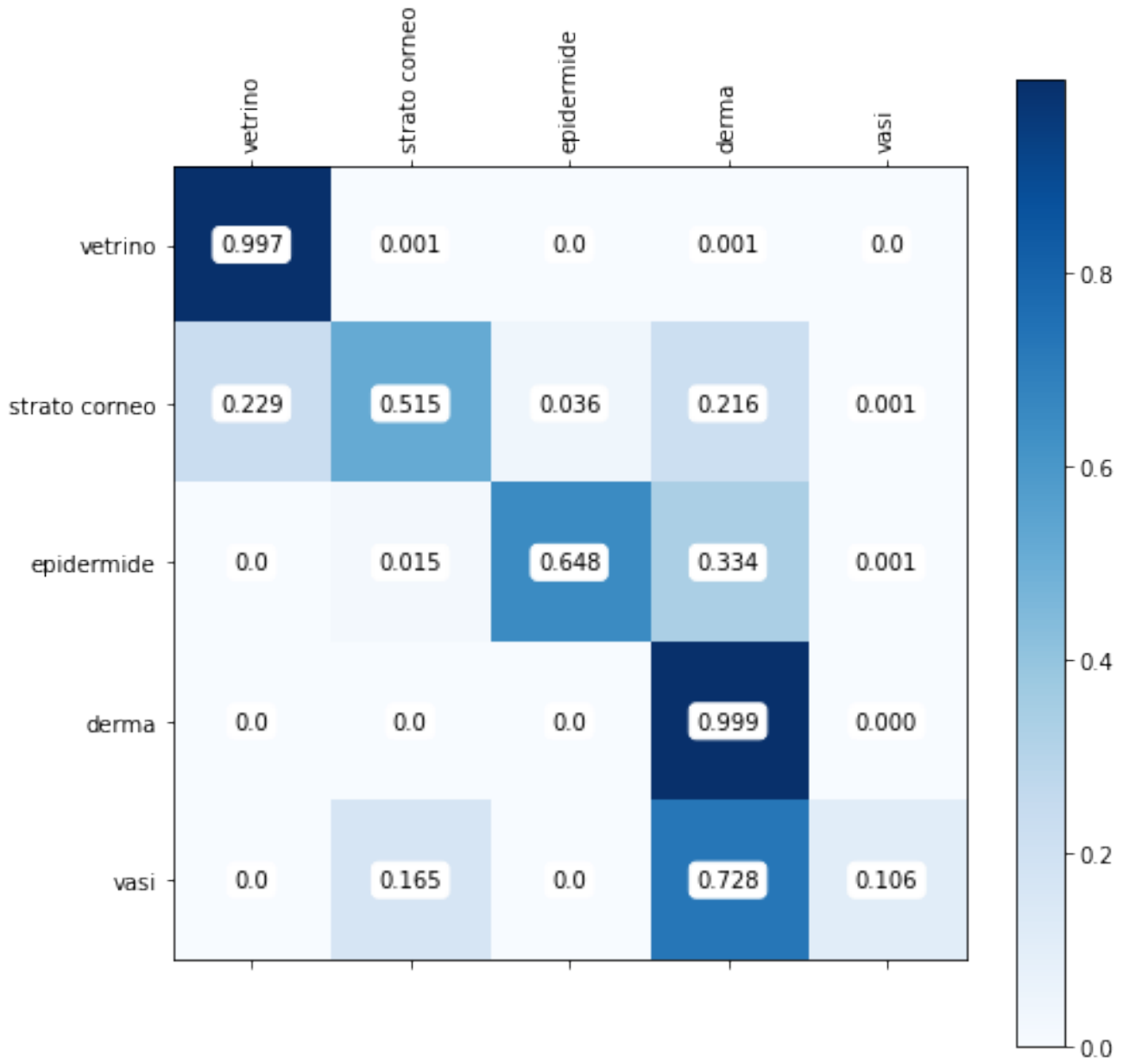


Figura 3.6: Matrice di confusione dell'immagine 104 sull'algoritmo SVM

3.2 Discussione immagine 1002 con i modelli Knn, RF e SVM

Su questa immagine (1002)(figure da pag.49 a pag.54), si nota che per quanto concerne lo strato corneo, il modello KNN lo riconosce metà come effettivamente strato corneo e metà come vetrino. Inoltre il modello riconosce gran parte dei vasi come derma. Si nota però un interessante particolare: la macchia bianca che si trova in basso nell'immagine originale e che è segmentata manualmente come vaso, viene riconosciuta dal modello come strato corneo, mentre la macchia scura a fianco, segmentata a sua volta come vaso, viene riconosciuta come epidermide; in questo ultimo caso solo l'area di contorno intorno alla zona viene riconosciuta effettivamente come vaso. Questo sta ad indicare che l'algoritmo KNN non lavora a memoria, ma è capace di individuare classi diverse in base al chiaro/scuro.

Il RF in questo caso riconosce le classi in un modo molto simile al KNN.

Con l'algoritmo SVM si fa fatica a riconoscere i bordi dal derma, individuandoli come vasi sanguigni, la macchia bianca di cui sopra, segmentata a mano come vaso viene riconosciuto anche in questo modello come strato corneo.

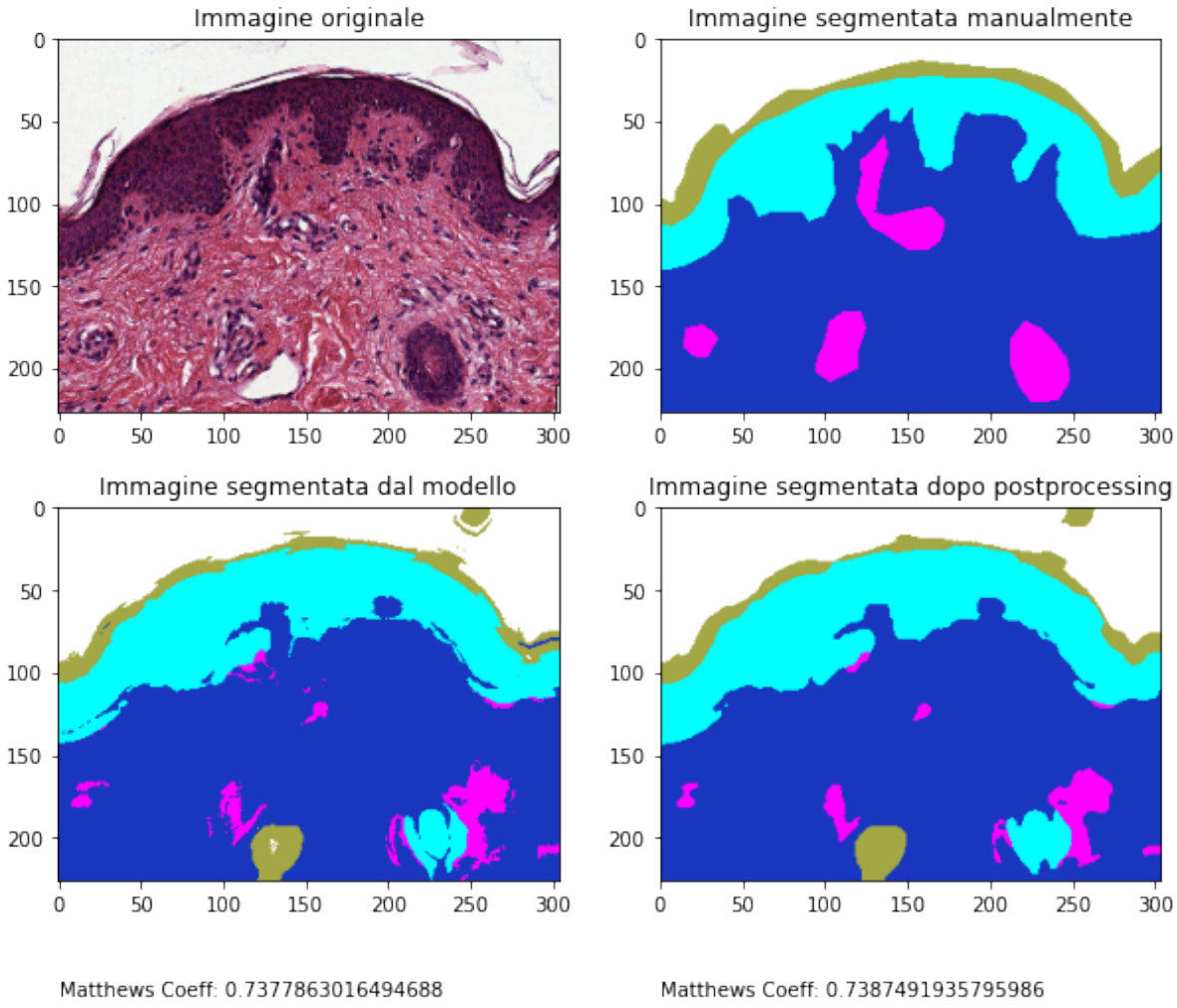


Figura 3.7: Immagine 1002 utilizzata per il test set sull'algoritmo KNN

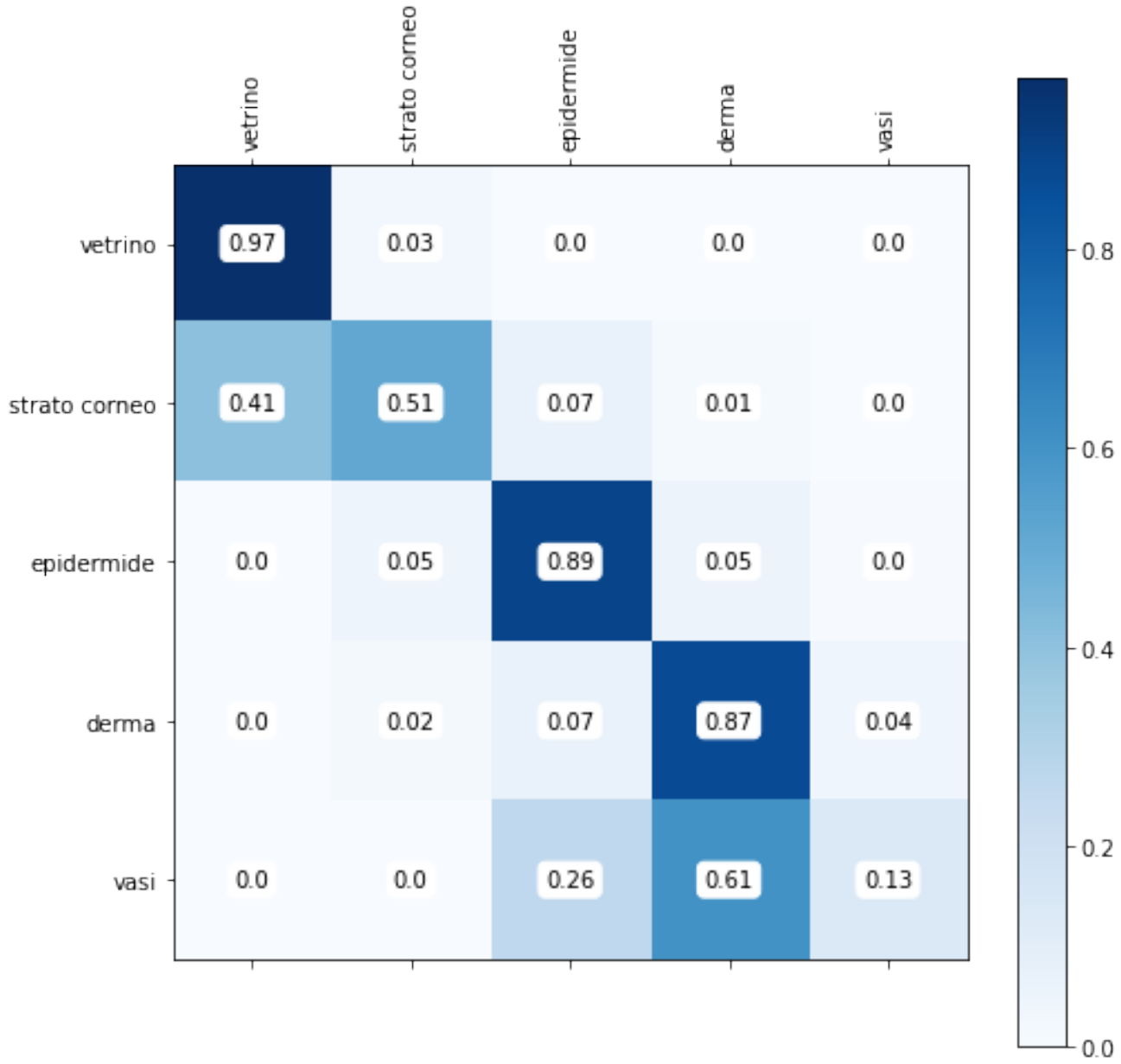


Figura 3.8: Matrice di confusione dell'immagine 1002 sull'algoritmo KNN

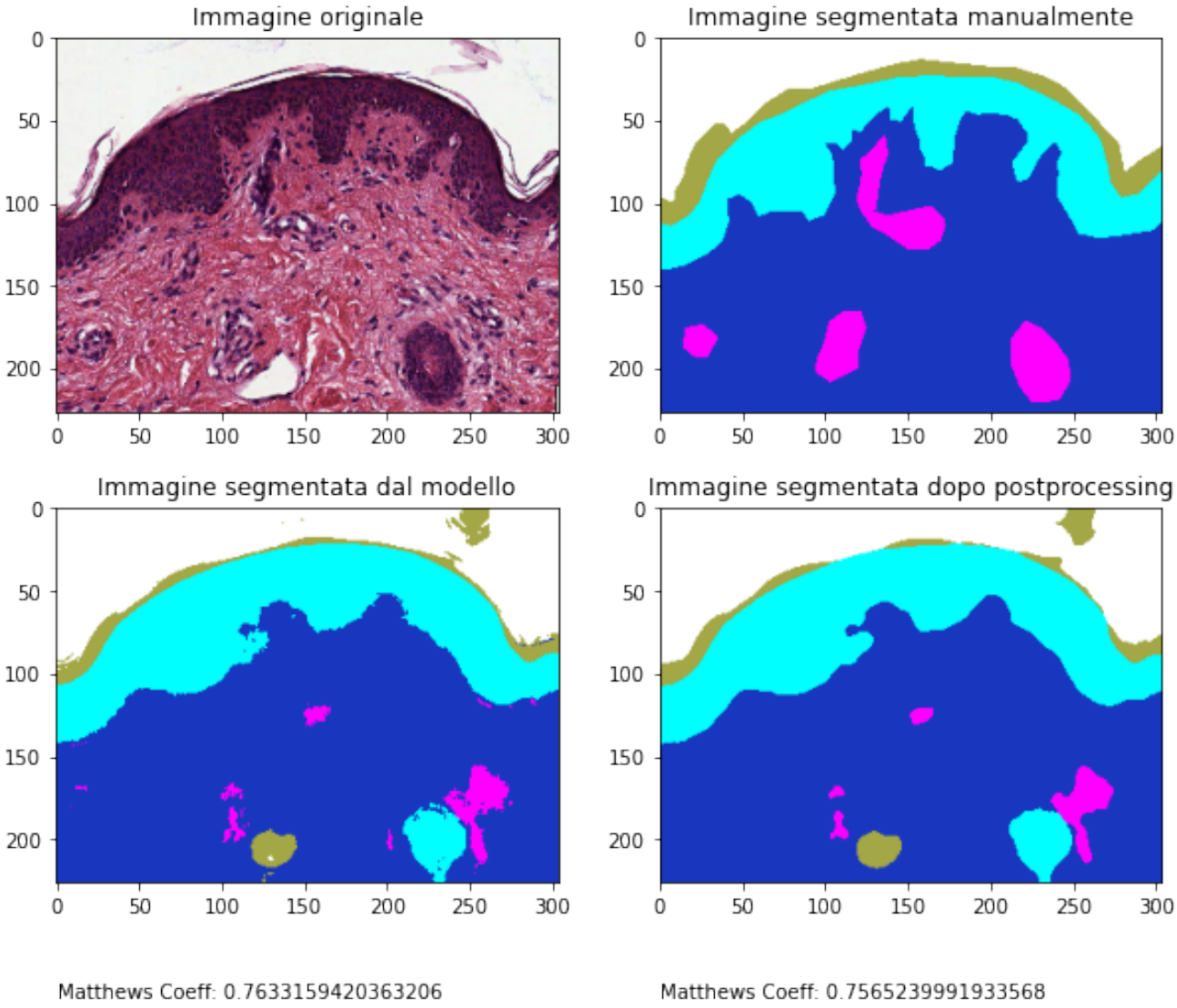


Figura 3.9: Immagine 1002 utilizzata per il test set sull'algoritmo RF

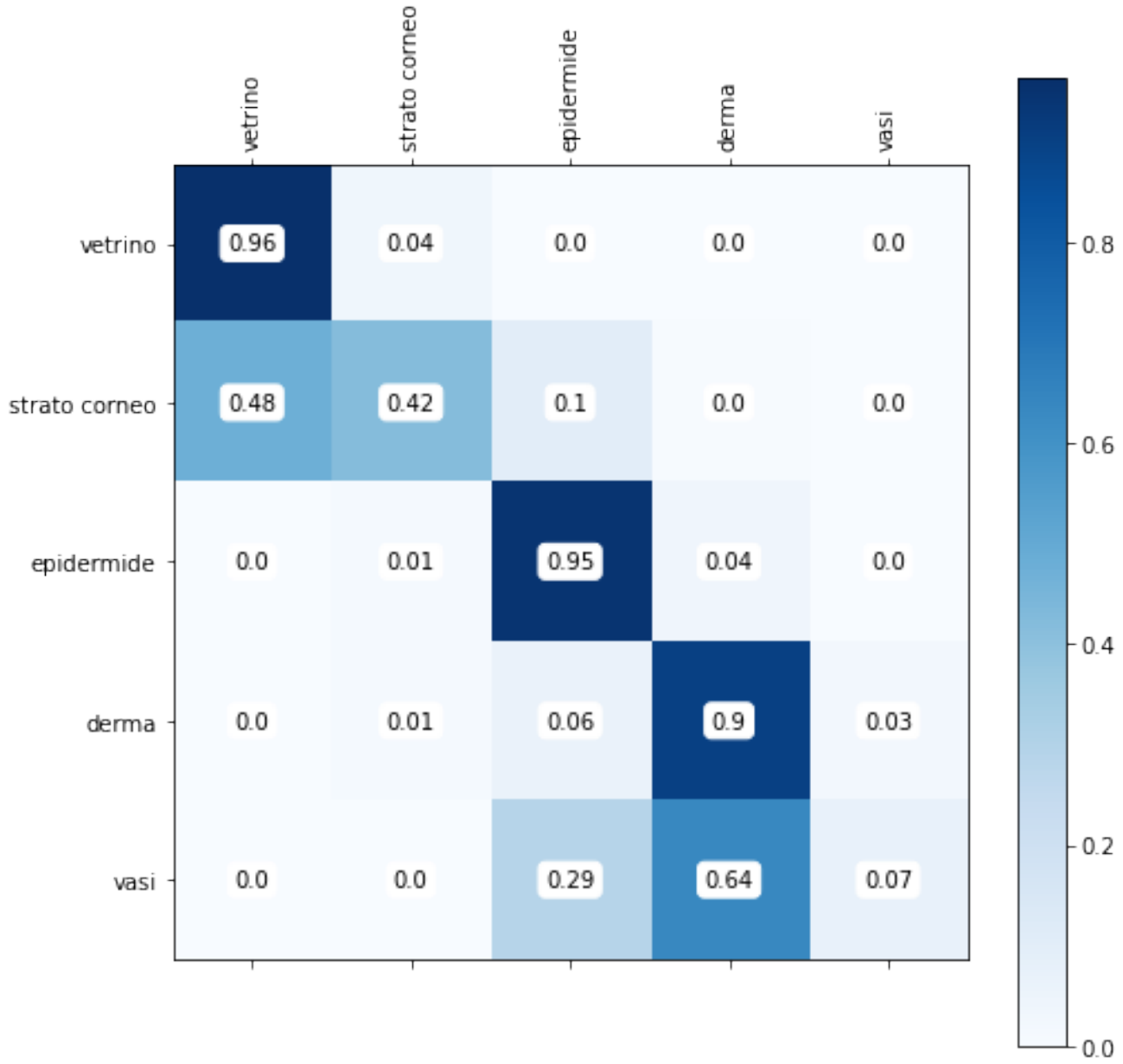


Figura 3.10: Matrice di confusione dell'immagine 1002 sull'algoritmo RF

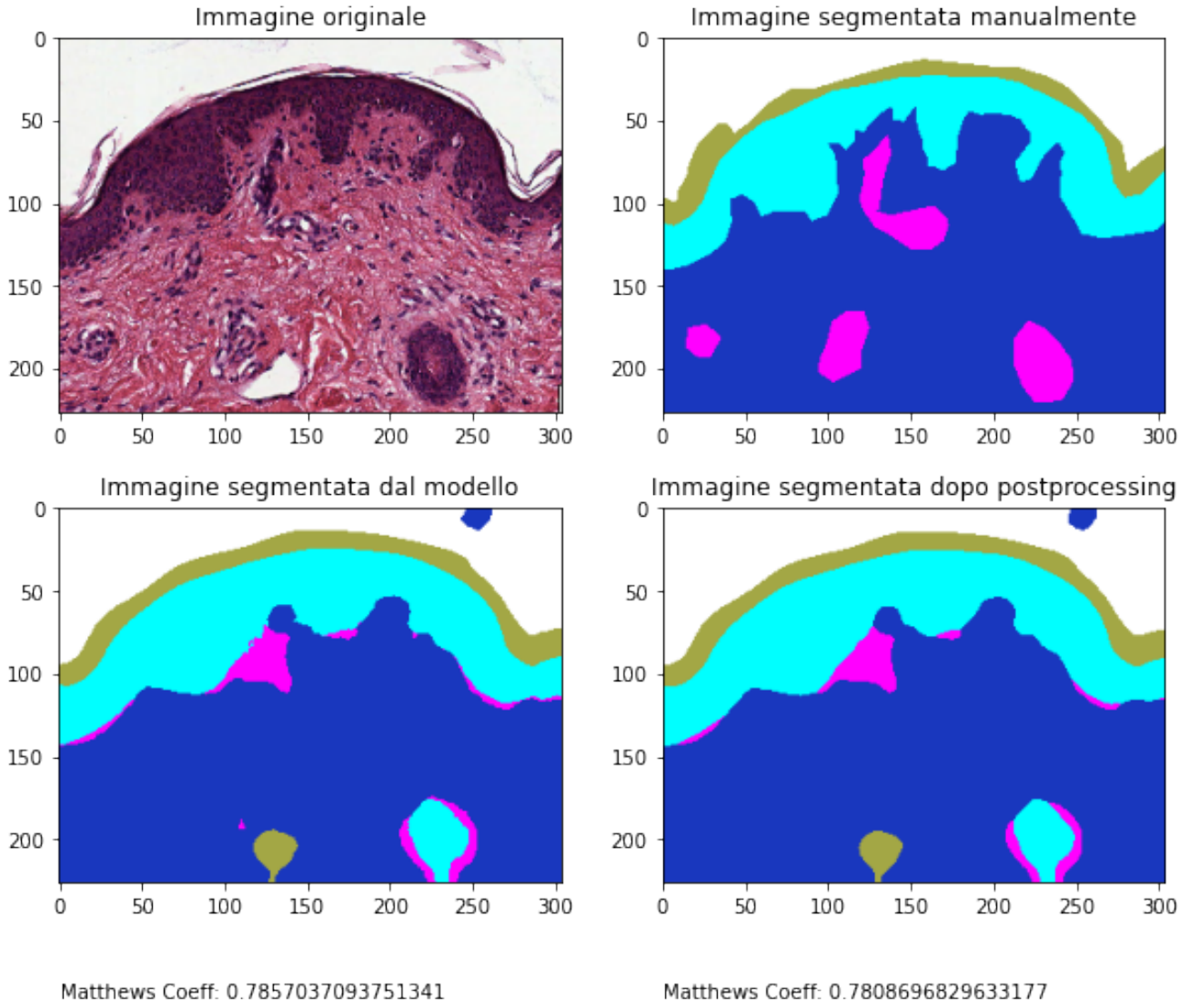


Figura 3.11: Immagine 1002 utilizzata per il test set sull'algoritmo SVM

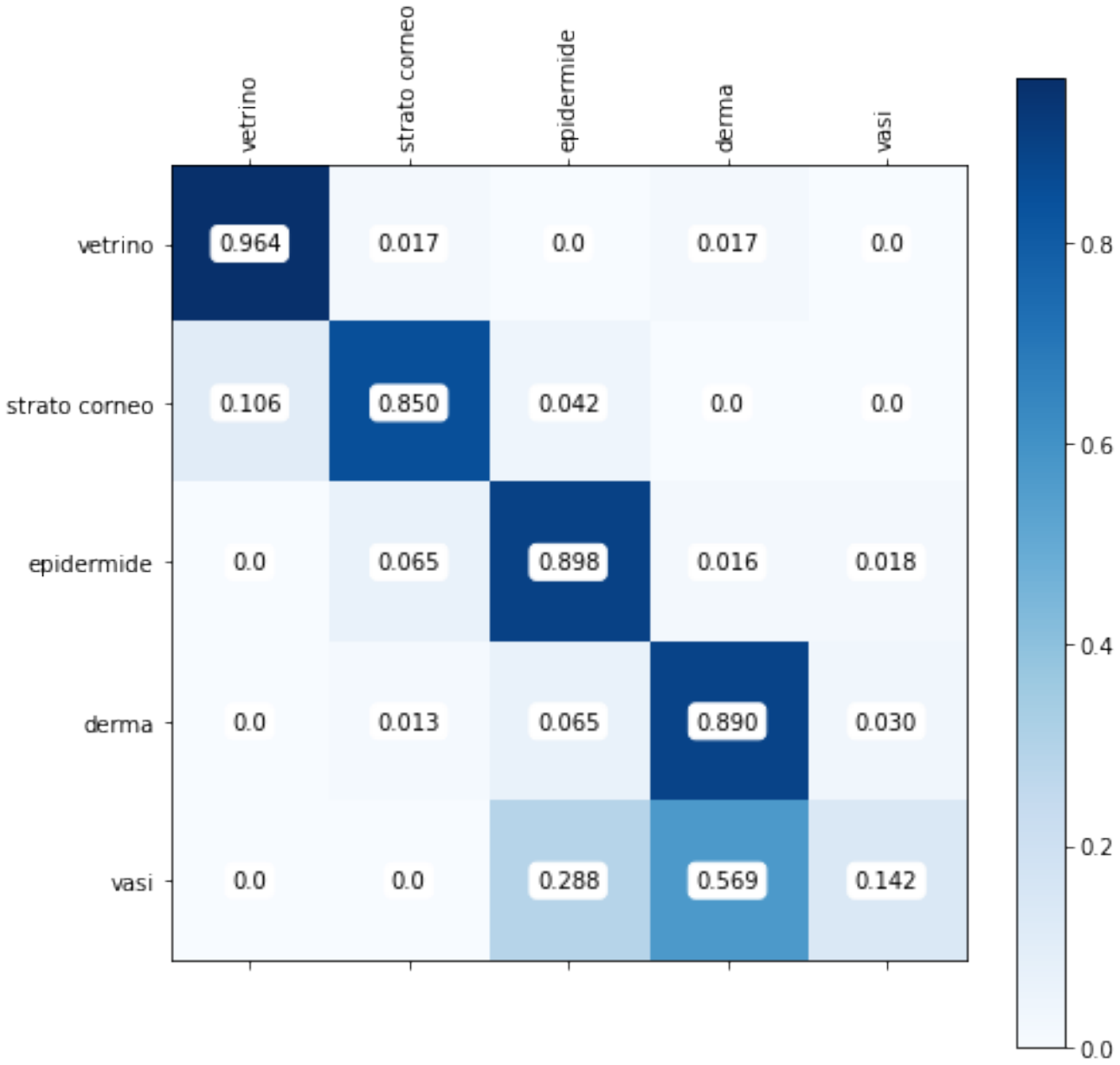


Figura 3.12: Matrice di confusione dell'immagine 1002 sull'algoritmo SVM

3.3 Discussione immagine 1001 con i modelli Knn, RF e SVM

Un discorso simile riguardo l'immagine precedentemente commentata, può essere fatto in merito all'immagine 1001 (si veda da pag. 55 a 60). Si nota che il modello KNN per la maggior parte riconosce lo strato corneo come effettivamente strato corneo e in parte minore come vetrino. I vasi invece sono quasi totalmente confusi in derma. La porzione chiara/bianca in basso a destra nell'immagine originale, segmentata manualmente come derma, viene erroneamente riconosciuta dal modello come strato corneo.

Un risultato molto similare si ottiene tramite il modello RF.

Per quanto riguarda il SVM è importante rilevare come il modello tenda a riconoscere ciò che era stato segmentato come vaso in derma. Inoltre, nell'area tra l'epdidermide ed il derma, il modello tende a rilevare erroneamente zone che dovrebbero essere di derma come vasi.

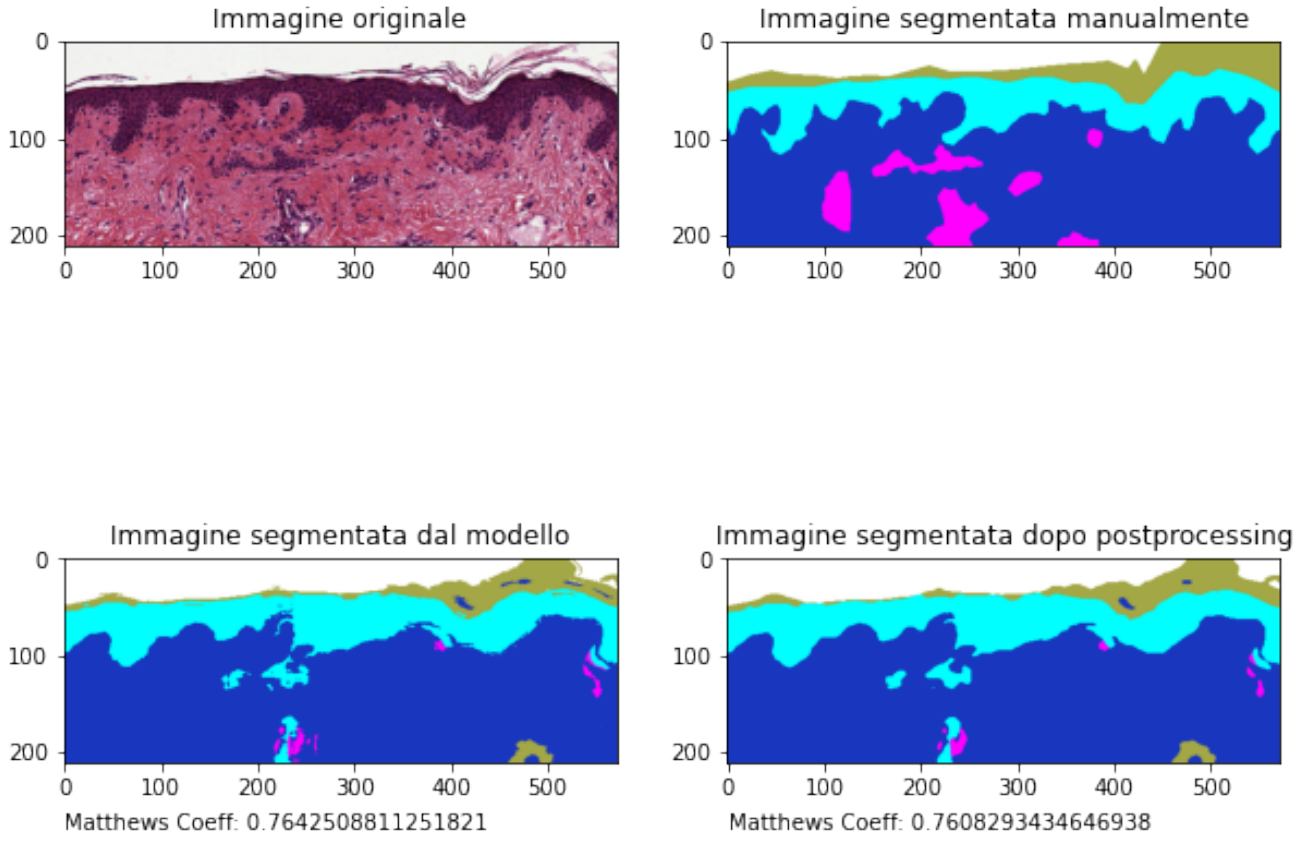


Figura 3.13: Immagine 1001 utilizzata per il test set sull'algoritmo KNN

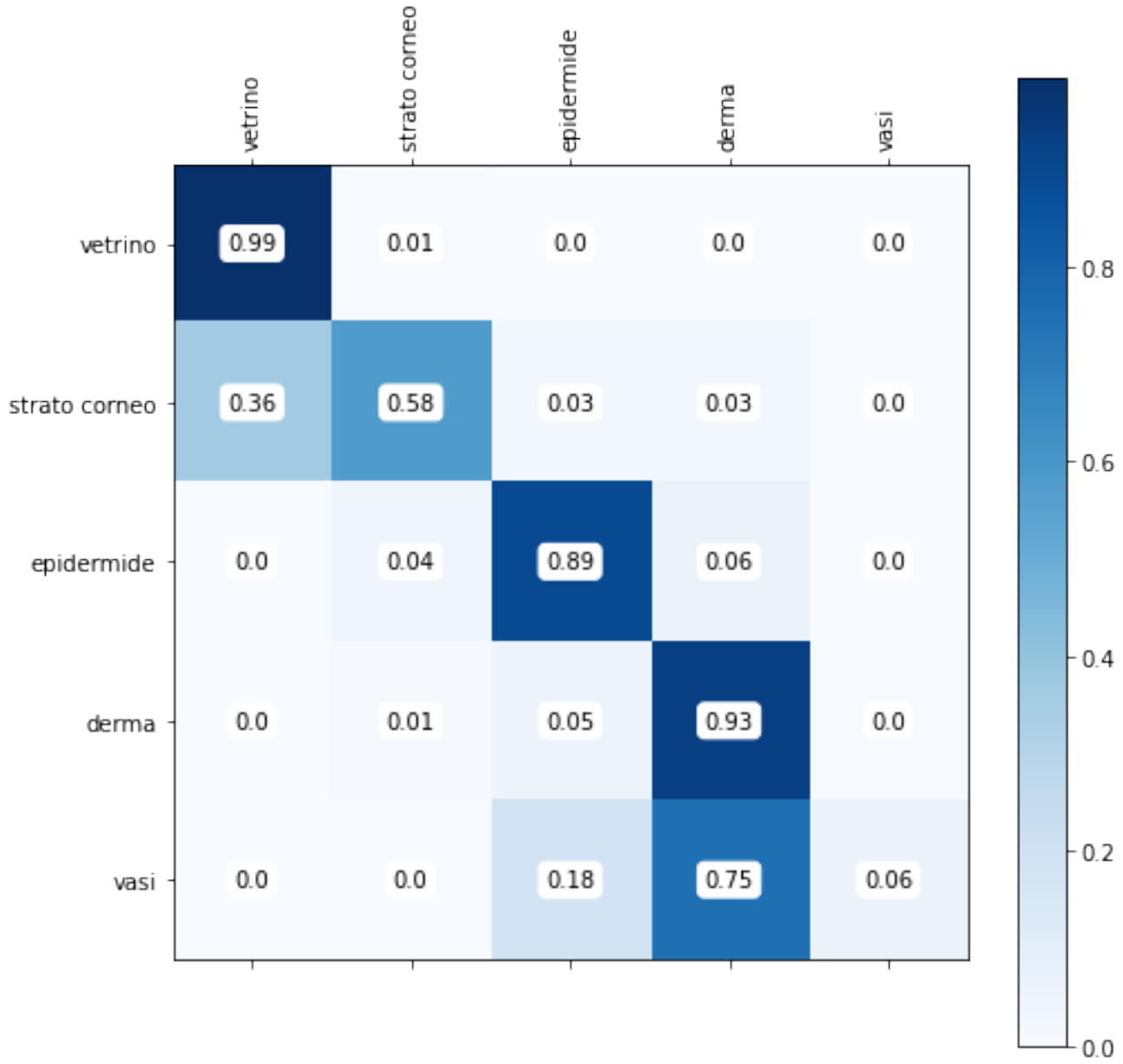


Figura 3.14: Matrice di confusione dell'immagine 1001 sull'algoritmo KNN

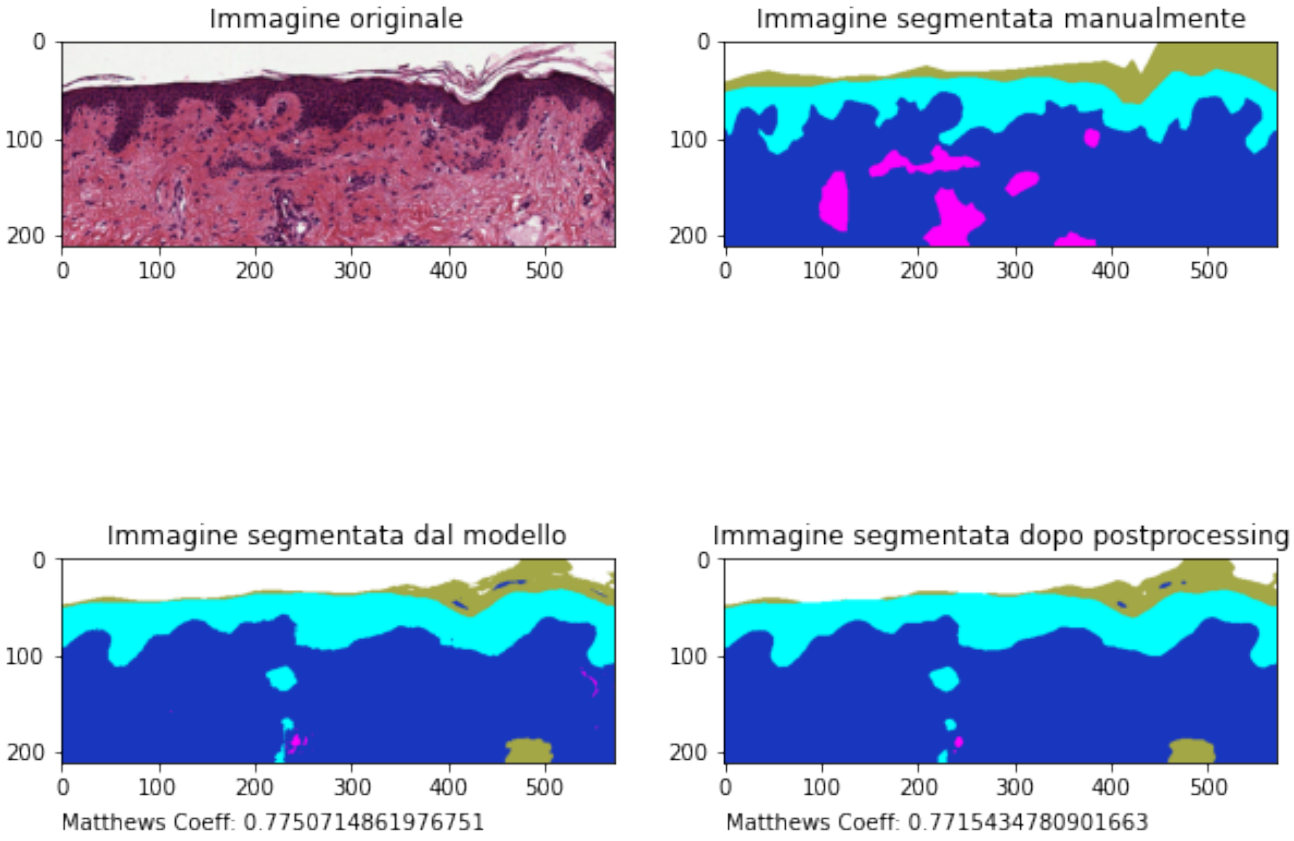


Figura 3.15: Immagine 1001 utilizzata per il test set sull'algoritmo RF

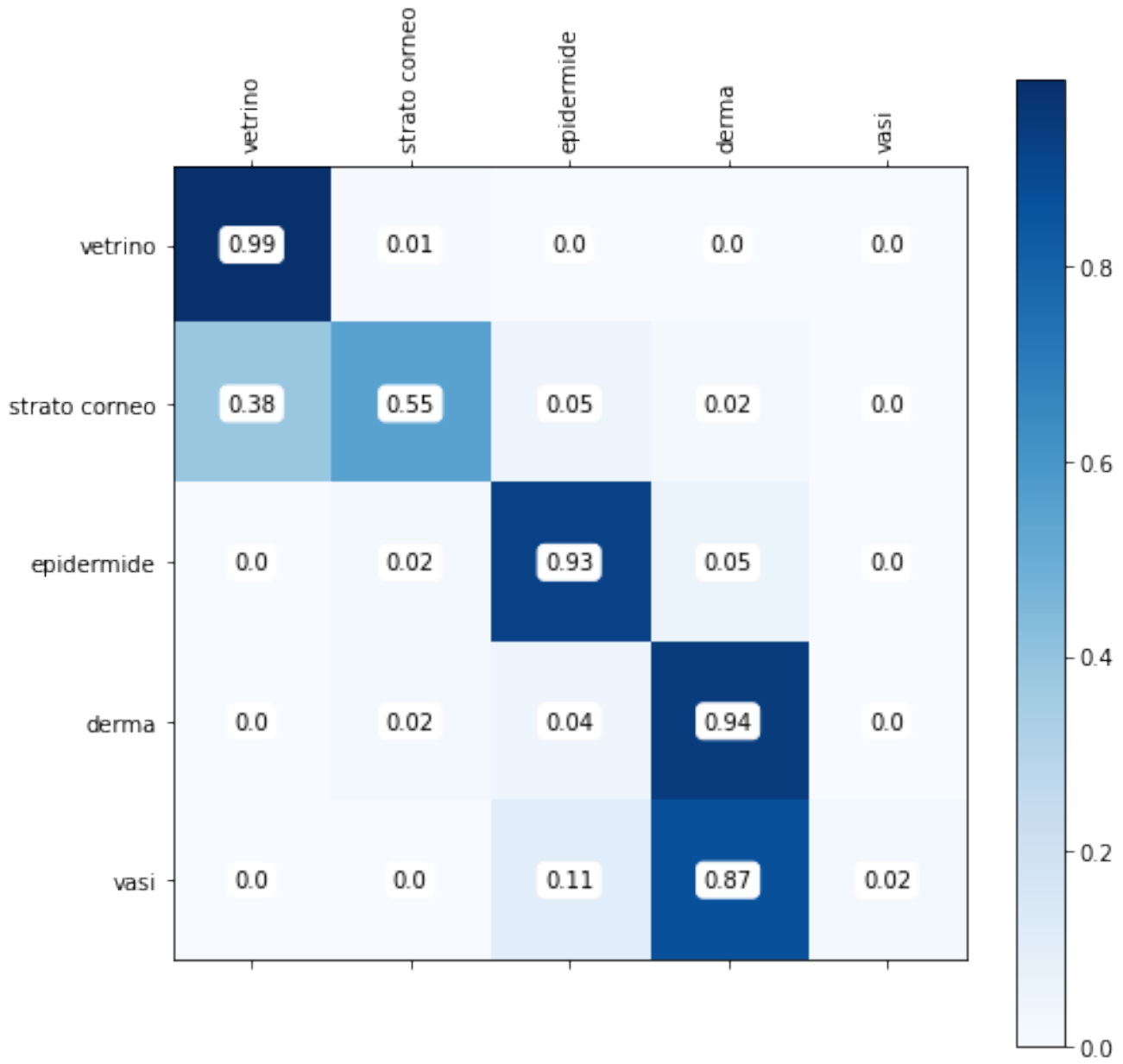


Figura 3.16: Matrice di confusione dell'immagine 1001 sull'algoritmo RF

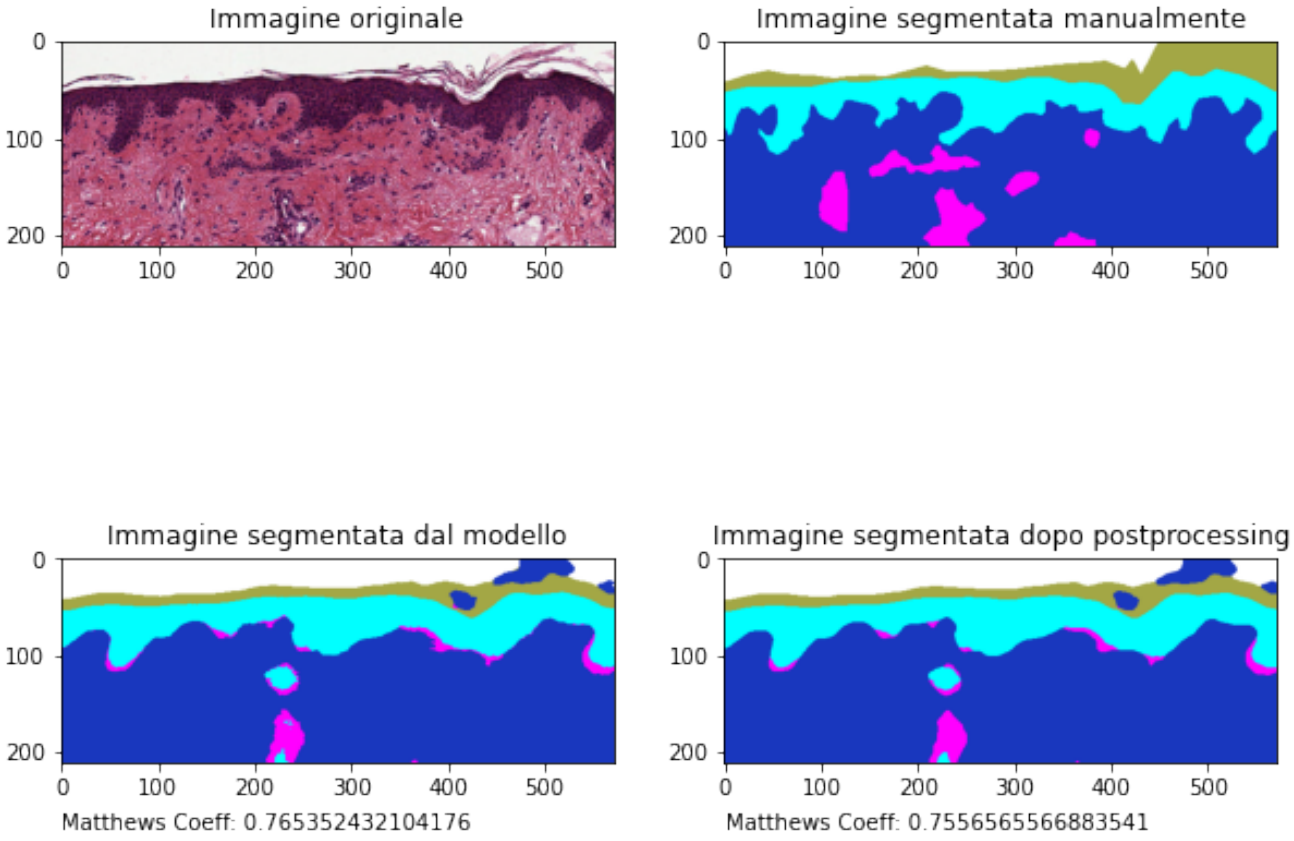


Figura 3.17: Immagine 1001 utilizzata per il test set sull'algoritmo SVM

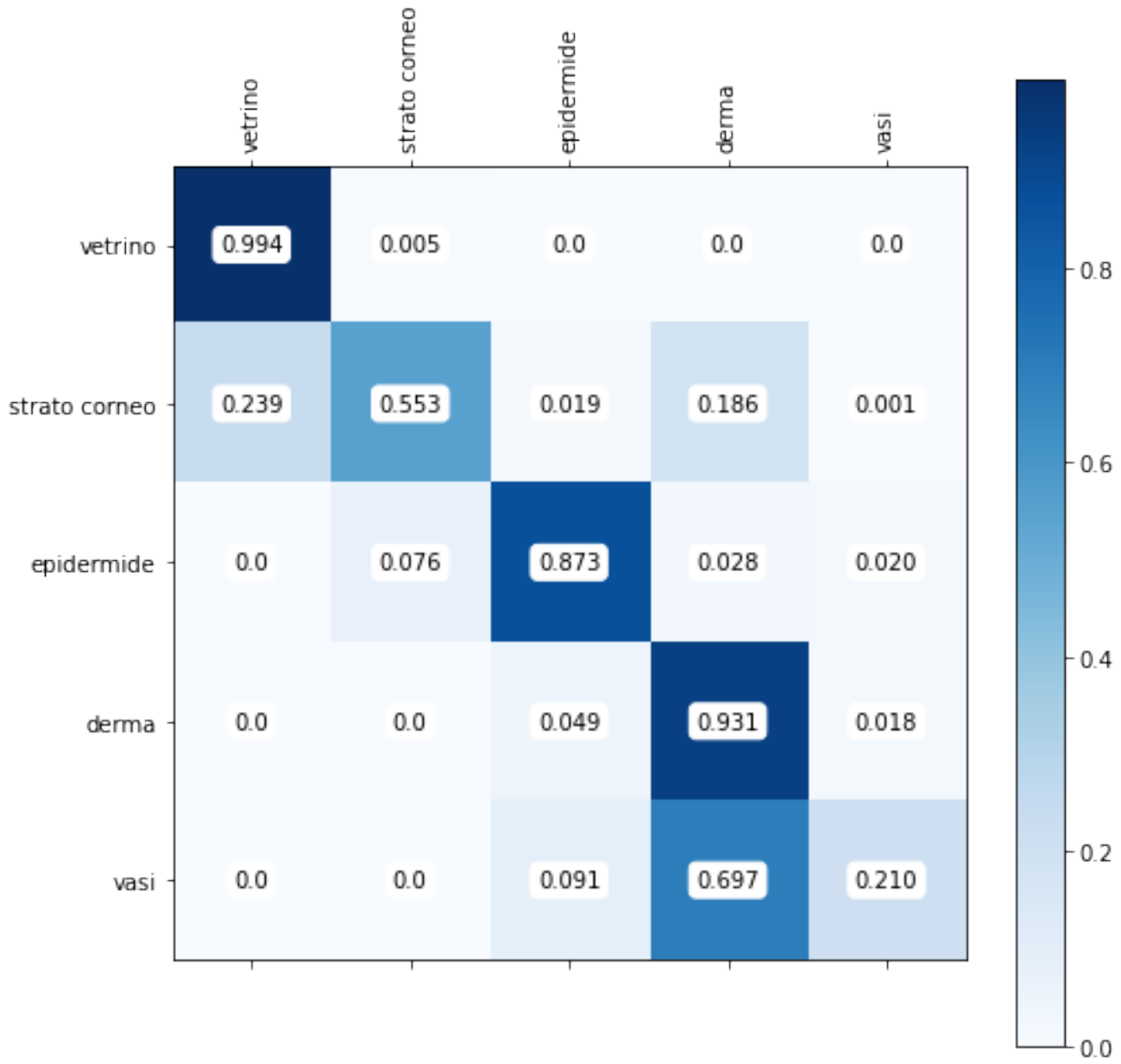


Figura 3.18: Matrice di confusione dell'immagine 1001 sull'algoritmo SVM

Capitolo 4

Conclusioni

Oggetto di questo lavoro di tesi è stato lo sviluppo di modelli di segmentazione semantica automatica di immagini WSI di dermatologia, prese da tessuti di melanoma raccolti in uno studio clinico del Policlinico Sant'Orsola di Bologna, basati sull'algoritmo KNN con particolare attenzione alla ricerca e selezione di features che consentano di ottenere i migliori risultati.

Nella prima parte di questa tesi è stato sviluppato un Background teorico (Cap.1), dove viene riportata la teoria necessaria che è alla base dello sviluppo del lavoro eseguito in questa tesi.

Nel capitolo 2 invece è riportato tutto il processo di lavoro che ci ha consentito di addestrare i nostri modelli. Siamo partiti da immagini segmentate a mano di patch di derma, suddividendole in cinque classi, rispettivamente: vetrino, strato corneo, epidermide, derma e vasi sanguigni. Successivamente attraverso l'utilizzo di vari filtri che sono stati calcolati su diversi spazi di colore (si rimanda alla sezione 2.2 per i dettagli), abbiamo ottenuto una pila di livelli per ogni immagine in cui ogni livello rappresenta una feature.

Successivamente abbiamo eseguito una grid searh sulla base di un training set e un validation set per ottenere il miglior modello possibile (successivamente valutato su un apposito test set) per la generazione delle segmentazioni automatiche.

L'idea di base in questo lavoro è stata quella di confrontare le performance dei vari algoritmi, al fine di utilizzarli nella segmentazione automatica in questo tipo di task.

In base a questi risultati ottenuti è stato evidenziato che nel complesso l'algoritmo KNN segmenta bene in modo automatico le varie classi, risultato non del tutto scontato.

Mentre ci si aspettava una segmentazione più precisa da parte del RF, che invece è risultato poco attendibile.

Quello che si nota è la cattiva performance per i vasi, probabilmente legata alla numerosità nei campioni. Per il modello SVM si è visto che i vasi vengono riconosciuti come transizioni tra chiaro e scuro, ed è per questo che va a segmentare anche il bordo tra l'epidermide e il derma.

L'algoritmo che fa più confusione nel riconoscimento delle classi è proprio il Random Forest.

Questo risultato probabilmente ci fa capire che per questo tipo di immagini l'algoritmo RF fa più fatica rispetto a metodi di tipo lineare come l'SVM e metodi locali come il KNN. Si nota inoltre una forte correlazione tra le diverse variabili: questo perchè il RF è pessimo quando si vuole considerare la correlazione tra variabili. In altre parole, se ciascuna variabile è significativa è possibile ricostruire delle patch in modo preciso, al contrario se sono presenti forti correlazioni e forti collinearità tra le variabili allora il RF fatica a individuarle.

Possiamo inoltre concludere che se vengono segmentate correttamente un numero limitato di immagini, in futuro questi metodi potranno essere impiegati per un addestramento semi-supervisionato dell'algoritmo.

Potenziali sviluppi futuri di questa ricerca possono essere il miglioramento dei modelli supervisionati per la segmentazione automatica ed auspicabilmente il passaggio successivo all'utilizzo di modelli non supervisionati, evoluzione che consentirebbe di alleggerire ulteriormente il carico di lavoro, anche in termini di tempo, per l'operatore medico.

Appendice A

Appendice

In questa sezione si allegano tutte le restanti figure (101, 102, 103, 105) allenate sul modello Knn e confrontate sugli altri due modelli (SVM e RF) con le rispettive matrici di confusione.

Si riporta anche la tabella della grid search.

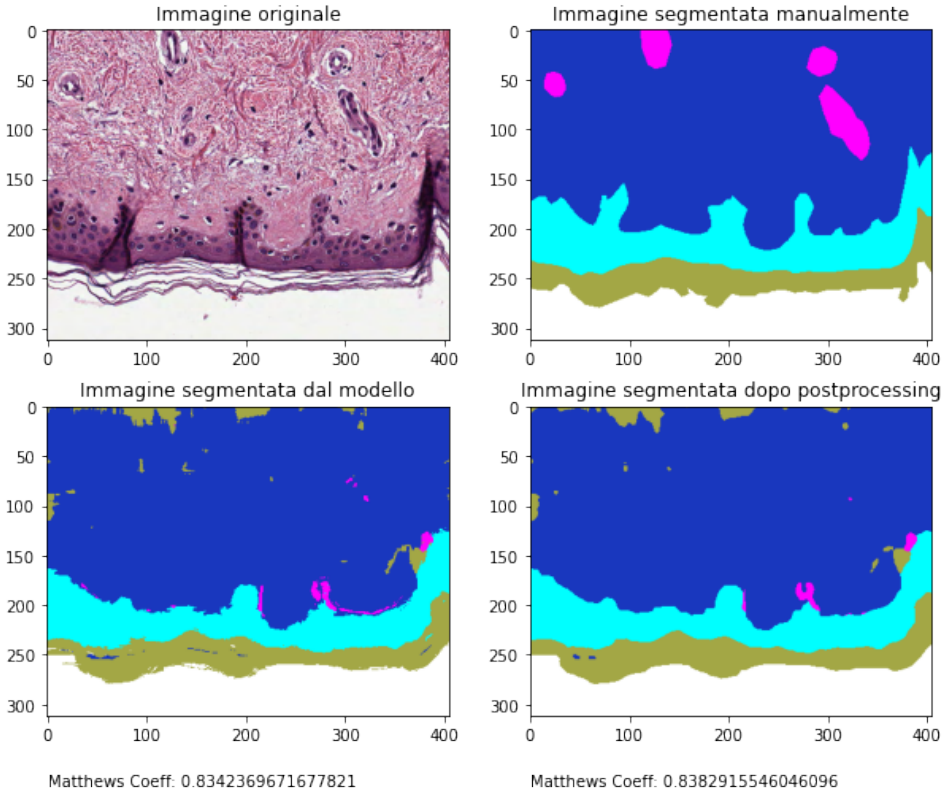


Figura A.1: Immagine 101 utilizzata per il test set sull'algoritmo KNN

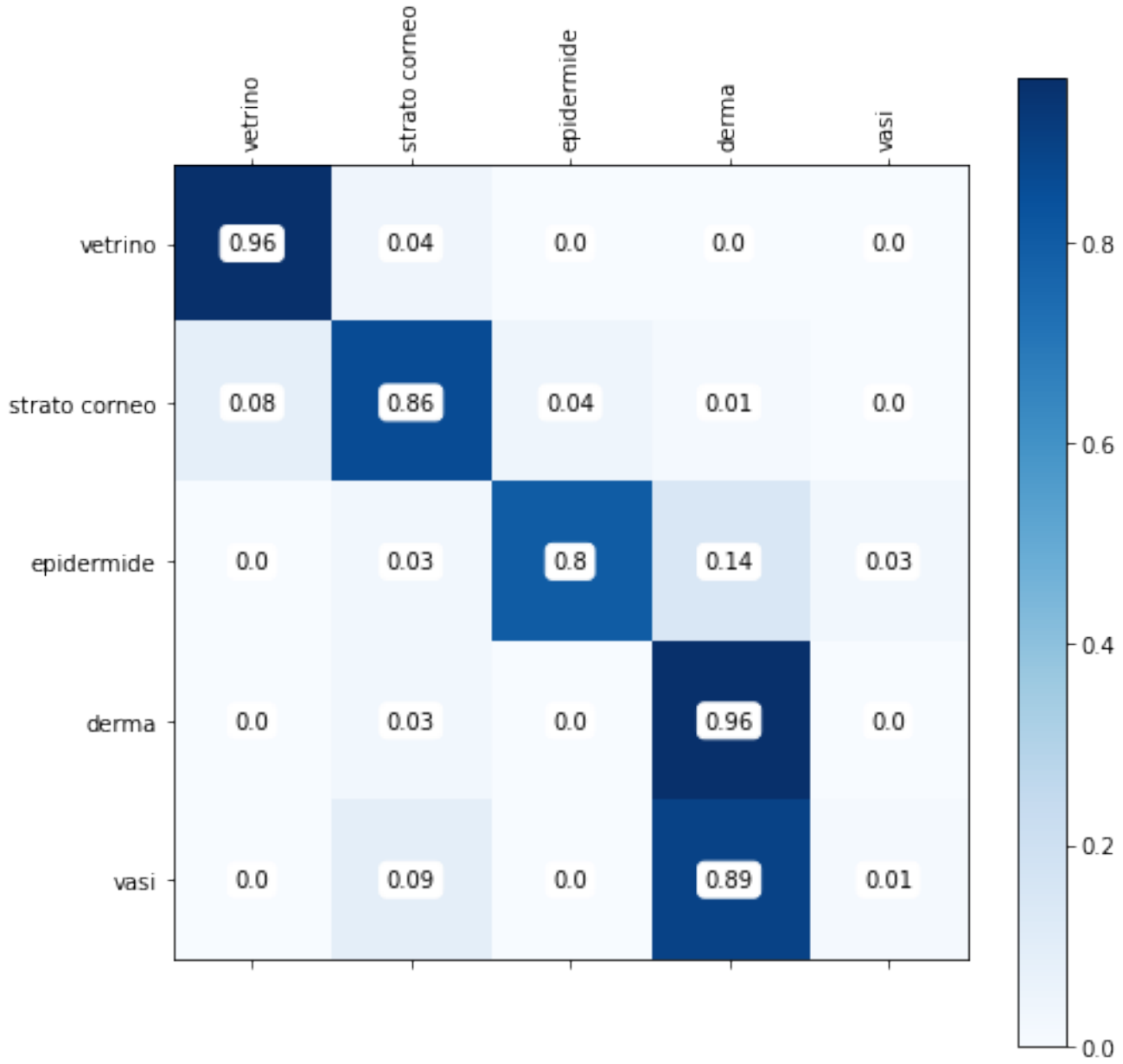


Figura A.2: Matrice di confusione dell'immagine 101 sull'algoritmo KNN

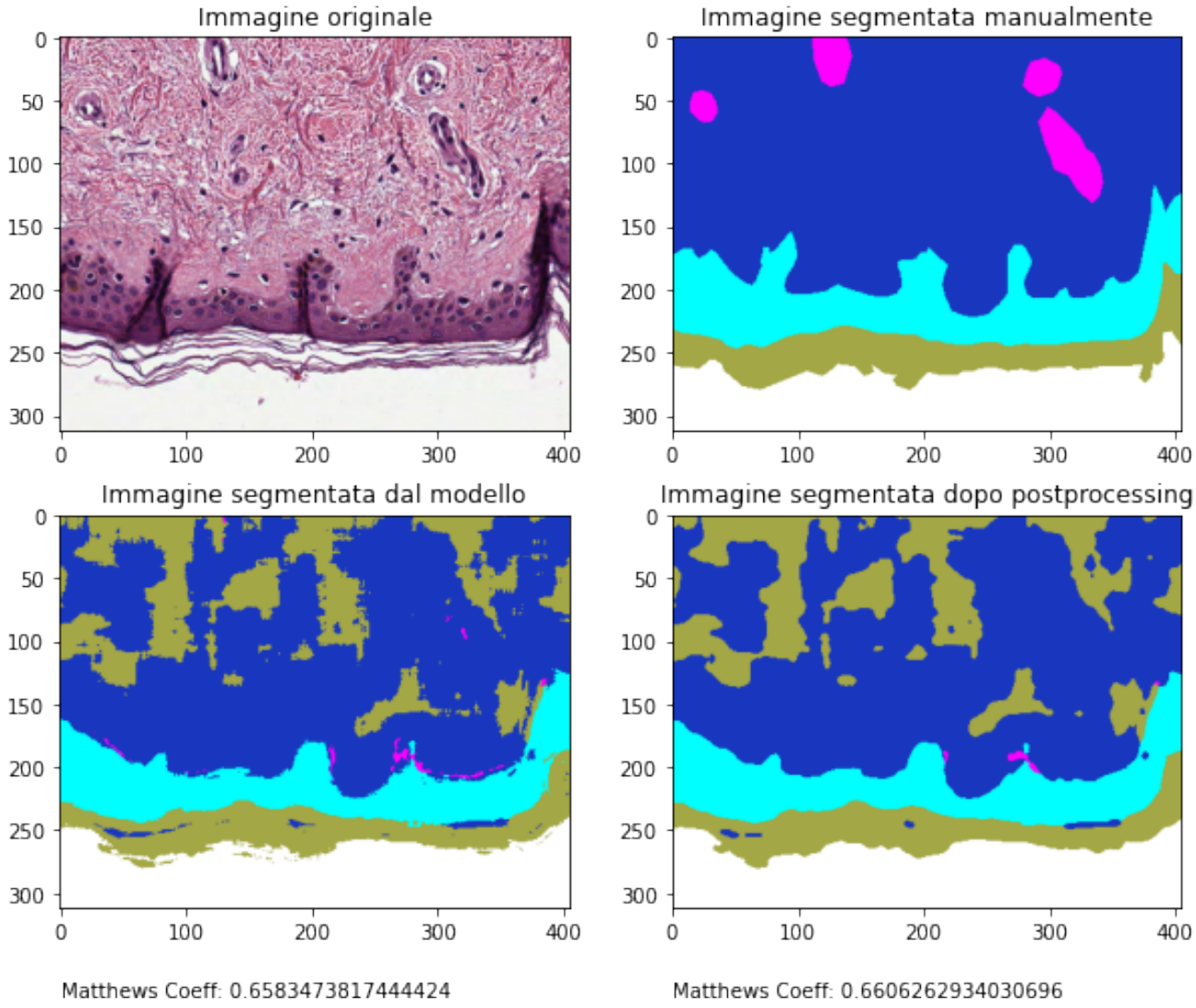


Figura A.3: Immagine 101 utilizzata per il test set sull'algoritmo RF

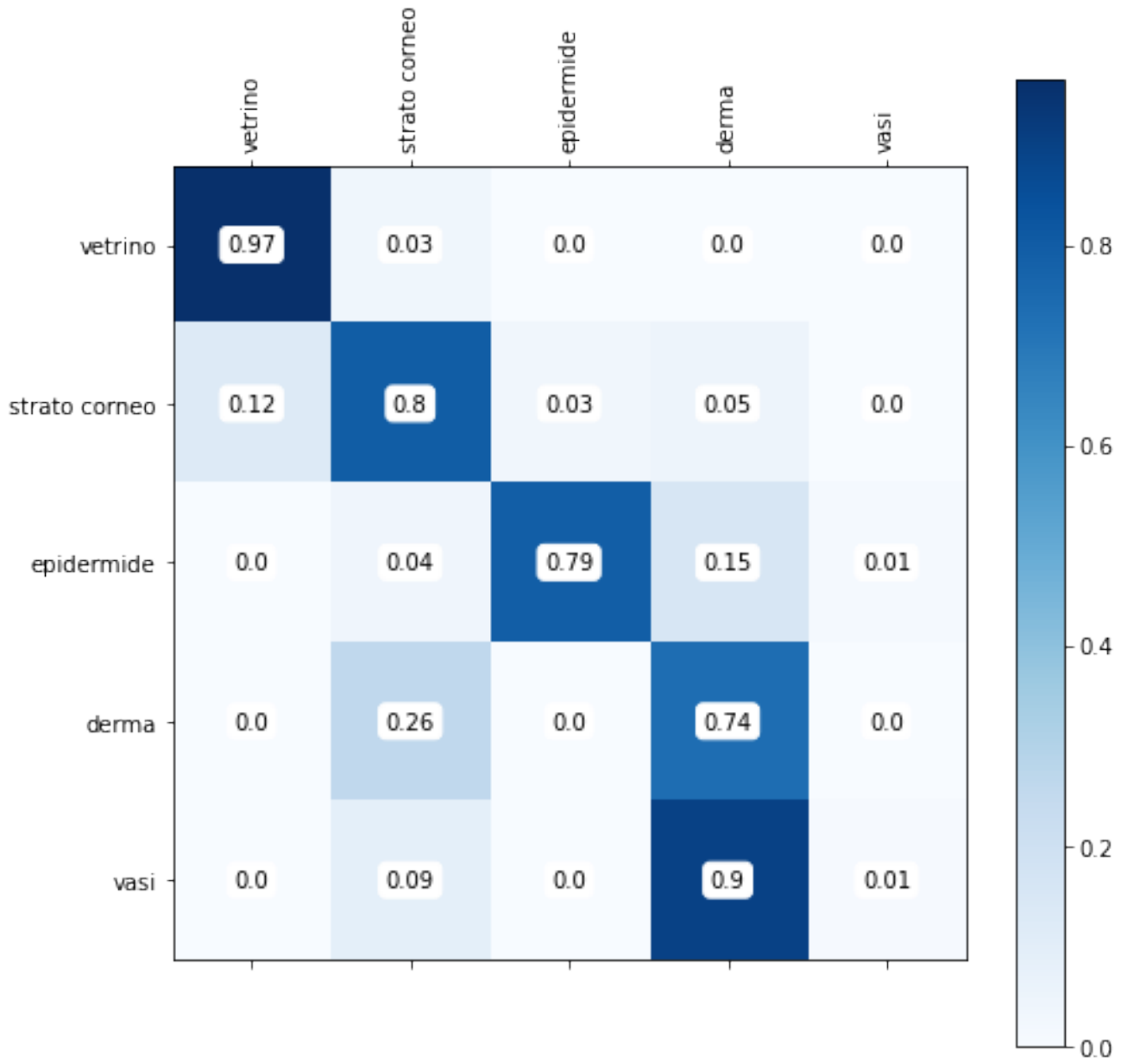


Figura A.4: Matrice di confusione dell'immagine 101 sull'algoritmo RF

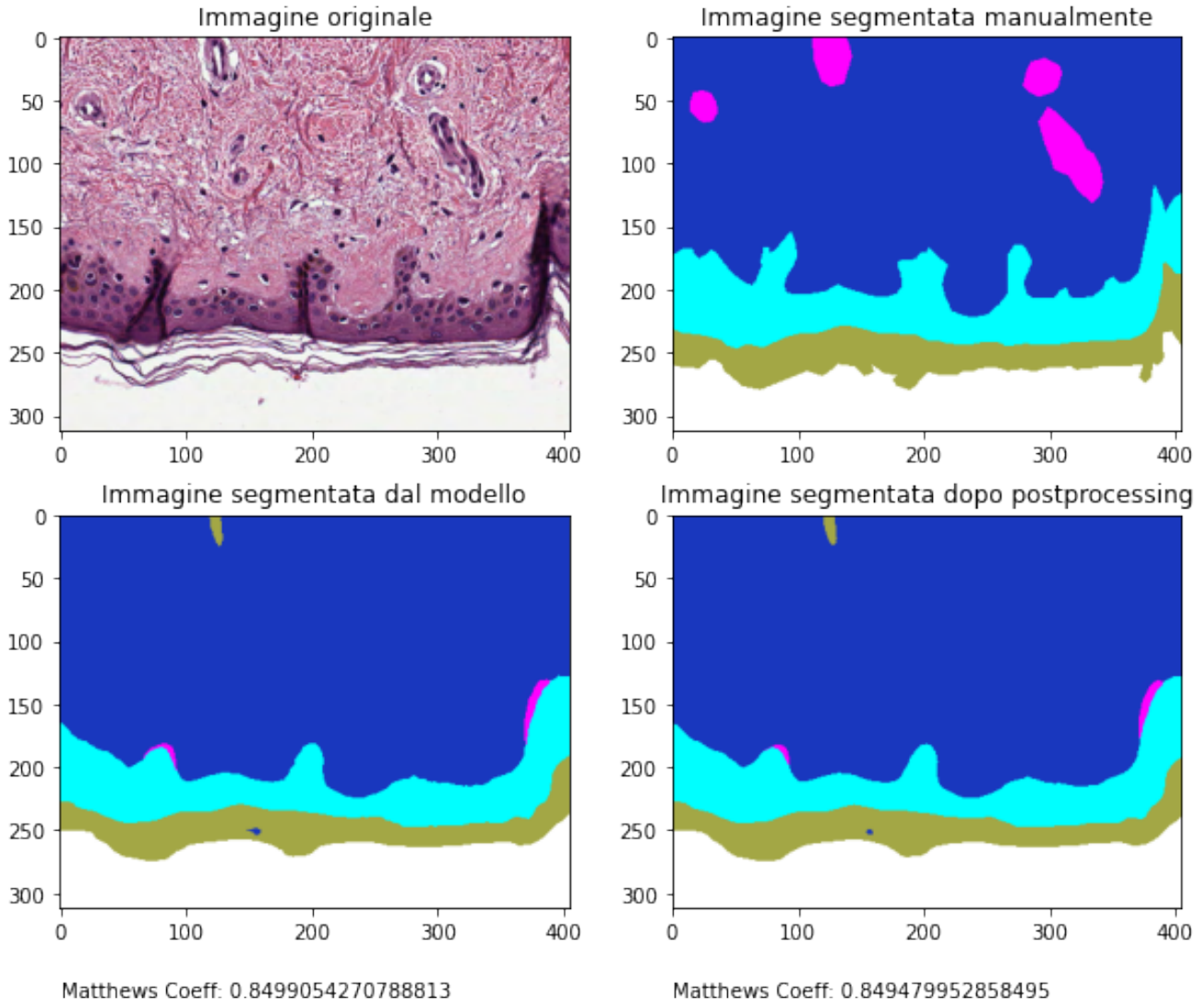


Figura A.5: Immagine 101 utilizzata per il test set sull'algoritmo SVM

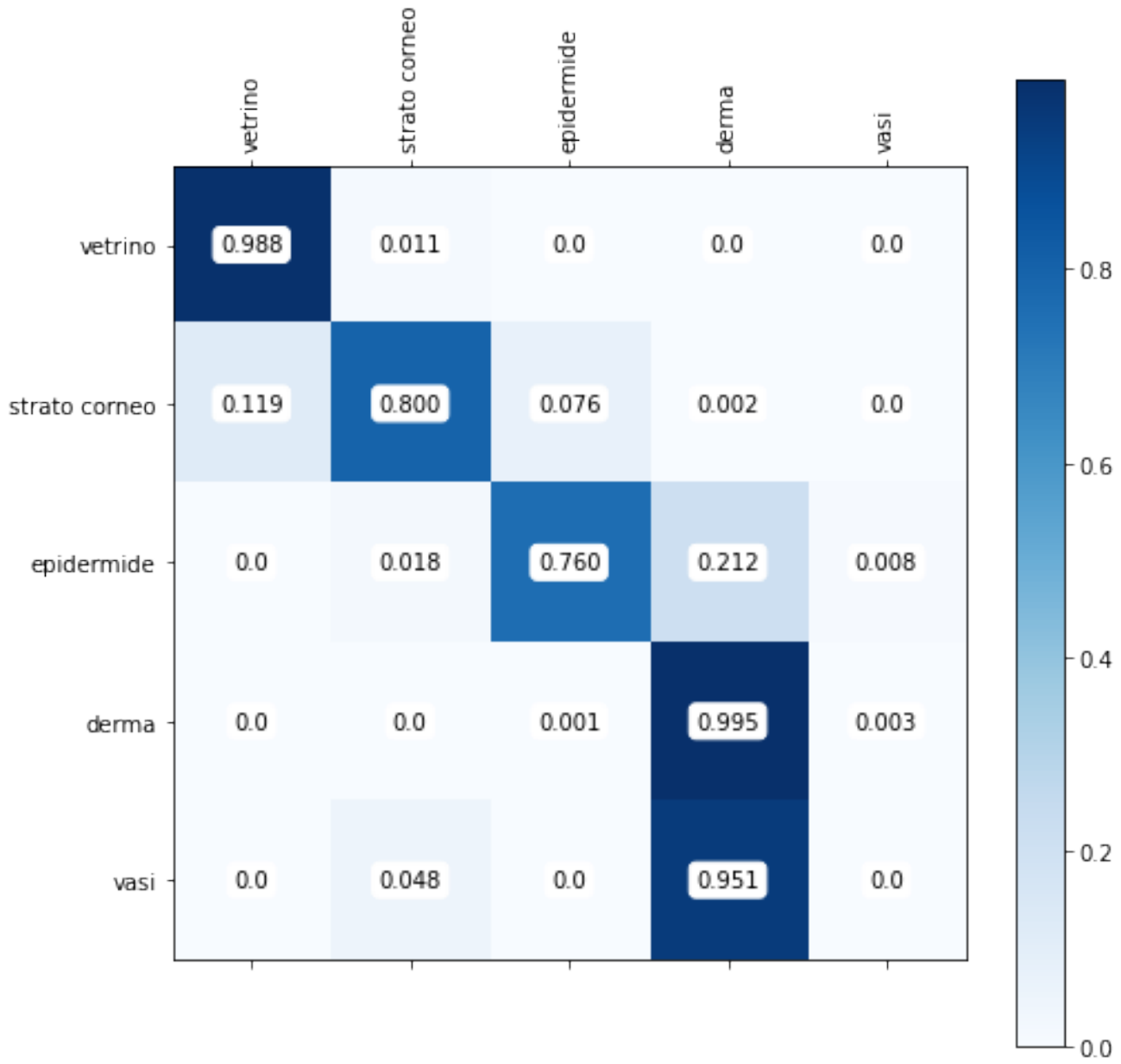


Figura A.6: Matrice di confusione dell'immagine 101 sull'algoritmo SVM

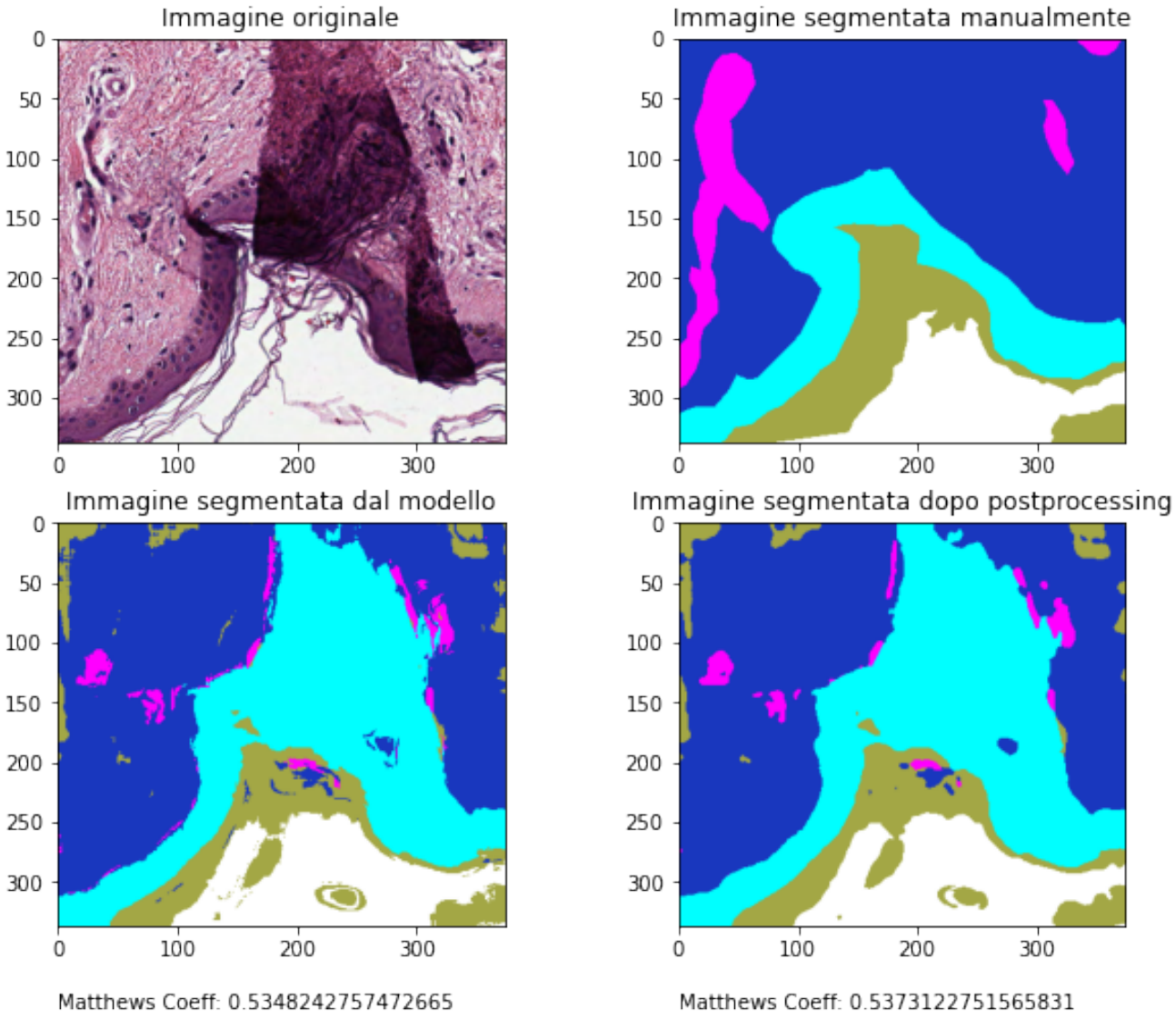


Figura A.7: Immagine 102 utilizzata per il test set sull'algoritmo KNN

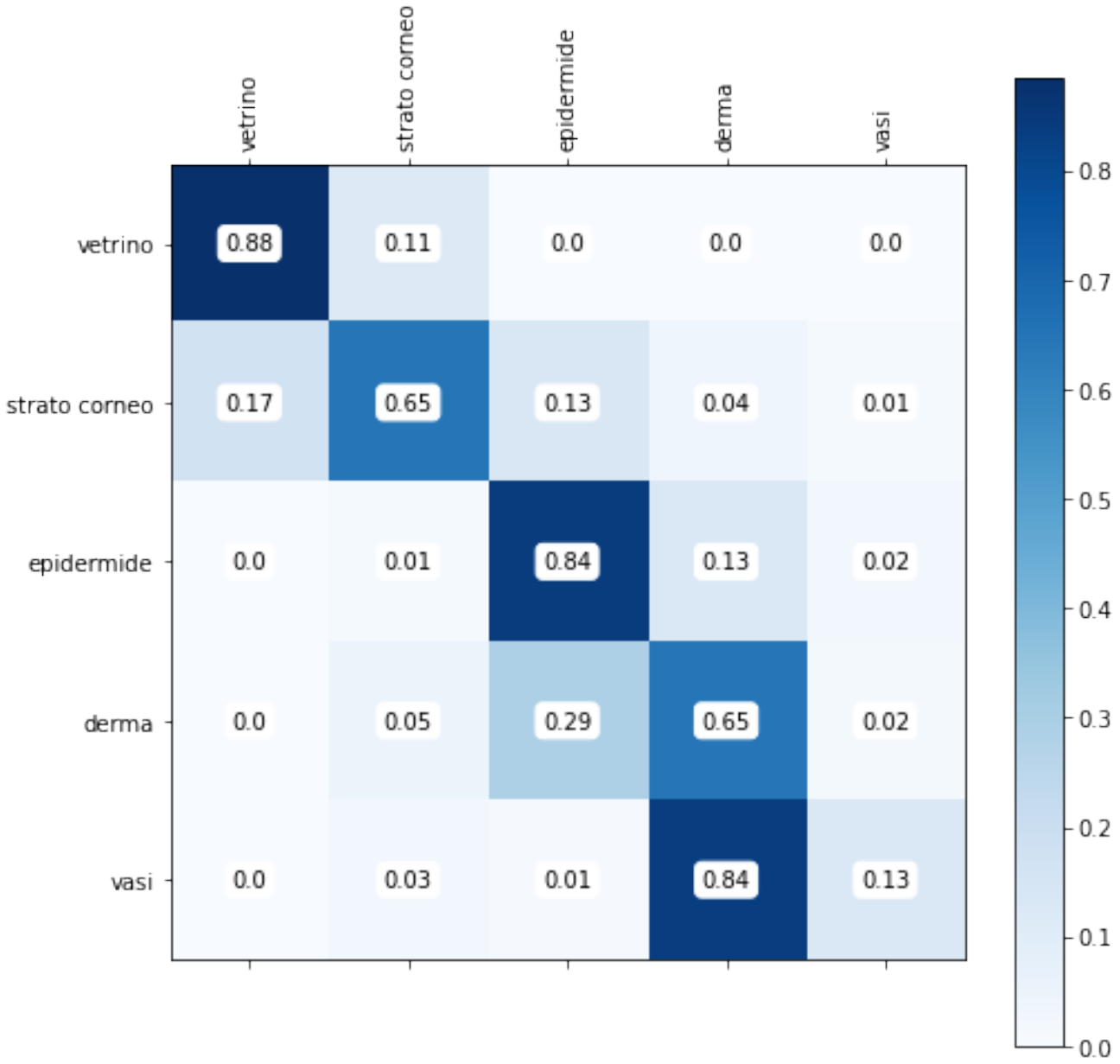


Figura A.8: Matrice di confusione dell'immagine 102 sull'algoritmo KNN

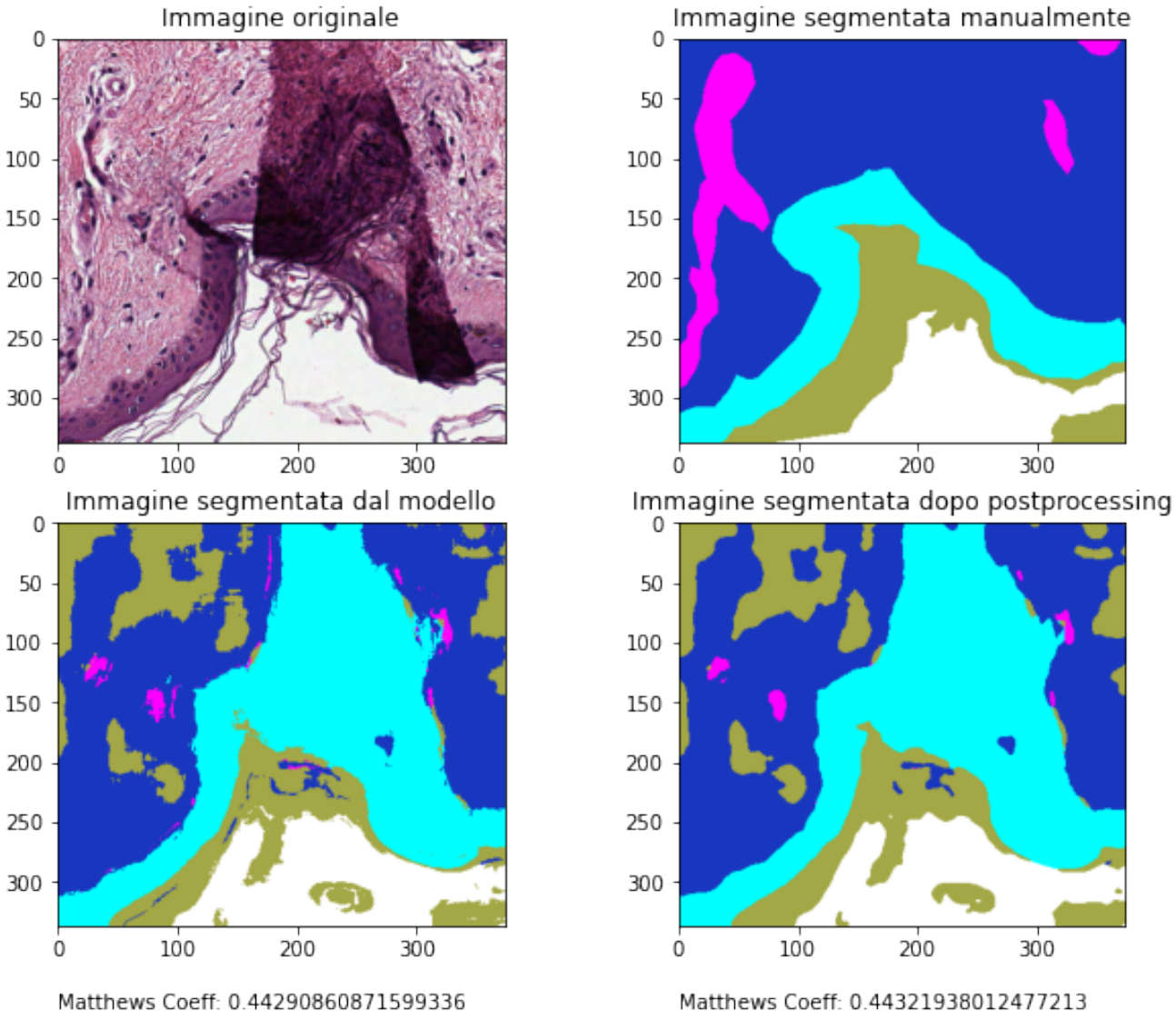


Figura A.9: Immagine 102 utilizzata per il test set sull'algoritmo RF

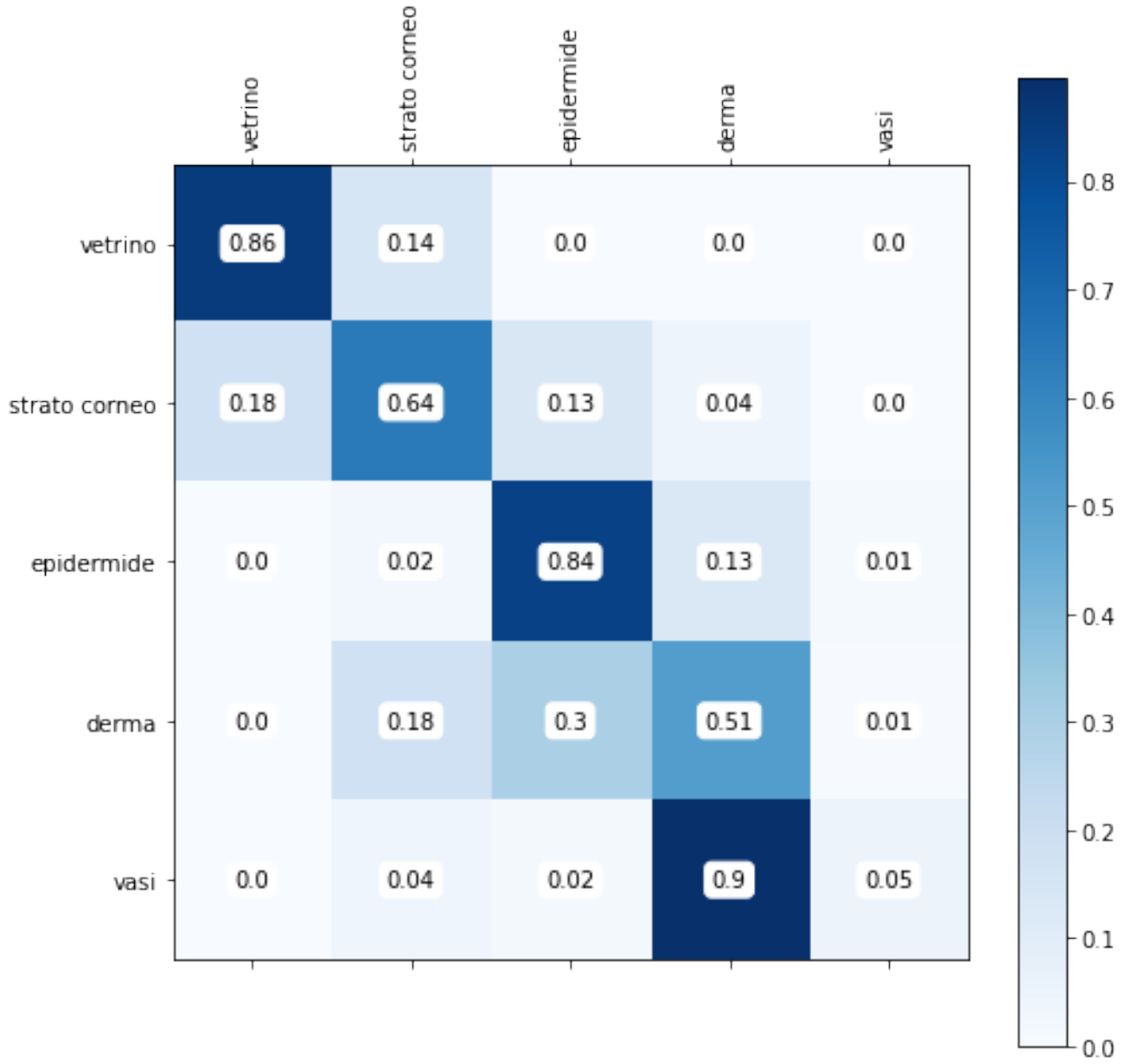


Figura A.10: Matrice di confusione dell'immagine 102 sull'algoritmo RF

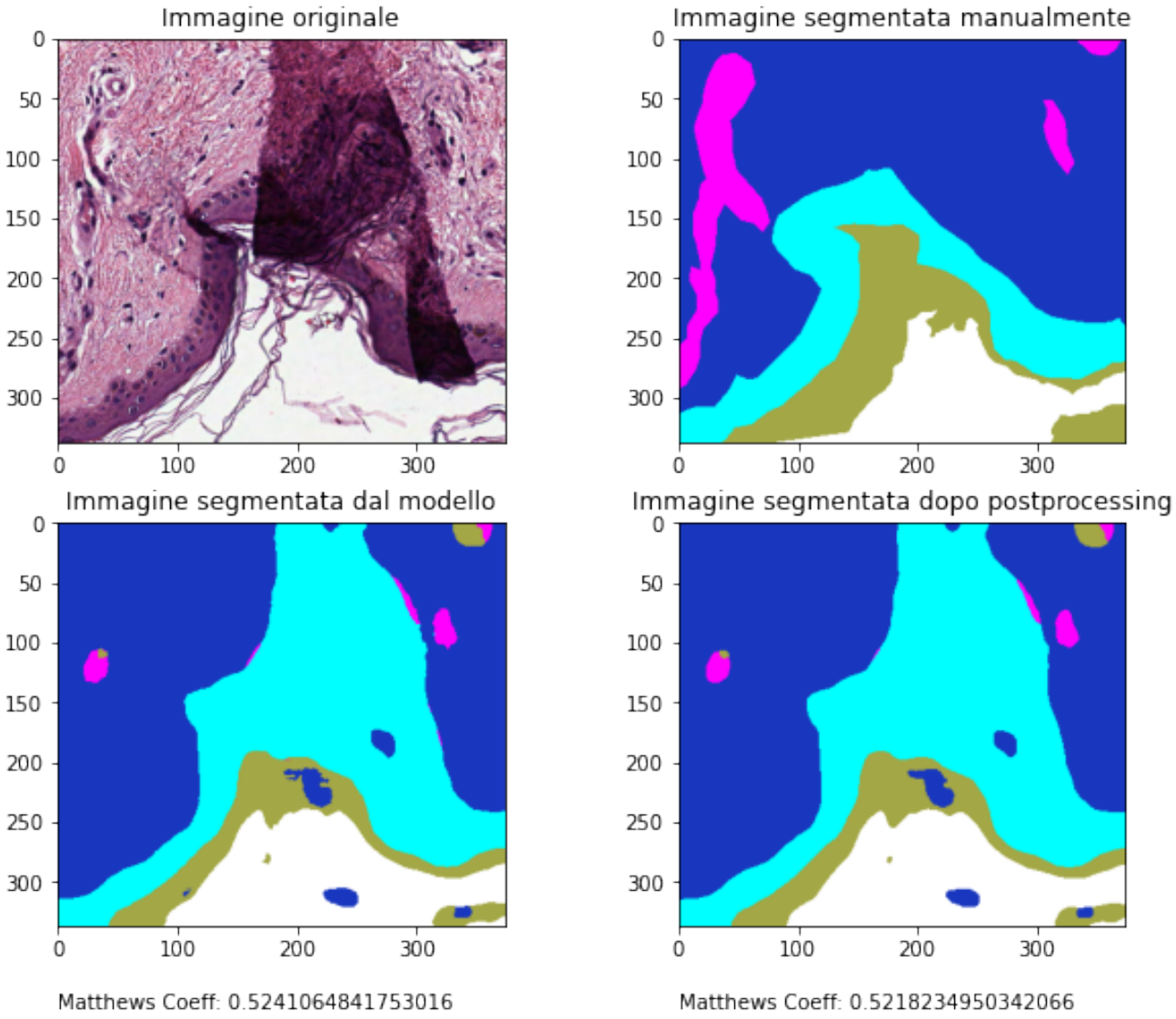


Figura A.11: Immagine 102 utilizzata per il test set sull'algoritmo SVM

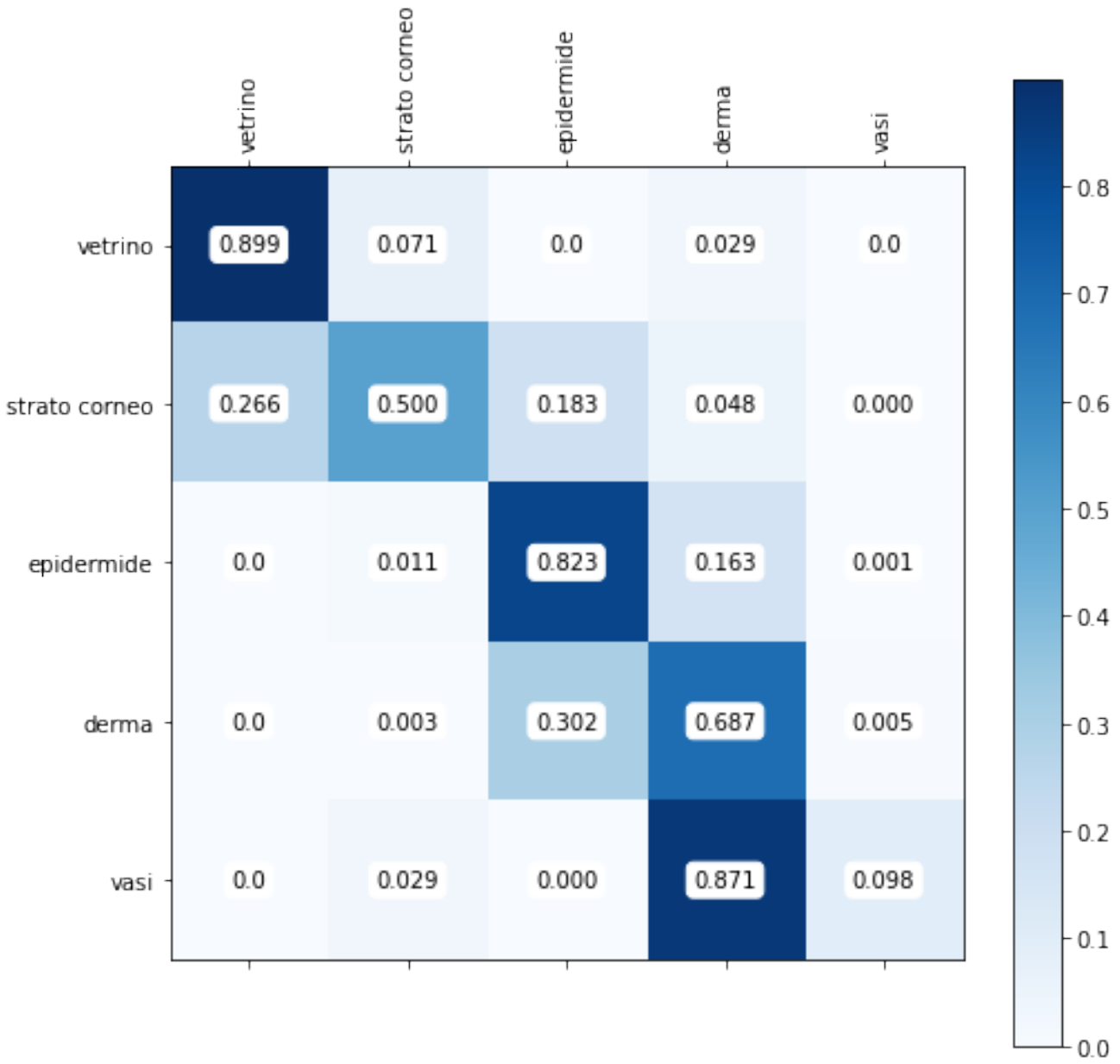


Figura A.12: Matrice di confusione dell'immagine 102 sull'algoritmo SVM

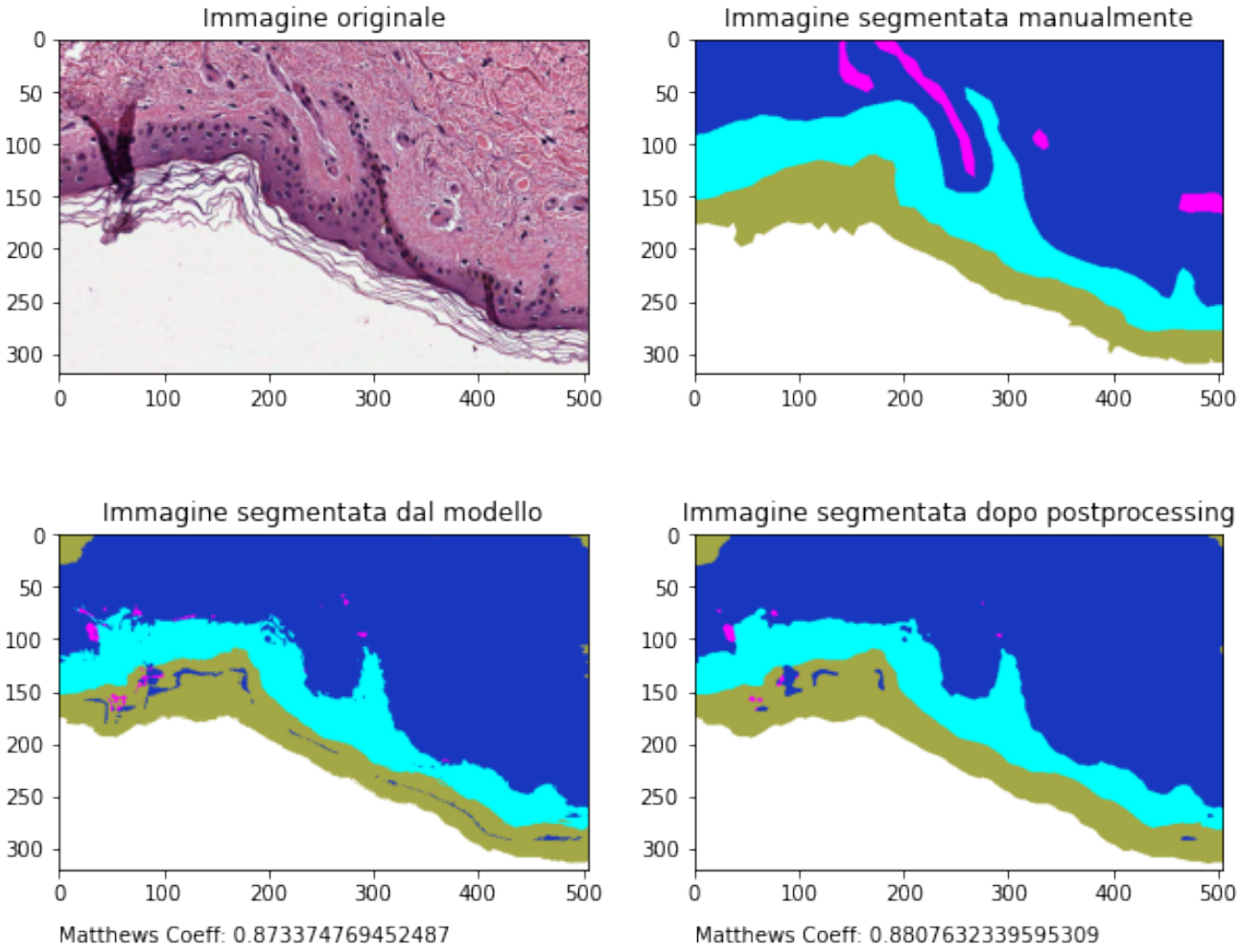


Figura A.13: Immagine 103 utilizzata per il test set sull'algoritmo KNN

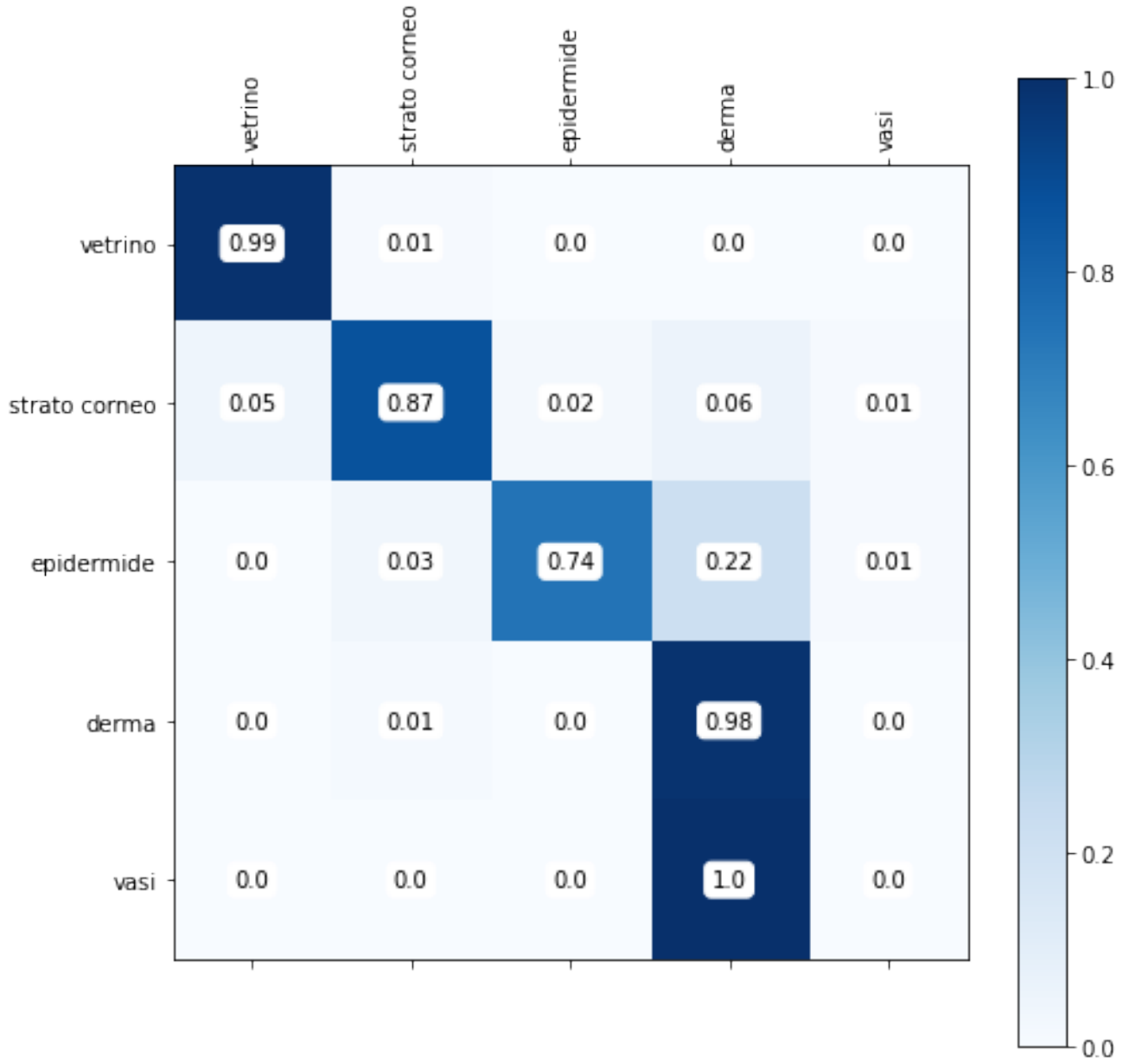


Figura A.14: Matrice di confusione dell'immagine 103 sull'algoritmo KNN

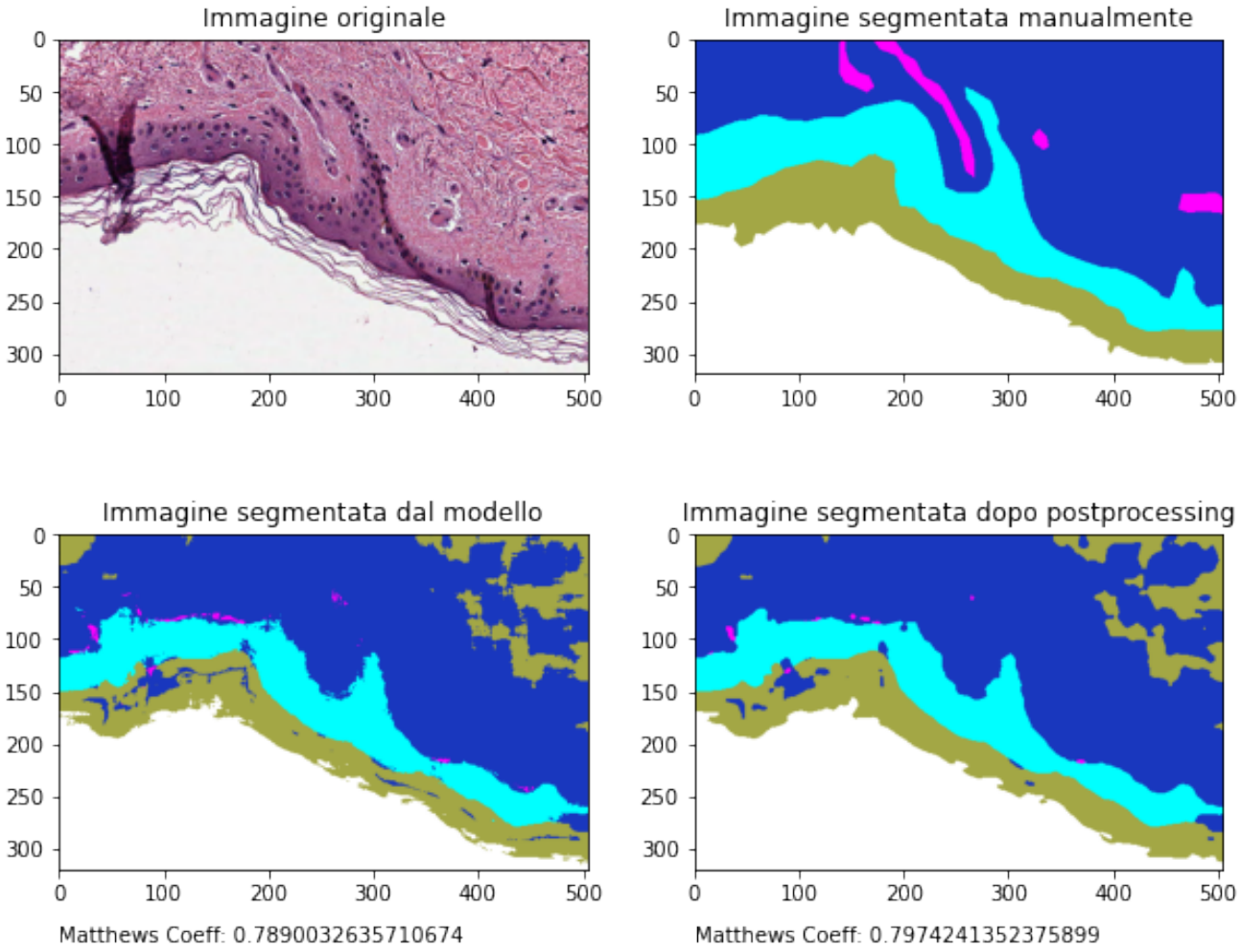


Figura A.15: Immagine 103 utilizzata per il test set sull'algoritmo RF

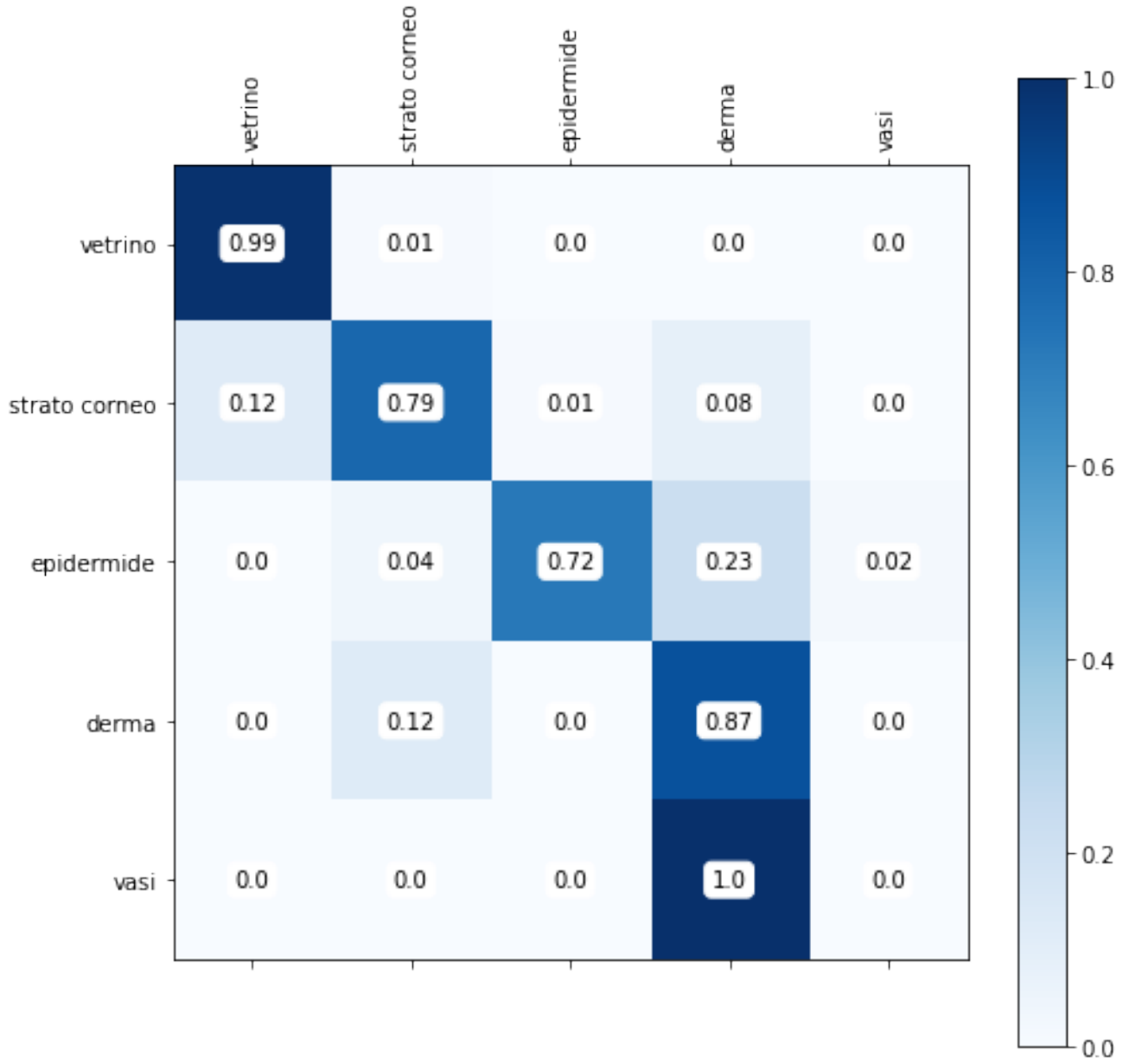


Figura A.16: Matrice di confusione dell'immagine 103 sull'algoritmo RF

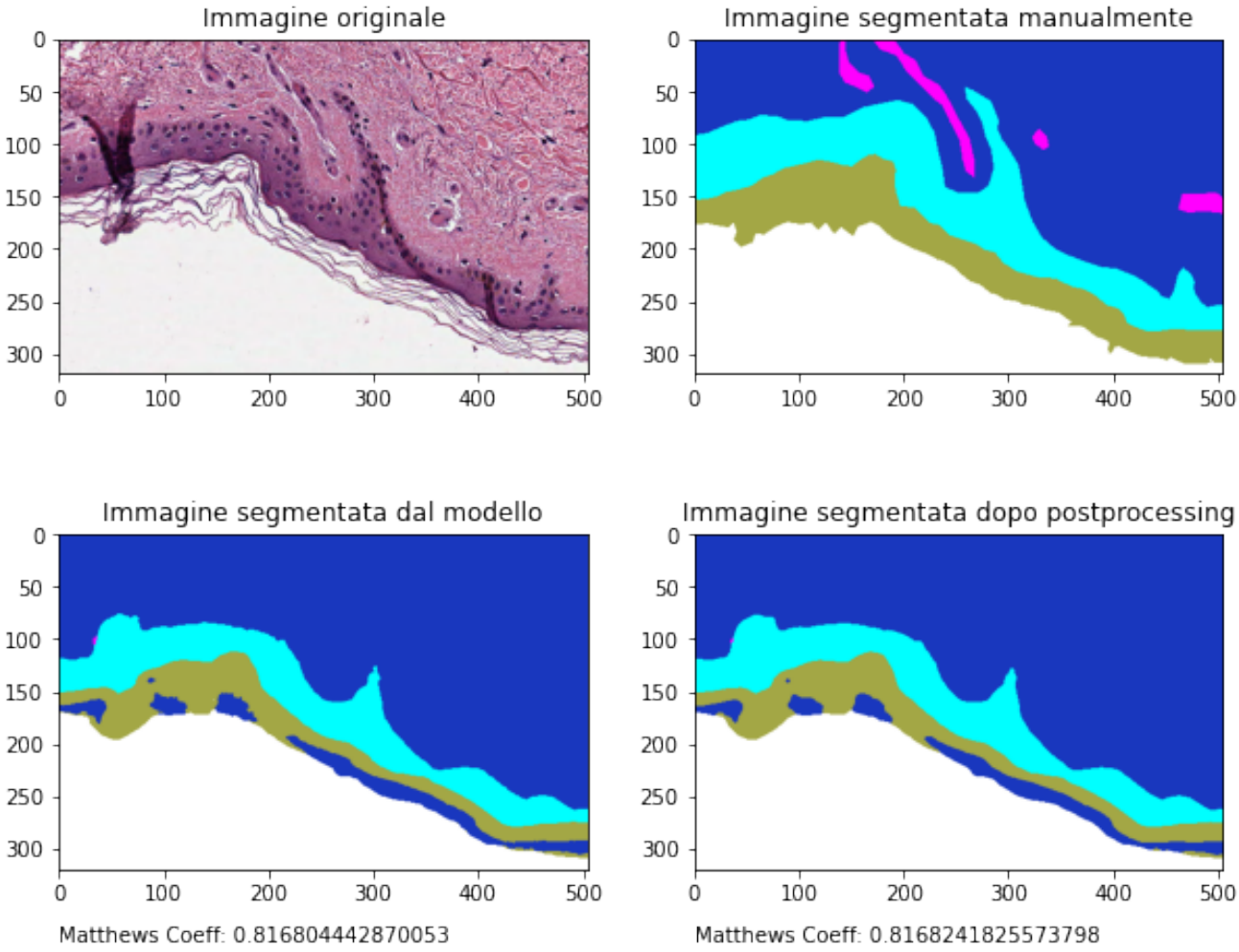


Figura A.17: Immagine 103 utilizzata per il test set sull'algoritmo SVM

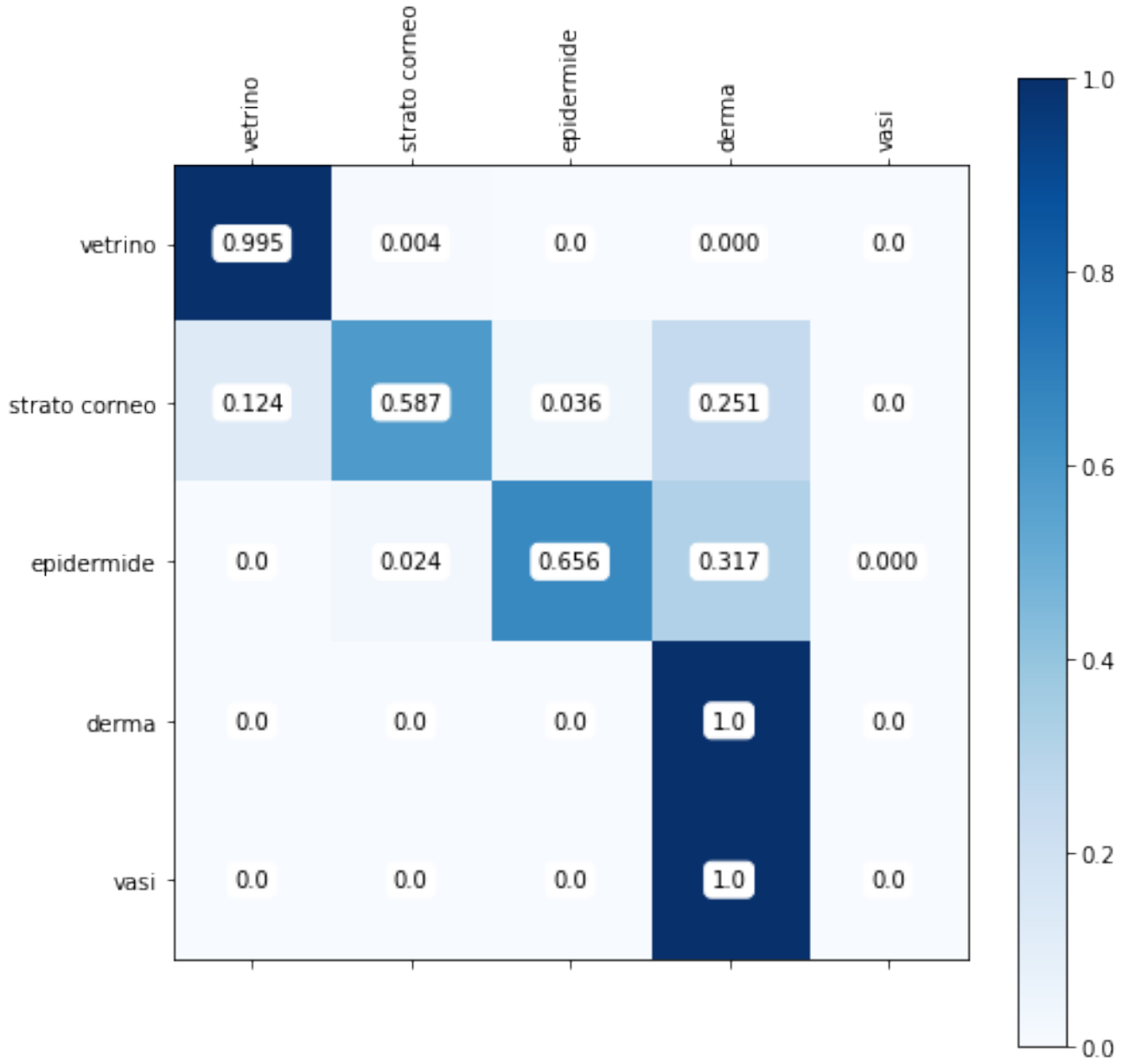


Figura A.18: Matrice di confusione dell'immagine 103 sull'algoritmo SVM

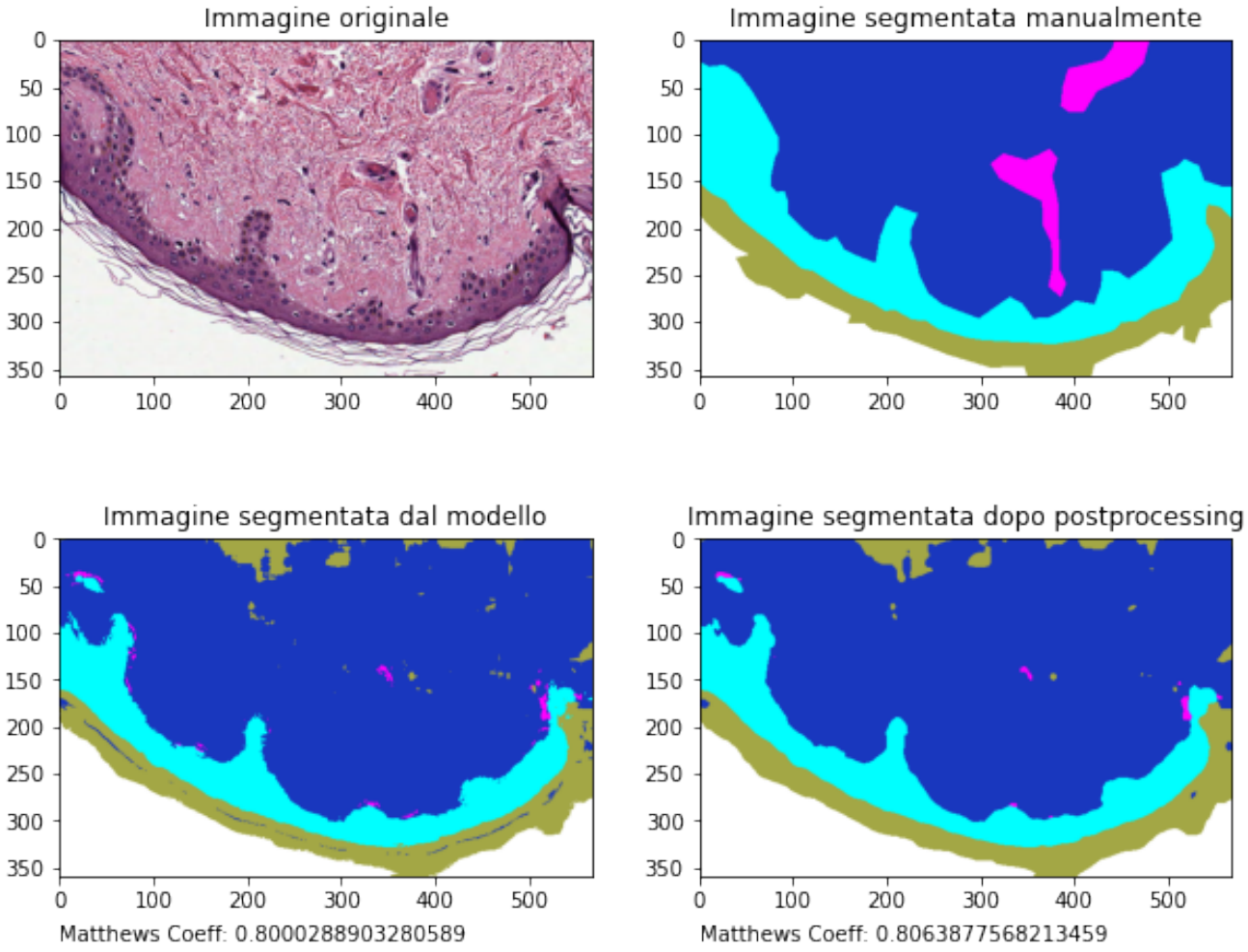


Figura A.19: Immagine 105 utilizzata per il test set sull'algoritmo KNN

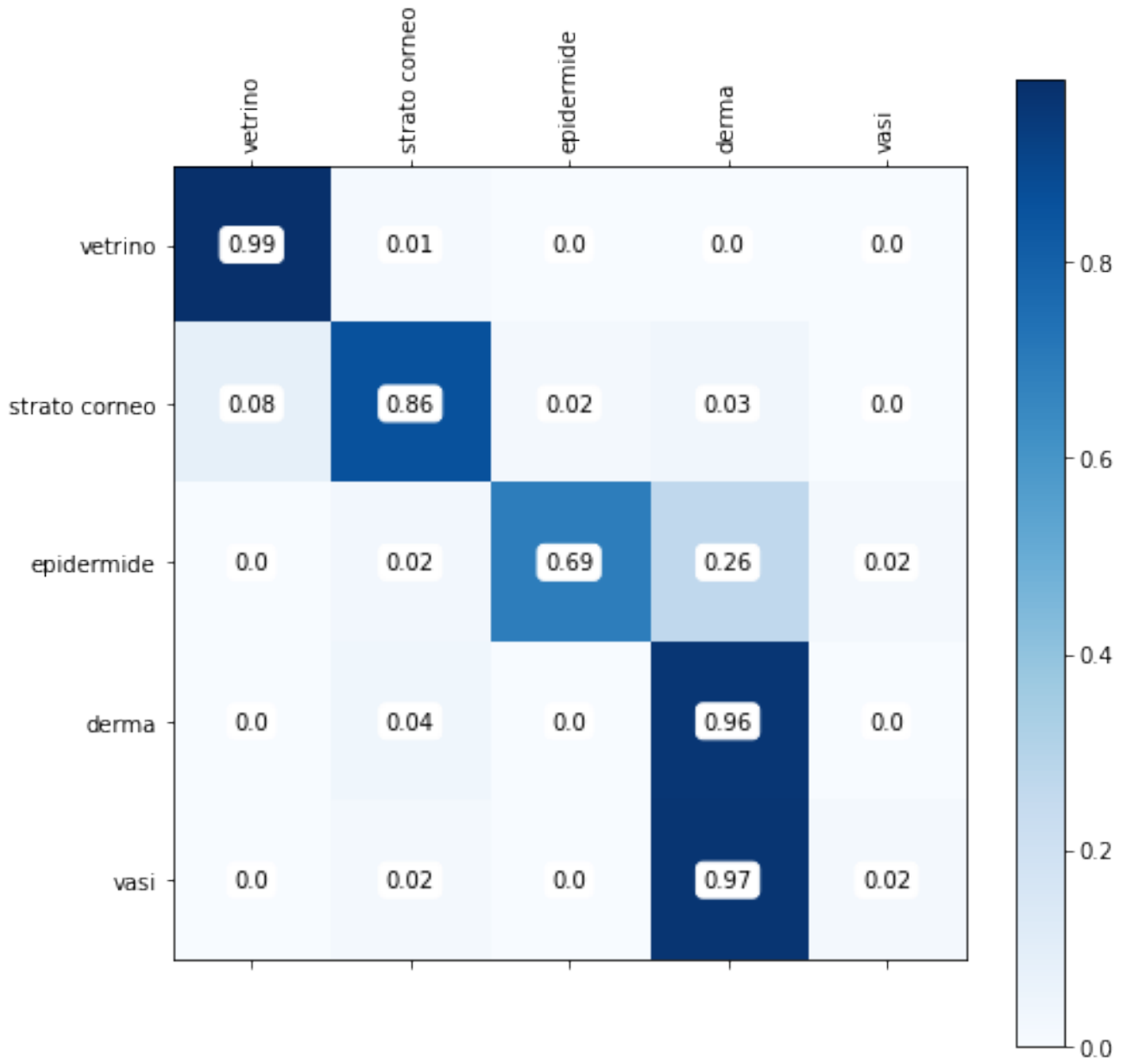


Figura A.20: Matrice di confusione dell'immagine 105 sull'algoritmo KNN

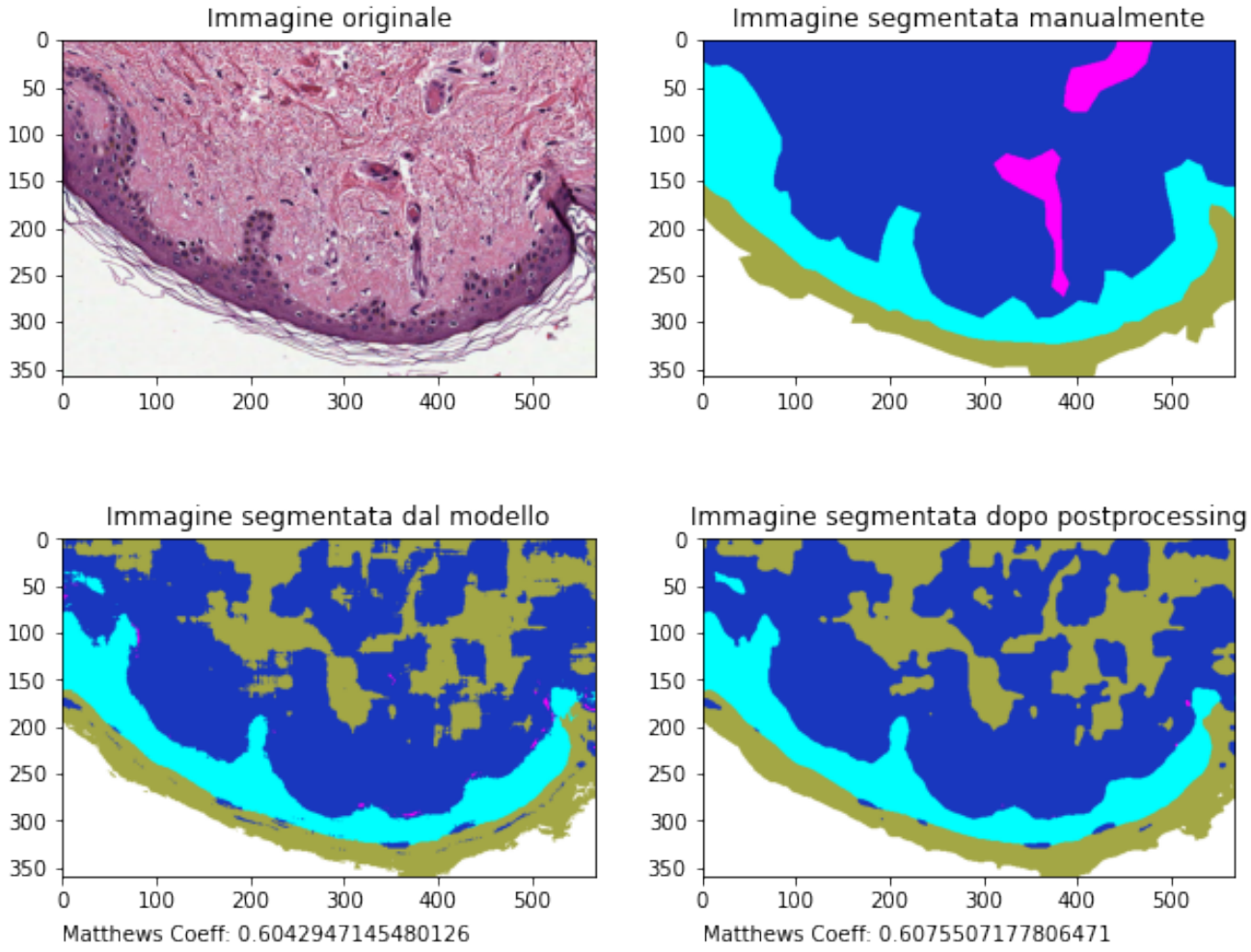


Figura A.21: Immagine 105 utilizzata per il test set sull'algoritmo RF

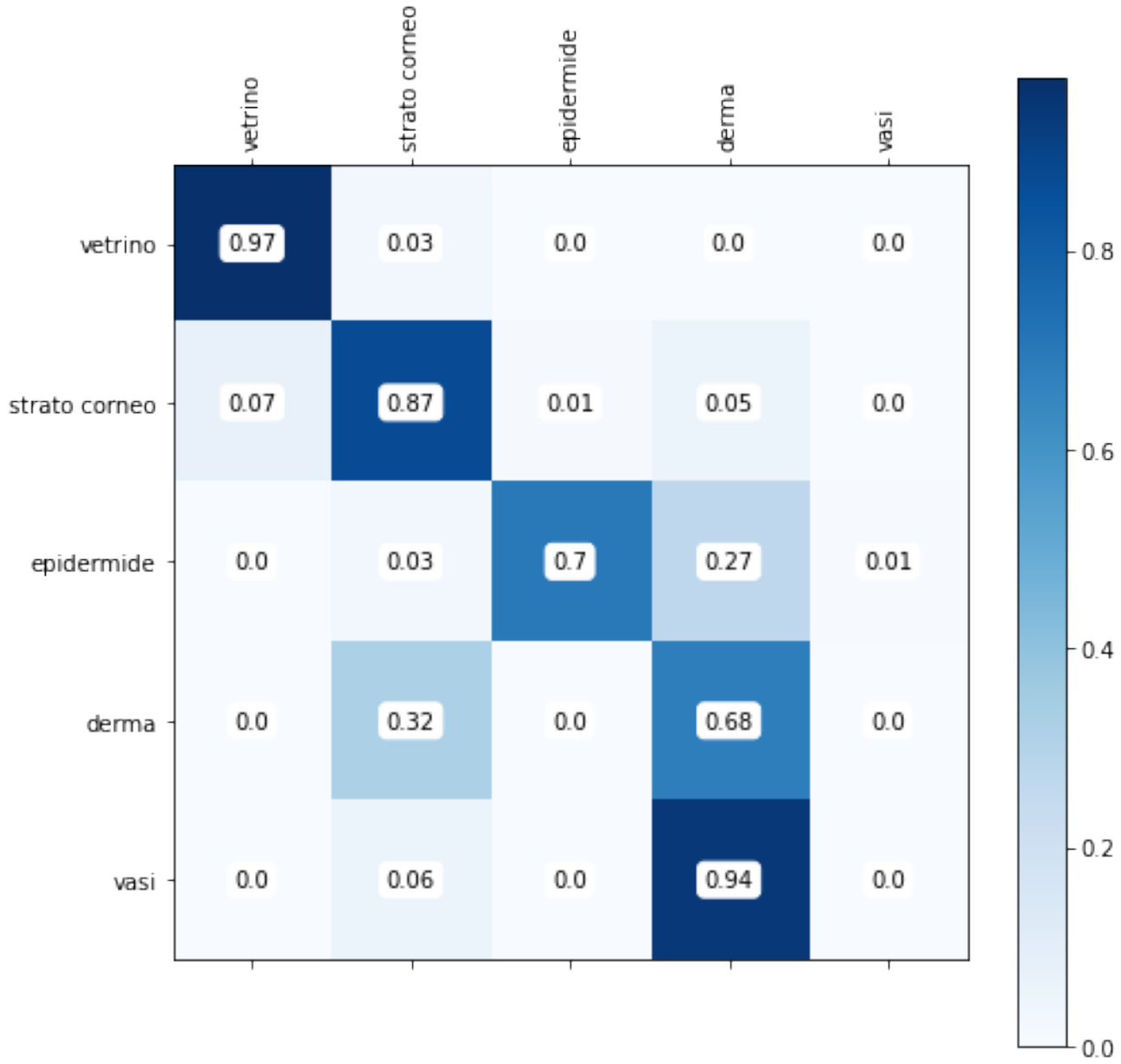


Figura A.22: Matrice di confusione dell'immagine 105 sull'algoritmo RF

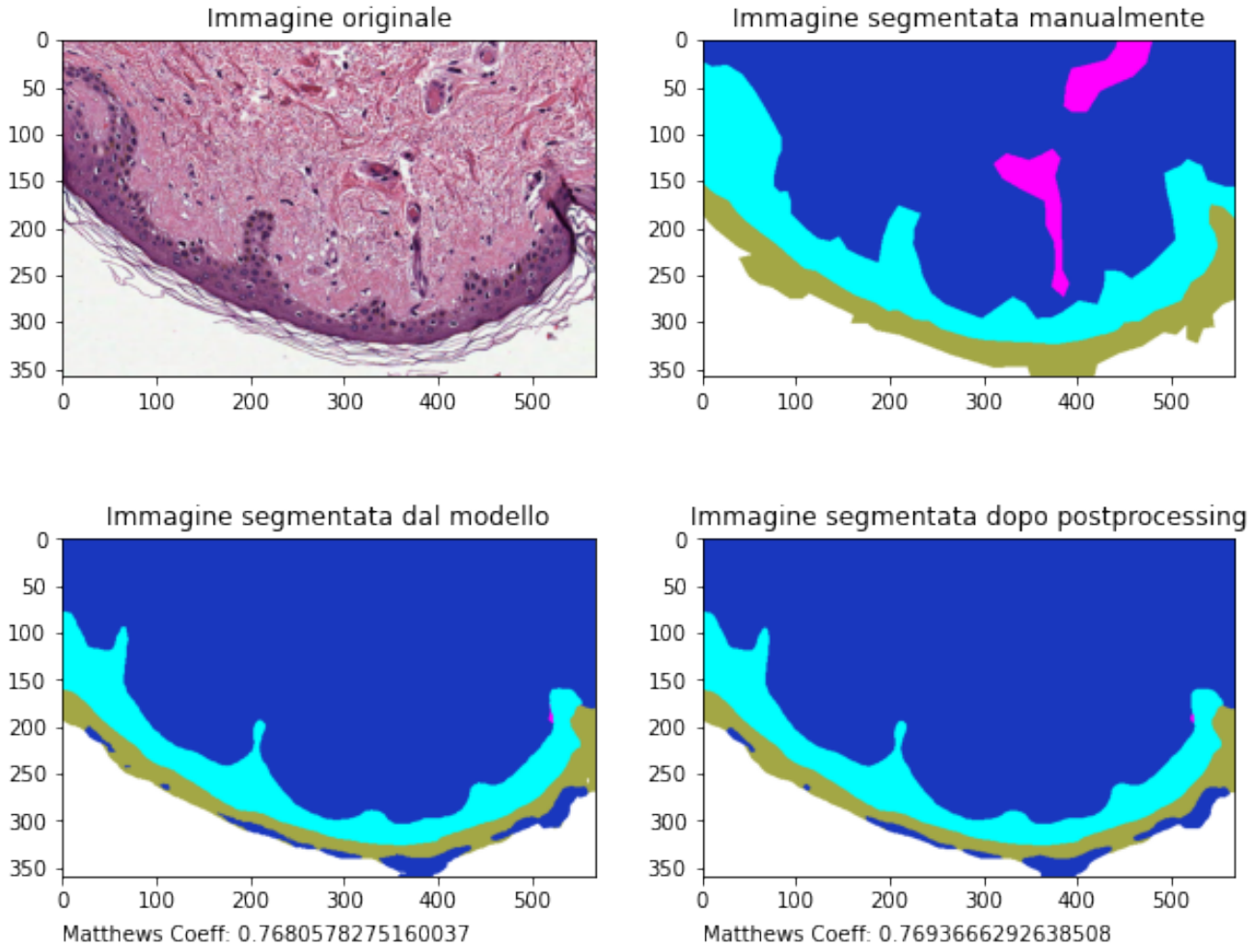


Figura A.23: Immagine 105 utilizzata per il test set sull'algoritmo SVM

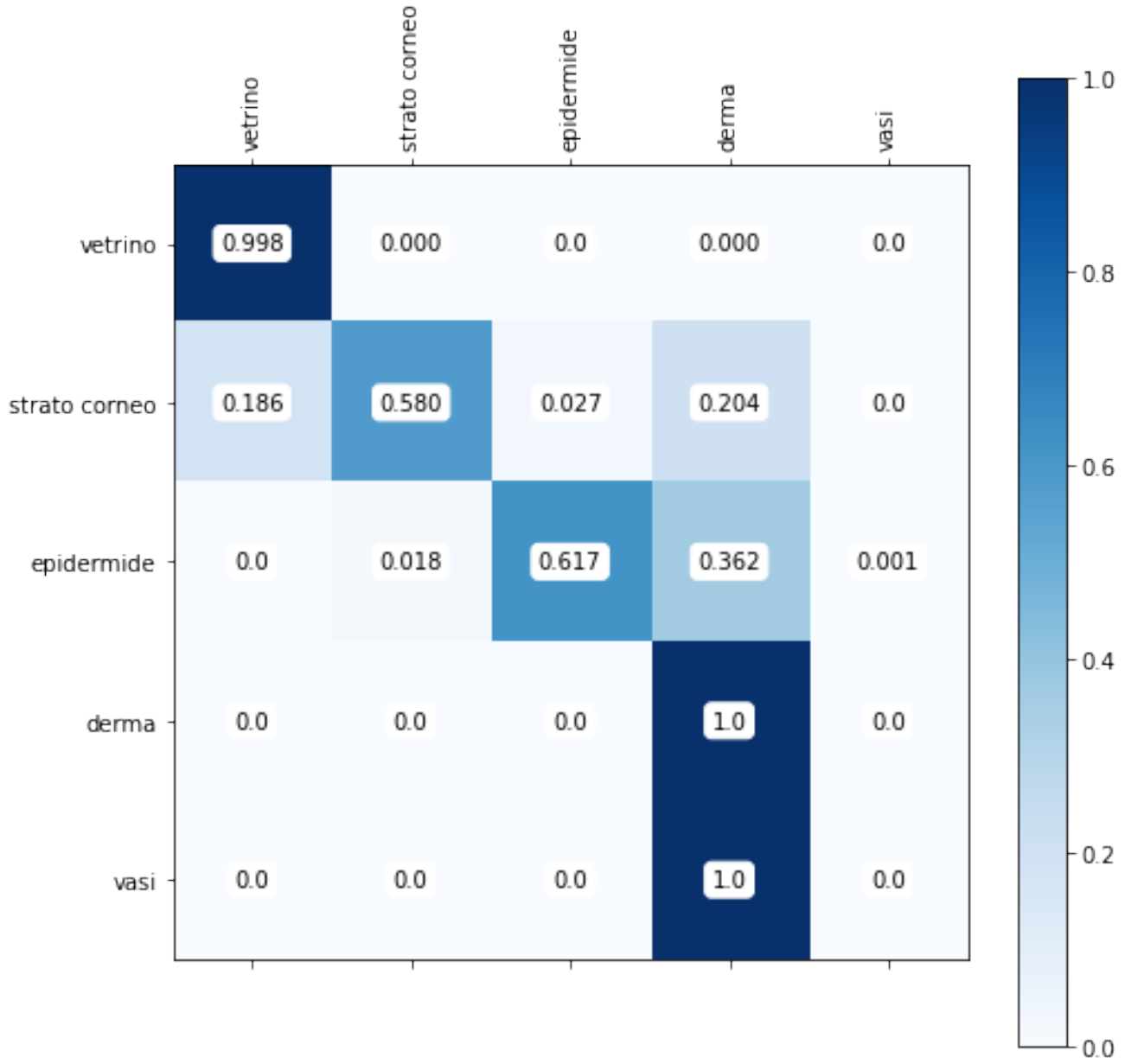


Figura A.24: Matrice di confusione dell'immagine 105 sull'algoritmo SVM

Tabella Grid Search allenati sul modello knn. La colonna evidenziata e quella dove abbiamo estrapolato il modello migliore per la segmentazione delle immagini.

score	window_size	features	neighbors
0.8121581491663432	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.8106640599940792	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.8089454882926896	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.8082705659179271	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.8078245168180355	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	20
0.8074503890806494	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	20
0.8068601185167039	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.8057333352830825	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	15
0.805331300029388	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.8052702892564052	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	15
0.8045235289746602	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	20
0.804005540746239	25	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	50
0.8027655244018161	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	10
0.802525097769183	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	15
0.8023577569172172	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	10
0.8015807548242O6	20	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	50
0.8014755797426947	25	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	25
0.8012131545477984	25	['R_mf', 'G_mf', 'B_mf', 'h_variance', 's_variance', 'v_variance', 'entropy']	50
0.8010646690930151	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	7
0.8007608300404241	25	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	20
0.8003482341092276	35	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.8001913314382711	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	7
0.7999925805207849	20	['R_mf', 'G_mf', 'B_mf', 'h_variance', 's_variance', 'v_variance', 'entropy']	50
0.7996274494279313	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	5
0.7994060332662062	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	10
0.7991418611488379	30	['R_mf', 'G_mf', 'B_mf', 'h_variance', 's_variance', 'v_variance', 'entropy']	50
0.7988823389879381	30	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	50
0.7986649549174792	25	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	15
0.798354523174525	20	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	25
0.798103090167788	25	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	5
0.7979944632832942	30	['R_mf', 'G_mf', 'B_mf', 'L', 'a', 'b', 'entropy']	50
0.797887805551681	20	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	20
0.797705246358203	30	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	3
0.7970344736879483	25	['R_mf', 'G_mf', 'B_mf', 'hsv_sin', 'hsv_cos', 'entropy']	50
0.7966739909908413	20	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	7
0.7966451848329217	35	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	25
0.796099737056131	30	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	25
0.7964314625455867	30	['h', 's', 'v', 'h_mf', 's_mf', 'v_mf', 'h_variance', 's_variance', 'v_variance', 'entropy']	50
0.7962882157240786	30	['R_mf', 'G_mf', 'B_mf', 'hsv_sin', 'hsv_cos', 'entropy']	50
0.7962004091556726	25	['R_mf', 'G_mf', 'B_mf', 'h_variance', 's_variance', 'v_variance', 'entropy']	25
0.7961956649367683	20	['R_mf', 'G_mf', 'B_mf', 'h_variance', 's_variance', 'v_variance', 'entropy']	25
0.7960719074926088	20	['R_mf', 'G_mf', 'B_mf', 'h_mf', 's_mf', 'v_mf', 'entropy']	50
0.7959754113200940	25	['R_mf', 'G_mf', 'B_mf', 'H_mf', 'E_mf', 'D_mf', 'entropy']	50
0.7957784610010726	25	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	10
0.7957726219688559	20	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	15
0.7957283005667402	30	['R_mf', 'G_mf', 'B_mf', 'L', 'a', 'b', 'entropy']	25
0.7956525084399608	35	['R_variance', 'G_variance', 'B_variance', 'H_mf', 'E_mf', 'D_mf', 'entropy']	20
0.7956004899498538	30	['R_mf', 'G_mf', 'B_mf', 'R_variance', 'G_variance', 'B_variance', 'entropy']	20
0.7952972312488873	20	['R_mf', 'G_mf', 'B_mf', 'hsv_sin', 'hsv_cos', 'entropy']	50

Bibliografia

[1] Farahani N, Parwani A, Pantanowitz L., *Whole slide imaging in pathology: advantages, limitations, and emerging perspectives.Pathology and Laboratory Medicine International.*, 2015;7:23-33.

[2] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince., *Current Methods in Medical Image Segmentation.* In: Annual Review of Biomedical Engineering 2.1 (2000). PMID: 11701515, pp. 315337. `https`//doi.org/10.1146/annurev.bioeng.2.1.315.

[3] D. Withey and Z. J. Koles., *A Review of Medical Image Segmentation: Meth- ods and Available Software*, 2008.

[4] S. Raschka, *Machine Learning con Python: Costruire algoritmi per generare conoscenza*, Apogeo Editore, 2017.

[5] G. Pellegrino, *L'intelligenza artificiale al servizio della sicurezza informatica. Un approccio dinamico.*, Youcanprint, 2021.

[6] C.-W. Hsu and C.-J. Lin., *A comparison of methods for multi-class support vector machines* , IEEE Transactions on Neural Networks, 13(2002), 415-425.

[7] Francesco Granata, Michele Saroli, Giovanni de Marinis, Rudy Gargano, *" Machine Learning Models for Spring Discharge Forecasting "*, Geofluids ,2018.

[8] Umesh P., *Image Processing in Python*, Department of Computational Biology and Bioinformatics, University of Kerala, 2012

[9] `https://www2.fch.vut.cz/lectures/imagesci/includes/harfa_screenshots_smoothmedian.inc.php`

[10] https://stackoverflow.com/questions/26870349/two-dimensional-array-median-filtering

[11] Gian Maria Gasparri – Michele Lanzetta, *Il filtro entropico per il rilevamento di difetti tramite analisi di immagini*, Dipartimento di Ingegneria Meccanica Nucleare e della Produzione, Università di Pisa, 2011

[12] R.C. Gonzalez,R.E. Woods *Digital Image Processing*, 2012

[13] AC Ruifrok e DA Johnston, *Quantification of histochemical staining by color deconvolution*, Analytical and quantitative cytology and histology / the International Academy of Cytology [and] American Society of Cytology, vol. 23, no. 4, pp. 291-9, Aug. 2001.

[14] Torresin, L., *Sviluppo ed applicazione di reti neurali per segmentazione semantica a supporto della navigazione di rover marziani*, 2019

[15] Annamaria D'Ambrosio, 2021 https://github.com/Annamaria598/segmentazione_knn

[16] Annamaria D'Ambrosio, 2021 https://github.com/Annamaria598/PatternRecognition

[17] Harris, C.R., Millman, K.J., van der Walt, S.J. et al., *Array programming with NumPy*, Nature 585, 357–362 (2020). DOI:0.1038/s41586-020-2649-2.

[18] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stefan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J.Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, Ilhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antonio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.*, Nature Methods, 17:261–272, 2020.

[19] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu e i collaboratori di scikit-image., *Scikit-image: elaborazione delle immagini in Python*, PeerJ 2: e453 (2014)

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research,12, 2825-2830 (2011).

[21] Chicco, D., Jurman, G., *I vantaggi del coefficiente di correlazione di Matthews (MCC) rispetto al punteggio F1 e alla precisione nella valutazione della classificazione binaria.*, BMC Genomics 21,6 (2020) `https://doi.org/10.1186/s12864-019-6413-7`

[22] John D. Hunter., *Matplotlib: A 2D Graphics Environment*,Computing in Science Engineering,9, 90-95 (2007),DOI: 10.1109 / MCSE.2007.55

[23] Van der Walt, S., Schonberger, Johannes L, Nunez-Iglesias, J., Boulogne, Franccois, Warner, JD, Yager, N.,Yu, T. , *Scikit-image: image processing in Python*, PeerJ,2, e453. (2014).

Ringraziamenti

Finalmente è giunto al termine questo percorso di studi.

Innanzi tutto vorrei ringraziare il Dott. Giampieri Enrico per la disponibilità in questo lavoro di tesi.

Ringrazio in particolar modo la mia famiglia che mi ha sempre sorretto e sostenuta in questo percorso, anche quando dopo anni di lavoro ho deciso di riprendere gli studi. Mi hanno sempre sorretto nelle miei decisioni giuste o sbagliate che siano. Senza di loro non potevo coronare tutto questo.

Un grazie anche ad Ale.

Grazie al mio ragazzo Carlo, che mi ha sempre sorretto e sopportato nei momenti quando andavo giù di matto. Ha sempre creduto in me e mi ha sempre incoraggiata a vedere positivo.

Un grazie va a Francesco che mi è stato di molto aiuto per il supporto.

E soprattutto va a tutti i miei amici, che anche solo con una chiamata mi sono stati di supporto sempre. E poi c'è lui, Federico, il mio coinquilino "rinco" :P ma anche il fratello che non ho mai avuto. Ricordo, nei miei momenti di crisi e di forti indecisioni, quanto ha fatto nell'accogliermi all'ascolto.

. . . che dire un grazie di cuore a tutti!