Analisi Dati 21/22: Progetto Finale

Giuseppe Bruno (579265)

I. Introduzione

Lo scopo dell'analisi è di sviluppare un metodo che, a partire dalle time series relative alle rilevazioni del satellite Sentinel-2 in una singola area geografica, preveda quale sia il tipo di suolo osservato. Nel corso della sperimentazione si è cercato di dare un certo peso all'interpretazione dei modelli ottenuti.

II. Dataset e preprocessing

A. Descrizione del dataset

Il dataset in esame contiene 199424 samples. Le 46 colonne rappresentano le intensità misurate nel corso dell'anno da Sentinel-2 per il pixel selezionato. La prima colonna invece indica a quale delle 20 classi appartiene il sample corrispondente. Come si nota dalla Figura 1 la distribuzione delle classi è fortemente sbilanciata, in particolare la classe 1 è la più rappresentata con 47696 istanze, contro le 412 della classe 13. Inoltre la 11 e la 16 non sono presenti.

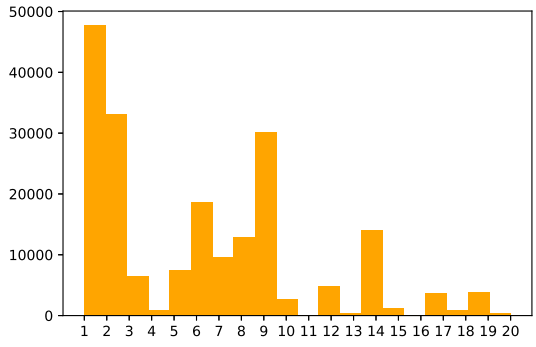

Figura 1. Distribuzione delle classi nel dataset di training.

B. Preprocessing

Inizialmente i dati sono stati importati e standardizzati. Per avere un'idea visiva del problema si è anche effettuato il plot dei primi 10000 samples per classe (Fig. 3) e della loro media (Fig. 4).

Si osserva immediatamente che le timeseries in generale tendono ad accumularsi attorno ad un pattern caratteristico della classe, l'intensità di tale fenomeno dipende però dalla classe stessa di appartenenza.

Il dataset inoltre è stato suddiviso in training set e test set tramite hold-out al 30% per permettere la cross-validation sui metodi analizzati. Lo splitting è stato effettuato in modo "stratificato"per mantenere lo stesso rapporto tra le classi durante le fasi di training, validation e test.

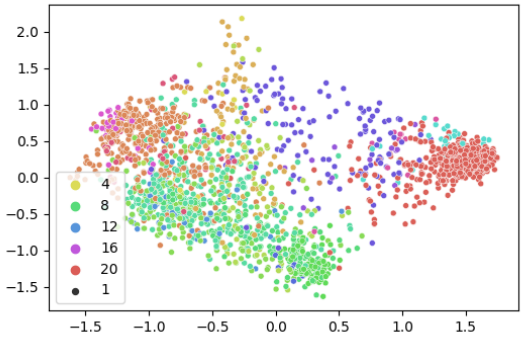

Figura 2. PCA sul piano principale dei primi 2000 samples.

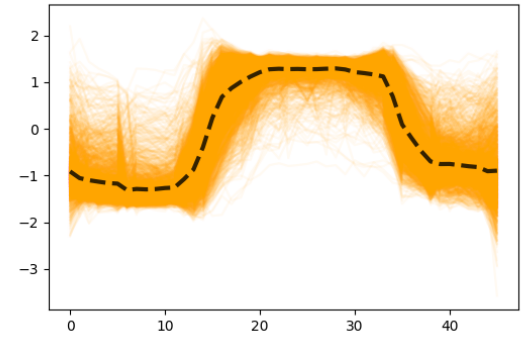

Figura 3. Plot delle prime 10000 istanze della classe 1 e della loro media (in nero).

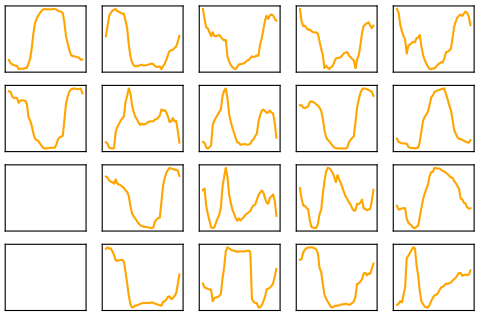

Figura 4. Medie delle 18 classi.

In un primo momento si è anche provato ad effettuare lo smoothing delle time series tramite *moving average* e tramite *Savitzky-Golay filter* [5]. Tuttavia le performance inferiori dei modelli allenati sui dati così trasformati hanno portato ad abbandonare tale idea e mantenere i dati nella versione originale.

C. Metriche utilizzate

La metrica di riferimento sfruttata per la valutazione dei modelli è stata l'*accuracy*, definita come rapporto tra numero di istanze correttamente classificate e cardinalità totale del test-set. Ad essa si è deciso di affiancare anche la cosiddetta *Kappa di Cohen*, il cui vantaggio consiste nel tenere conto dello sbilanciamento delle classi. Permette di confrontare la "observed accuracy" p_o e la "expected accuracy" p_e attraverso la formula:

$$k := \frac{p_o - p_e}{1 - p_e}.$$

Tuttavia occorre sottolineare come in letteratura non ci sia un accordo generale su quanto sia vantaggiosa tale metrica, pur essendo frequentemente utilizzata. In particolare viene criticata la bassa interpretabilità e la dipendenza dalla composizione del dataset [3]. Per queste ragioni e per quanto sarà discusso nella sezione V si è quindi deciso di dare priorità all'accuracy.

Infine nell'ultima pagina si trova un report dettagliato del modello scelto, comprendente metriche quali *Precision*, *Recall* ed *f1-Score*.

III. Modelli classici

Nella prima fase sono stati addestrati e validati modelli classici: kNN, SVM e Random Forests. Nella tabella in appendice sono riportati i risultati ottenuti per ogni tipo di modello. La scelta degli iperparametri è stata svolta mediante *GridSearch* con cross-validation (5-fold) su un sottoinsieme del dataset di 10000 samples.

Il *Support Vector Classifier* ha raggiunto uno dei risultati migliori nel corso dell'intera sperimentazione, nonostante l'elevato tempo richiesto sia nella fase di training che di predizione. Le *Random Forests* analizzate invece tendono ad overfittare, probabilmente una più attenta scelta dei parametri può ridurre il problema, ma non si è ritenuto interessante (una soluzione parziale alternativa è illustrata nella sezione V). Infine il kNN ha ottenuto un'accuracy superiore alle aspettative, pertanto si è deciso di analizzare una sua versione con distanza modificata nella sezione VI.

IV. Estrazione di ulteriori features

Per aumentare la quantità di informazioni presenti nel dataset e allo stesso tempo l'interpretabilità dei modelli, come discusso in [7], sono state estratte ulteriori features dalle istanze note. In particolare:

A. Descrizione delle nuove features

- *Features statistiche*: media, mediana, deviazione standard, massimo, minimo, punto medio, ampiezza, posizione del massimo e posizione del minimo;
- *Features fenologiche*: inizio (*sos*), fine (*eos*) e durata (*los*) della stagione di crescita, come riportate in Fig. 5. Sono state ottenute tramite l'utilizzo della funzione `peaks_widths` di scipy.signal sulle istanze duplicate ed affiancate (nel caso di stagioni che proseguono oltre l'anno in esame).
- *Features dalla serie di Fourier* [6]: sono stati calcolati tramite integrazione numerica i primi 7 coefficienti (c_n e φ_n) della serie di Fourier di ogni istanza:

$$c_0 + \sum_{n=1}^{7} c_n \cos\left(\frac{2\pi n x}{N} + \varphi_n\right).$$

Le features estratte corrispondono allora alle prime tre φ_n riscalate tra 0 e 1 e alle b_n definite come:

$$b_n := \frac{c_n^2}{\sum_{i=1}^{7} c_i^2}.$$

Come si può notare dalla Fig. 6 le b_n quantificano l'intensità del comportamento stagionale delle istanze, ed allo stesso tempo forniscono una versione dell'andamento temporale senza le frequenze più elevate (e probabilmente dovute a rumore).

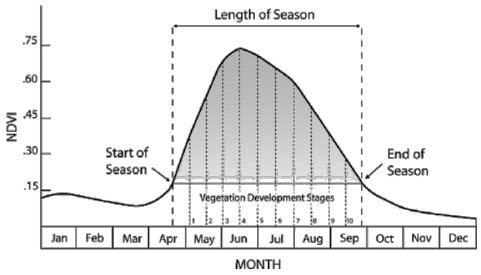

Figura 5. Features fenologiche tipiche per una coltivazione.

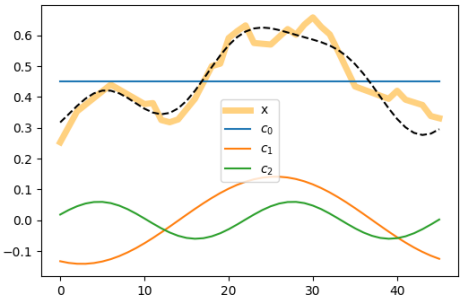

Figura 6. Primi tre termini della serie di Fourier. In nero la loro somma.

B. *Features importance*

A causa degli elevati tempi di calcolo necessari non è stato possibile procedere con una *subset selection* sulle 20 features appena descritte. Si è tuttavia fatto ricorso alla "feature importance" ottenuta dal training di una Random Forest tramite sklearn. I risultati sono riportati in Fig. 7. La fase del primo termine della serie di Fourier risulta di gran lunga più importante delle altre colonne, seguita poi dalle fasi successive e dalla deviazione standard.

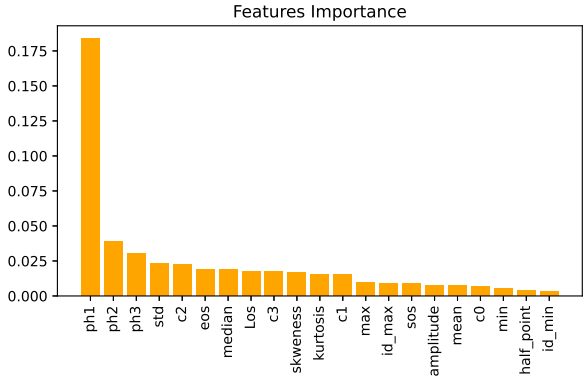

Figura 7. Features importance estratta da una RF in sklearn.

L'aggiunta di tali features al dataset originale non ha prodotto miglioramenti, come si può notare dalla tabella in appendice. É comunque interessante notare che sono sufficienti le quattro statistiche *ph1, ph2, mean, std* per ottenere un'accuracy del 70% tramite un semplice Decision Tree. Ciò potrebbe essere utile per classificare una zona più ampia con limitate risorse computazionali.

V. Il problema delle classi sotto-rappresentate

A. *Dataset enlargement*

In un primo momento si è deciso di non modificare la composizione del dataset, sia perché è la stessa di quella del dataset di test finale, sia perché si suppone che rispecchi la naturale distribuzione dei terreni nella zona analizzata. L'obiettivo è stato infatti interpretato come quello di classificare correttamente la più ampia superficie possibile.

Tuttavia, per affrontare il problema della bassa accuracy su classi poco rappresentate nel dataset, si è deciso di provare comunque tre differenti approcci:

- diversa loss function: tramite la modifica del parametro `class_weights` in "*balanced*"durante il training di reti neurali in keras, ovvero modificando la loss function assegnando un peso maggiore alle classi sotto-rappresentate;
- over-sampling: bootstrap per riequilibrare la distribuzione delle classi;
- SMOTE [2]: per ogni sample della classe in inferiorità numerica sono generati k nuovi samples considerando combinazioni convesse casuali tra il sample selezionato ed i suoi k-nearest neighbours nella stessa classe.

I primi due metodi non hanno portato a risultati statisticamente rilevanti, mentre il terzo ha prodotto un certo miglioramento sulla Random Forest (si pensa dovuto alla riduzione dell'overfitting del modello), come riportato nella Tab. I.

Modello	No-aug	SMOTE
RF	82.19 %	82.96 %

Tabella I
RISULTATI DELLA RF SUL DATASET ORIGINALE E SUL DATASET AUMENTATO CON SMOTE

VI. Modelli specifici

La libreria *sktime* offre un gran numero di modelli specifici per la classificazione di time series. Tali modelli però richiedono tempi di training elevati ed è stato possibile allenarli soltanto su un sottoinsieme del dataset molto ridotto (circa 5000 samples).

Si è anche testato il 1-Nearest Neighbour con distanza *dtw*, che aumenta l'accuracy del 1-NN classico sul dataset ridotto dal 71% al 74%. Nonostante l'elevato costo computazionale lo renda difficilmente applicabile al caso in esame, è comunque un'interessante opzione per eventuali problemi di clustering.

VII. Reti neurali

Oltre al classico MLP sono stati scelti tre tra i modelli per la classificazione di time series più performanti in letteratura. Il miglior risultato è stato ottenuto partendo da una *Fully-Convolutional Network* (FCN), il cui modello base proposto in [12] è stato modificato traendo spunto da altre architetture tipiche per la classificazione di time series. In particolare si sono utilizzati 3 *Convolutional layers* da 256, 512, 256 filtri rispettivamente di dimensione 24, 12 e 3 (con l'idea "euristica"di rappresentare mezzo anno, una stagione, un mese). Inoltre il *GlobalAveragePooling* è stato sostituito con un *MaxPooling* ed è stato aggiunto un layer *denso* finale composto da 1024 neuroni con output dato da 18 neuroni con attivazione *softmax*. La rete presenta più di 4 milioni di parametri e tende ad overfittare i dati. Per tentare di ridurre tale fenomeno si è allora aggiunto un layer *Dropout* al 50% dopo l'ultimo layer convoluzionale. L'architettura completa è riportata in Fig. 8.

Gli altri due modelli sono varianti di quest'ultimo: GRU-FCN [4] ed LSTM-FCN [9]. Infine è stato testato

anche il modello BiLSTM suggerito in [1]. A causa dell'elevato tempo necessario per il training purtroppo il tuning dei parametri di questi ultimi modelli è stato ridotto al minimo, concentrandosi sui valori consigliati nelle pubblicazioni corrispondenti (solitamente testati sui datasets UCR).

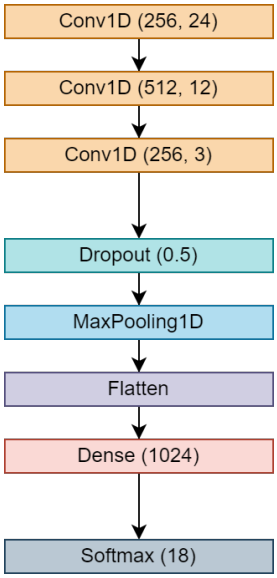

Figura 8. Architettura della FCN modificata.

VIII. Scelta del modello

Dalle analisi svolte i modelli migliori risultano il SVC (84.75%) e la FCN (85.43%), tuttavia la scelta tra i due si è rivelata problematica: nonostante il primo abbia ottenuto un'accuracy inferiore rispetto alla rete neurale, la natura dell'algoritmo consentirebbe un training finale sull'intero dataset. Al contrario il fitting della FCN ha richiesto l'implementazione di uno scheduling del learning rate[1] basato sulla validation accuracy, e di conseguenza utilizzare l'intero dataset soltanto per il training porterebbe ad un elevato rischio di overfitting (di cui la rete mostrava segni già durante la fase precedente di sperimentazione), senza possibilità di avere un riscontro sulla reale efficacia dell'aggiornamento dei pesi. A ciò si aggiungono le diverse fonti di aleatorietà (dall'inizializzazione casuale allo svolgimento di operazioni in parallelo tramite GPU [10]) che ostacolavano la riproducibilità del training della rete (e di conseguenza delle prestazioni) anche una volta fissati gli iper-parametri ed il *seed*.

Per procedere con cautela nella scelta tra i due algoritmi si era pensato allora di ridurre le dimensioni del test set fino al 10% del dataset totale, procedendo poi con la cross-validation tramite 10-fold nel seguente modo:

- SVC: 90% train set, 10% test set;
- FCN: 85% train set, 5% validation set, 10% test set;

per ogni fold. Un semplice hold-out al 10% infatti non si era giudicato statisticamente affidabile per il confronto dei due modelli. Tuttavia i limiti di Google Colab non hanno permesso di portare a termine lo studio nel secondo caso.

Vista la situazione si è quindi scelto di presentare come modello finale la più "robusta"tra le due soluzioni, ovvero il SVC.

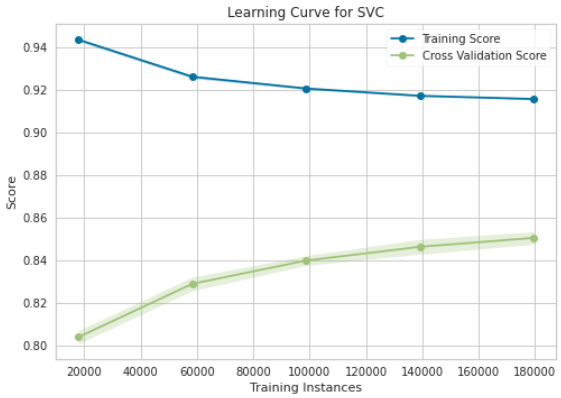

Figura 9. Learning-curve del SVC.

IX. Interpretazione del modello

Guardando alle prestazioni della FCN si è comunque cercato di capire quali siano gli aspetti delle time series utilizzati dalla rete durante la classificazione. In particolare si è sfruttato l'approccio *Gradient-weighted Class Activation Mapping* (Grad-CAM) [11] [8]: partendo dal modello di rete neurale FCN discusso precedentemente è possibile visualizzare le *class activation maps* relative all'ultimo layer convoluzionale.

In breve: sia y^c l'output del neurone corrispondente alla classe c prima dell'applicazione dell'attivazione *softmax*. Siano $A^1, ..., A^n$ le *feature activation maps* dell'ultimo layer convoluzionale (ovvero gli output degli n filtri, nel caso unidimensionale A^k sarà un vettore di taglia $(1, m)$). Allora tramite backpropagation è possibile calcolare il gradiente $\frac{\partial y^c}{\partial A^k_j}$ per il sample da analizzare. Si definisce quindi:

$$\alpha^c_k = \frac{1}{m} \sum_{j=1}^{m} \frac{\partial y^c}{\partial A^k_j}$$

ed il risultato è dato dal vettore:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_{k=1}^{n} \alpha^c_k A^k \right)$$

riscalato e sovrapposto al sample di riferimento.

[1] L'optimizer scelto è ADAM con *EarlyStopping* e *ReduceLROnPlateau* con fattore 0.1 rispetto alla metrica *val_accuracy*.

L'idea è quindi quella di linearizzare la parte finale della rete neurale per poi assegnare un maggior peso alle misurazioni del satellite le cui intensità contribuiscono ad aumentare y^c.

Dalla Fig. 10, in cui è riportata la heatmap restituita dall'algoritmo (applicato ad un sample della classe 1), si nota come la rete neurale concentri la sua attenzione sull'inizio della stagione di crescita, sulla lunghezza del plateau, ma anche sull'improvviso crollo finale. Questo confermerebbe che il modello sia riuscito ad apprendere le statistiche fenologiche discusse nella sezione IV autonomamente ed in modo migliore rispetto all'estrazione manuale descritta precedentemente.

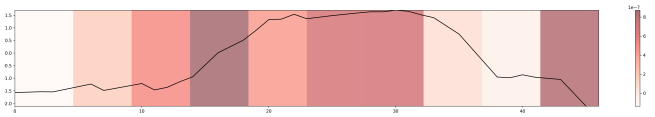

Figura 10. Grad-CAM applicato ad un elemento della classe 1.

X. Stacking

I due modelli discussi finora presentano prestazioni simili. Inoltre entrambi sono in grado di restituire la probabilità che un sample appartenga ad una data classe.

Si è quindi deciso di esplorare la semplice idea di combinare i due modelli in un solo metodo FCN+SVM che restituisca la media delle probabilità predette dai singoli algoritmi, in modo da diminuire la varianza e combinare i punti di forza di entrambi.

Nella Fig. 11 è riportata la learning-curve (fino a 40000 samples) dei due metodi e del modello combinato. I risultati sono promettenti, tuttavia i tempi di training necessari affinché il SVC restituisca anche le probabilità aumentano considerevolmente.

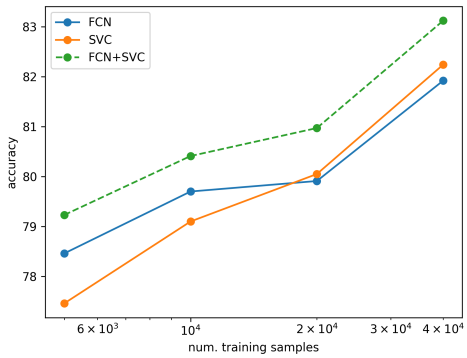

Figura 11. Learning curve fino a 40000 samples dei 3 modelli.

Anche sul solito hold-out al 30% il FCN+SVC supera entrambi i modelli, diventando il migliore tra quelli analizzati durante l'intera sperimentazione:

Modello	Accuracy
SVC	84.75 %
FCN	85.43 %
FCN+SVC	**86.06** %

Tabella II
Confronto tra i tre modelli

Le considerazioni sulla scelta del modello discusse nella sezione VIII diventano qui meno rilevanti, viste le prestazioni del modello ibrido sempre superiori a quelle di entrambi i modelli. Si è deciso pertanto di consegnare, oltre al SVC, il FCN+SVC ottenuto dalla rete neurale allenata sul 90% del dataset (con 10% di validation) ed il SVC allenato sull'intero dataset.

XI. Conclusioni

Un ulteriore sviluppo potrebbe riguardare la costruzione di un ensemble che raccolga i risultati di più modelli dipendenti separatamente dalle time series, dalle statistiche e dalle features derivate dalla serie di Fourier.

Si è anche osservato che il SVC è l'unico tra gli algoritmi presi in esame che sfrutta la modalità *OneVsRest* nella classificazione. Si potrebbe dunque pensare all'implementazione di una classificazione di tipo "gerarchico", distinguendo prima tra macro-classi e poi per ognuna di esse allenare un modello ristretto al sotto-problema corrispondente. Per far ciò occorrerebbe conoscere la reale corrispondenza tra labels e tipi di terreni (dato che i clusters ottenuti tramite *k-means* non si sono ritenuti soddisfacenti). A sostegno delle potenzialità di questo metodo si possono notare nella Fig. 12 gruppi di classi che vengono confuse tra di loro (ad esempio 3,4,5).

D'altra parte gli scarsi risultati ottenuti da modelli ristretti a questo gruppo fanno dubitare della capacità discriminatoria di alcuni dati.

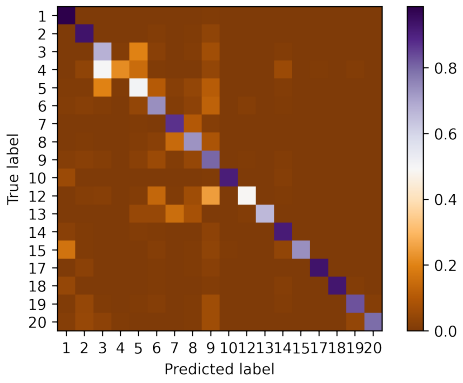

Figura 12. Confusion Matrix del SVC.

Per quanto riguarda i modelli proposti, guardando la learning curve in Fig. 9, si deduce che il SVC non sembra avere un gran margine di miglioramento e che i benefici che si otterrebbero da un aumento delle dimensioni del dataset sarebbero comunque limitati.

Al contrario tra i modelli analizzati le reti neurali convoluzionali appaiono come le più promettenti e sicuramente meriterebbero un'analisi più approfondita, in particolare nella scelta dell'architettura, dei parametri ma soprattutto nelle modalità di apprendimento.

XII. Link allo script

Link al notebook su Google Colab.

Riferimenti bibliografici

[1] M. Campos-Taberner, F. J. García-Haro, B. Martínez, E. Izquierdo-Verdiguier, C. Atzberger, G. Camps-Valls, and M. A. Gilabert. Understanding deep learning in land use classification based on sentinel-2 time series. *Scientific reports*, 10(1):1–12, 2020.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] R. Delgado and X.-A. Tibau. Why cohen's kappa should be avoided as performance measure in classification. *PloS one*, 14(9):e0222916, 2019.

[4] N. Elsayed, A. S. Maida, and M. Bayoumi. Deep gated recurrent and convolutional network hybrid model for univariate time series classification. *arXiv preprint arXiv:1812.07683*, 2018.

[5] J. N. Hird and G. J. McDermid. Noise reduction of ndvi time series: An empirical comparison of selected techniques. *Remote Sensing of Environment*, 113(1):248–258, 2009.

[6] M. E. Jakubauskas, D. R. Legates, J. H. Kastens, et al. Harmonic analysis of time-series avhrr ndvi data. *Photogrammetric engineering and remote sensing*, 67(4):461–470, 2001.

[7] K. Jia, S. Liang, X. Wei, Y. Yao, Y. Su, B. Jiang, and X. Wang. Land cover classification of landsat data with phenological features extracted from time series modis ndvi data. *Remote sensing*, 6(11):11518–11532, 2014.

[8] Kaggle. Clf. and exp. arrhythmia from rr-int. using cnn. https://www.kaggle.com/bjoernjostein/clf-and-exp-arrhythmia-from-rr-int-using-cnn.

[9] F. Karim, S. Majumdar, H. Darabi, and S. Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[10] Keras. How can i obtain reproducible results using keras during development? https://keras.io/getting_started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[12] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

Modello	Parametri	Dataset	Train Acc.	**Test Acc.**	k	Tempo
kNN	$k = 5$	Base	87.21%	82.05%	0.79	-
		Base+Stat	86.13%	80.73%	0.78	-
		Base+Four	86.65%	81.54%	0.79	-
		SMOTE	98.28%	79.86%	0.77	-
NB-Classifier	Gaussian	Base	51.22%	51.23%	0.45	-
		Base+Stat	51.11%	51.03%	0.45	-
		Base+Four	51.16%	51.14%	0.46	-
		SMOTE	56.05%	48.76%	0.43	-
SVM	$C = 3$, $\gamma = 0.2$, rbf	Base	91.7%	**84.75**%	0.82	1035 s
		Base+Stat	94.42%	84.11%	0.81	1759 s
		Base+Four	94.16%	84.35%	0.81	1449 s
		SMOTE	97.46%	84.26%	0.82	5810 s
RF	estim=200, depth=250	Base	100%	82.11%	0.79	165 s
		Base+Stat	100%	81.94%	0.79	205 s
		Base+Four	100%	82.31%	0.80	223 s
		SMOTE	98.32%	82.91%	0.81	1216 s
XGBoost	Default	Base	94.85%	82.22%	0.79	581 s
		Base+Stat	95.02%	81.45%	0.79	650 s
		Base+Four	95.65%	82.05%	0.79	713 s
		SMOTE	96.20%	81.15%	0.78	4245 s
MLP	Dense(256),(512),(256)	Base	88.13%	83.41%	0.81	30 s
FCN		Base	99.95%	**85.43**%	0.83	632 s
LSTM-FCN		Base	93.05%	83.93%	0.81	1836 s
GRU-FCN		Base	94.77%	83.69%	0.81	1612 s
BiLSTM		Base	91.41%	82.82%	0.80	3405 s
Arsenal	Default	Ridotto	86.29%	73.45%	0.69	790 s
kNN-DTW	k=1, distance=dtw	Ridotto	100%	74.11%	0.74	- s
TimeSeriesForest	Default	Ridotto	100%	74.12%	0.69	29 s
SVM	C=3, γ=0.2, rbf	Ridotto	96.83%	76.69%	0.73	2 s

Tabella III
RIASSUNTO DEI RISULTATI PER MODELLO OTTENUTI DURANTE LA SPERIMENTAZIONE.

Classe	Precision	Recall	F1-Score	Support
1	0.968754	0.985883	0.977244	14309
2	0.949944	0.935448	0.942640	9961
3	0.611906	0.677568	0.643065	1957
4	0.595745	0.216216	0.317280	259
5	0.534208	0.505095	0.519244	2257
6	0.769173	0.736805	0.752641	5608
7	0.763102	0.873137	0.814420	2885
8	0.711461	0.731601	0.721391	3886
9	0.746043	0.801126	0.772604	9061
10	0.955026	0.911616	0.932817	792
12	0.812217	0.488435	0.610025	1470
13	0.854167	0.661290	0.745455	124
14	0.927176	0.914170	0.920627	4206
15	0.918301	0.735602	0.816860	382
17	0.955679	0.934116	0.944774	1108
18	0.960000	0.926641	0.943026	259
19	0.954410	0.831606	0.888786	1158
20	0.878788	0.794521	0.834532	146
accuracy	0.847463	0.847463	0.847463	0.847463
macro avg	0.825895	0.758938	0.783191	59828
weighted avg	0.848588	0.847463	0.845995	59828

Tabella IV
REPORT DETTAGLIATO DEL SVC FINALE SCELTO.