Università degli Studi di Udine

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Corso di Laurea in Magistrale Comunicazione Multimediale e Tecnologie dell'Informazione

Machine Learning – A.A. 2024/25

Docenti: prof. Christian Micheloni, prof. Claudio Piciarelli

Diagnosi del Morbo di Parkinson dalla Voce tramite Random Forest e Conversione in TensorFlow Lite

Studente: Angela Rossi

Matricola: 174288

OBIETTIVO GENERALE DEL PROGETTO

Il progetto qui presentato mira a costruire uno strumento capace di identificare il morbo di Parkinson analizzando campioni vocali, il tutto affidato a tecniche di machine learning. Oltre a realizzare e testare il modello, si è voluta anche l'idea di tradurlo in un formato leggero, adatto a telefoni o circuiti embedded, sfruttando TensorFlow Lite.

PANORAMICA DEL DATASET

- **Dimensioni del dataset**: 195 campioni, 24 caratteristiche.
- **Distribuzione delle classi**:
    - Classe 1 (affetto da Parkinson): 147 campioni
    - Classe 0 (non affetto): 48 campioni

Il dataset è sbilanciato verso la classe positiva (malattia presente), e ciò è stato tenuto in considerazione nella valutazione del modello.

MODELLO DI MACHINE LEARNING

Per questo progetto è stato utilizzato un **Random Forest Classifier**, un algoritmo di apprendimento supervisionato basato su un insieme di alberi decisionali. L'idea principale è quella di creare molti alberi (una "foresta") e combinare le loro predizioni per ottenere un risultato più robusto, riducendo l'overfitting e migliorando la generalizzazione del modello.

Per ottimizzare le prestazioni del modello, è stato impiegato **GridSearchCV**, una tecnica che permette di cercare automaticamente la combinazione migliore di iperparametri tramite una validazione incrociata.

I **migliori iperparametri trovati** sono stati:

- **n_estimators = 100**
  Questo parametro definisce **quanti alberi** compongono la foresta. Un numero maggiore di alberi può migliorare la precisione del modello, ma anche aumentare il tempo di calcolo. In questo caso, sono stati utilizzati 100 alberi.
- **max_depth = None**
  Indica la **profondità massima di ciascun albero**. Impostando None, si

consente agli alberi di crescere **fino a quando ogni foglia contiene meno di min_samples_split campioni**. Questo può portare a modelli più complessi, ma se combinato con altri parametri (come min_samples_split) può mantenere una buona generalizzazione.

- **min_samples_split = 5**
  Questo parametro specifica il **numero minimo di campioni richiesto per dividere un nodo interno**. Impostando questo valore a 5, si evita che l'albero cresca troppo in profondità su pochi campioni, contribuendo a prevenire l'overfitting.

Perché Random Forest?

- È **robusto al rumore** e agli **outlier**.
- Funziona bene con **dataset piccoli o di medie dimensioni**, anche se sbilanciati.
- Non richiede molta **pre-elaborazione dei dati** (es. scaling).
- Fornisce **stima dell'importanza delle feature**, utile per analisi successive.

PRESTAZIONI DEL MODELLO

Matrice di Confusione

	Predetto 0	Predetto 1
Reale 0	8	2
Reale 1	1	28

Il modello ha commesso solo 3 errori su 39 campioni nel set di test, ottenendo un'accuratezza complessiva del 92%.

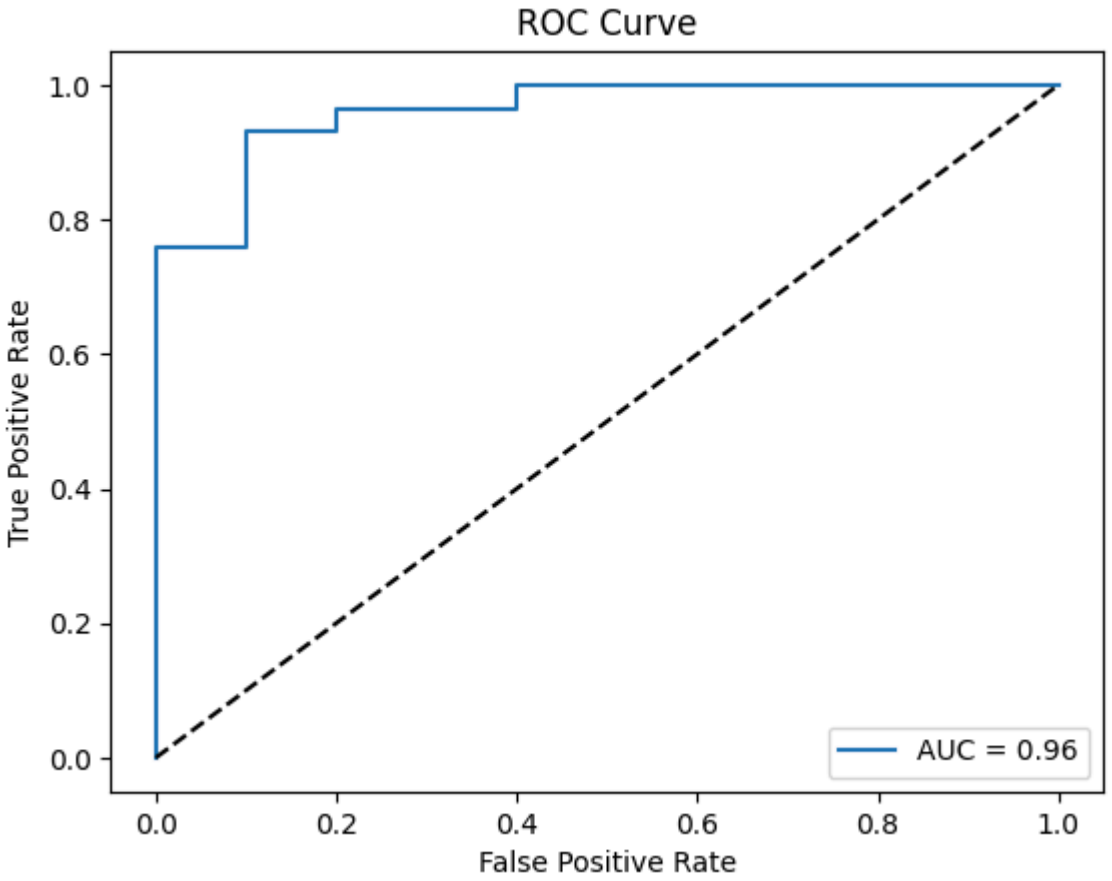
Classificazione

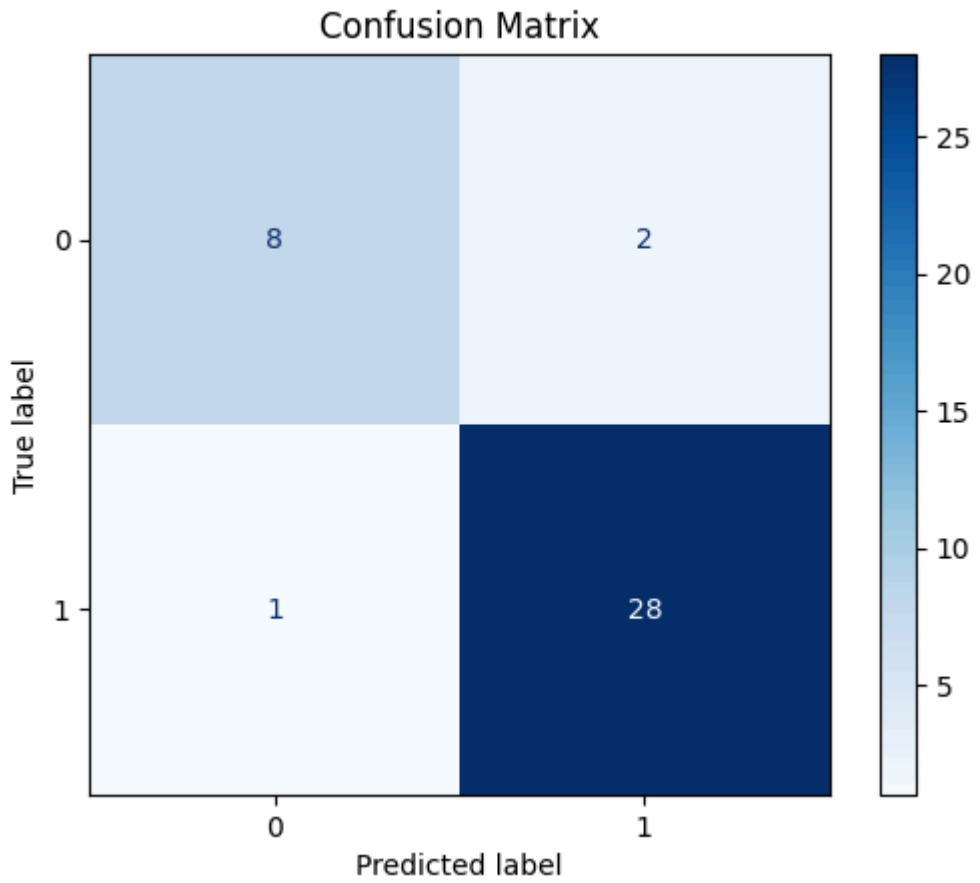
	Precision	Recall	F1-score	Support
Classe 0	0.89	0.8	0.84	10
Classe 1	0.93	0.97	0.95	29

Precisione media ponderata: 0.92

- AUC-ROC: 0.96, come mostrato nella curva ROC allegata.

Il modello si è dimostrato particolarmente efficace nel riconoscere correttamente i pazienti affetti (classe 1), con un recall del 97%.





CONVERSIONE IN TENSORFLOW LITE

Dopo il training, il modello è stato convertito con successo in formato **TensorFlow Lite (TFLite)** per l'esecuzione su dispositivi a bassa potenza.

Specifiche del modello convertito:

- **Input shape**: (None, 22) — Le 22 caratteristiche selezionate del dataset.
- **Output**: Probabilità per ciascuna delle due classi. I modello restituisce **due valori**, che rappresentano la **probabilità** che un campione appartenga a ciascuna delle due classi.
- **Esempio di inferenza TFLite**:
    - Output shape: (1, 2)
    - Probabilità predetta: [0.0118, 0.9881] → Predizione: **Classe 1 (Parkinson)**

La conversione si è completata con successo e il modello è pronto per essere integrato in applicazioni mobili o embedded per il supporto diagnostico.

Le 22 caratteristiche:

MDVP:Fo(Hz) – Frequenza fondamentale media della voce

MDVP:Fhi(Hz) – Frequenza massima

MDVP:Flo(Hz) – Frequenza minima

MDVP:Jitter(%) – Variazione percentuale della frequenza

MDVP:Jitter(Abs) – Variazione assoluta della frequenza

MDVP:RAP – Jitter relativo medio

MDVP:PPQ – Period perturbation quotient

Jitter:DDP – Derivato del jitter

MDVP:Shimmer – Variazione dell'ampiezza

MDVP:Shimmer(dB) – Shimmer in decibel

Shimmer:APQ3

Shimmer:APQ5

MDVP:APQ

Shimmer:DDA

NHR – Rapporto rumore-armonico

HNR – Rapporto armonico-rumore

RPDE – Entropia del segnale vocale

DFA – Analisi della fluttuazione detrended

spread1 – Misura non lineare della voce

spread2

D2 – Dimensione correlata della dinamica vocale

PPE – Entropia di predizione per la voce

La forma (None, 22) significa:

- None → numero variabile di campioni (es. 1, 10, 100, ecc.)
- 22 → ciascun campione ha 22 valori numerici (una per ogni feature)

LAVORI FUTURI

- **Bilanciamento del dataset con SMOTE**
  SMOTE sta per Synthetic Minority Over-sampling Technique. È una tecnica molto usata in machine learning per bilanciare dataset sbilanciati.
  Analizza i campioni della classe minoritaria (in questo caso, i "sani"), per ogni punto, calcola i vicini più simili (es. k=5), genera nuovi punti sintetici tra i campioni reali e i loro vicini, quindi non copia i dati, crea nuovi esempi simili ma non identici.

```python
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

# Supponendo che X sia il dataframe delle feature e y il target
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Suddivisione in train/test set
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.2, random_state=42, stratify=y_resampled
)
```

- **Modello neurale con MLPClassifier**
  MLPClassifier è un modello di rete neurale artificiale fornito da scikit-learn, dove MLP sta per Multi-Layer Perceptron. È un tipo di rete neurale feed-forward, cioè i dati passano in una sola direzione: dall'ingresso all'uscita, senza cicli.

```python
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report

mlp = MLPClassifier(hidden_layer_sizes=(64, 32), activation='relu',
                    max_iter=500, random_state=42)
mlp.fit(X_train, y_train)

# Valutazione
y_pred = mlp.predict(X_test)
print(classification_report(y_test, y_pred))
```

- **Importanza delle feature (Random Forest)**
  Quando si usa un modello come Random Forest, il modello prende decisioni basandosi su certe colonne (feature) del dataset.
  Feature importance ti dice quali colonne hanno avuto il maggiore impatto nel determinare il risultato. In parole semplici: "Quali caratteristiche vocali sono state più utili per prevedere se una persona ha il Parkinson?"
  Random Forest è composto da tanti alberi decisionali. Ogni albero sceglie le feature da usare per dividere i dati in nodi.
- Per ogni feature, il modello misura quanto migliora la purezza dei nodi (cioè quanto rende più "chiare" le classificazioni) ogni volta che quella feature viene usata.

Cosa si intende per "purezza"?

- Un nodo è "puro" se contiene solo esempi della stessa classe. Quindi, una feature importante è quella che riesce meglio a separare i dati in modo netto.
- Il punteggio finale è una media pesata dell'importanza di quella feature su tutti gli alberi della foresta.

```python
import matplotlib.pyplot as plt
import numpy as np

importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X.columns

plt.figure(figsize=(10, 6))
plt.title("Importanza delle Feature")
plt.bar(range(X.shape[1]), importances[indices], align='center')
plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)
plt.tight_layout()
plt.show()
```

CONCLUSIONI

Nel progetto ho creato un sistema di apprendimento automatico che classifica la malattia di Parkinson analizzando la voce delle persone. Per farlo abbiamo scelto un Random Forest Classifier, messo a punto con la grid search (GridSearchCV) così da tirare fuori il massimo dalle caratteristiche audio.

I numeri parlano chiaro:
- Accuratezza complessiva: 92%
- Recall per la classe positiva (Parkinson): 97%
- AUC-ROC: 0.96, che indica un eccellente equilibrio fra sensibilità e specificità.

Il modello riesce dunque a individuare quasi sempre i pazienti affetti da Parkinson, suggerendo che possa davvero aiutare i medici nella diagnosi. Convertendolo in TensorFlow Lite, siamo riusciti a renderlo leggero e veloce, pronto per smartphone o schede embedded e quindi per scenari clinici sul campo.