Diagnosing Parkinson's Disease from Voice Using Random Forest and Conversion in TensorFlow Lite

By Angela Rossi

OrcID: https://orcid.org/0009-0001-8485-8666

GENERAL OBJECTIVE OF THE PROJECT
The project presented here aims to build a tool capable of identifying Parkinson's disease by analyzing voice samples, all using machine learning techniques. In addition to building and testing the model, I also wanted to translate it into a lightweight format, suitable for phones or embedded circuits, using TensorFlow Lite.

DATASET OVERVIEW

- **Dataset size**: 195 samples, 24 features.
- **Class distribution**:
    - Class 1 (Parkinson's disease): 147 samples
    - Class 0 (unaffected): 48 samples

The dataset is skewed towards the positive class (disease present), and this was taken into account in the evaluation of the model.

MACHINE LEARNING MODEL

For this project a**Random Forest Classifier**, a supervised learning algorithm based on a set of decision trees. The main idea is to create many trees (a "forest") and combine their predictions to obtain a more robust result, reducing overfitting and improving the model's generalization.

To optimize the performance of the model, it was used**GridSearchCV**, a technique that allows you to automatically search for the best combination of hyperparameters through cross-validation.

I **best hyperparameters found they** were:

- **n_estimators = 100**
  This parameter defines how **many trees make** up the forest. A greater number of trees can improve the model's accuracy, but also increase computation time. In this case, 100 trees were used.

- **max_depth = None**
  Indicates the maximum **depth of each tree**. SettingNone, trees are allowed to grow until **each leaf contains less thanmin_samples_splitchampions**This can lead to more complex models, but when combined with other parameters

(such asmin_samples_split) can maintain good generalization.

- **min_samples_split = 5**
  This parameter specifies the **minimum number of samples required to split an internal node**Setting this value to 5 prevents the tree from growing too deep over a few samples, helping to prevent overfitting.

Why Random Forest?

- AND **robust to noise** and to the **outlier**.
- Works well with **small or medium-sized datasets**, even if unbalanced.
- It doesn't require much **data preprocessing** (es. scaling).
- It provides **feature importance estimation**, useful for subsequent analysis.

MODEL PERFORMANCE

Confusion Matrix

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

The model made only 3 errors out of 39 samples in the test set, achieving an overall accuracy of 92%.

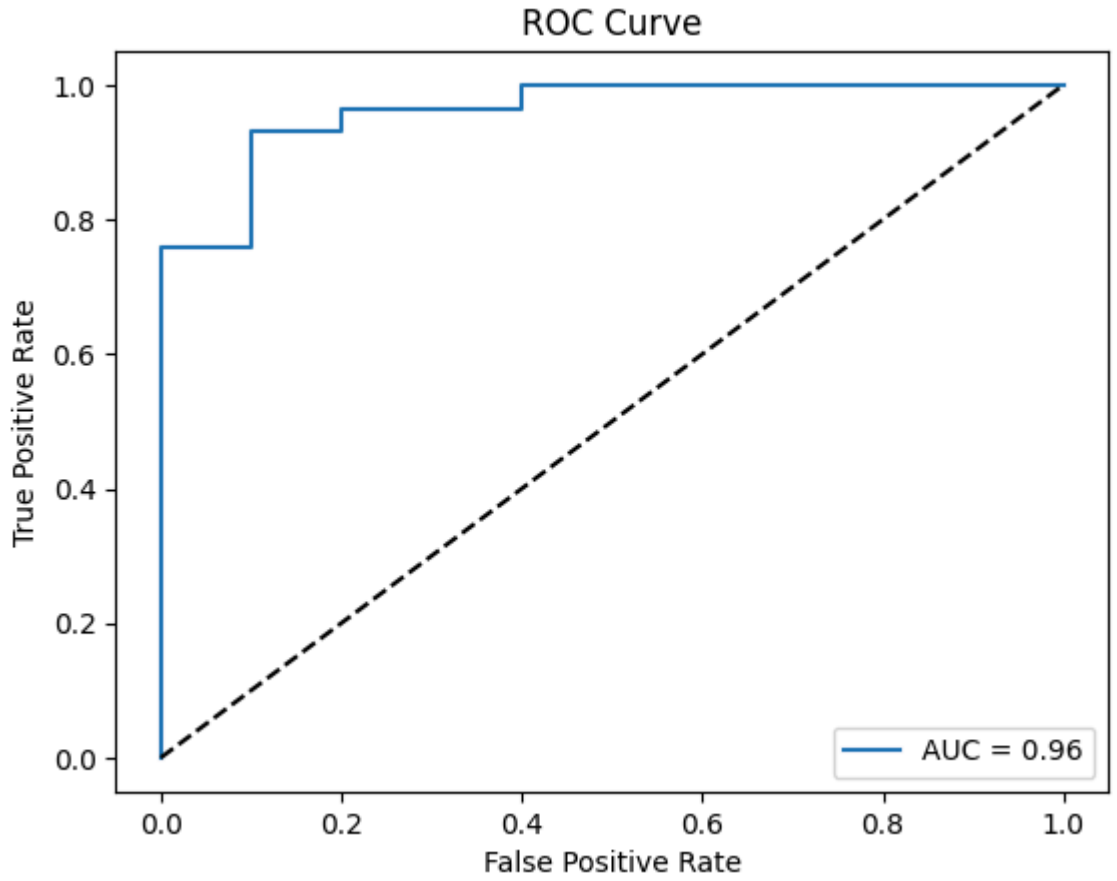
Classification

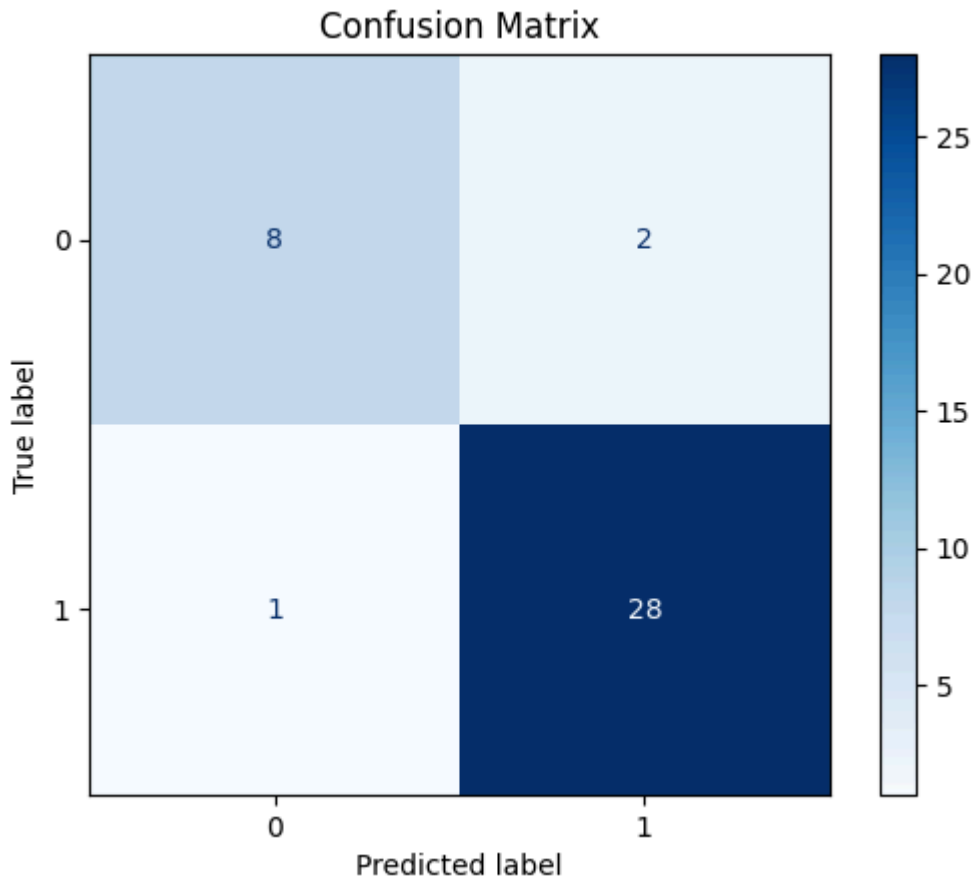
	Precision	Recall	F1-score	Support
Classe 0	0.89	0.8	0.84	10
Classe 1	0.93	0.97	0.95	29

Weighted average accuracy: 0.92

- AUC-ROC: 0.96, as shown in the attached ROC curve.

The model proved to be particularly effective in correctly recognizing affected patients (class 1), with a recall of 97%.





CONVERSIONE IN TENSORFLOW LITE

After training, the model was successfully converted to**TensorFlow Lite (TFLite)**for running on low-power devices.

Converted model specifications:

- **Input shape**: (None, 22)— The 22 selected features of the dataset.
- **Output**: Probability for each of the two classes. The model returns two **values**, which represent the chance that a sample belongs to each of the two classes.
- **TFLite Inference Example**:
    - Output shape: (1, 2)
    - Predicted probability:[0.0118, 0.9881]→ Prediction:**Classe 1 (Parkinson)**

The conversion has been successfully completed and the model is ready to be integrated into mobile or embedded applications for diagnostic support.

The 22 characteristics:

MDVP:Fo(Hz)– Average fundamental frequency of the voice

MDVP:Fhi(Hz)– Maximum frequency

MDVP:Flo(Hz)– Minimum frequency

MDVP:Jitter(%)– Percentage change in frequency

MDVP:Jitter(Abs)– Absolute frequency variation

MDVP:RAP– Average relative jitter

MDVP:PPQ – Period perturbation quotient

Jitter:DDP– Jitter derivative

MDVP:Shimmer– Amplitude variation

MDVP:Shimmer(dB) – Shimmer in decibel

Shimmer:APQ3

Shimmer:APQ5

MDVP:APQ

Shimmer:DDA

NHR– Noise-harmonic ratio

HNR– Harmonic-to-noise ratio

RPDE– Entropy of the voice signal

DFA– Detrended Fluctuation Analysis

spread1– Nonlinear voice measurement

spread2

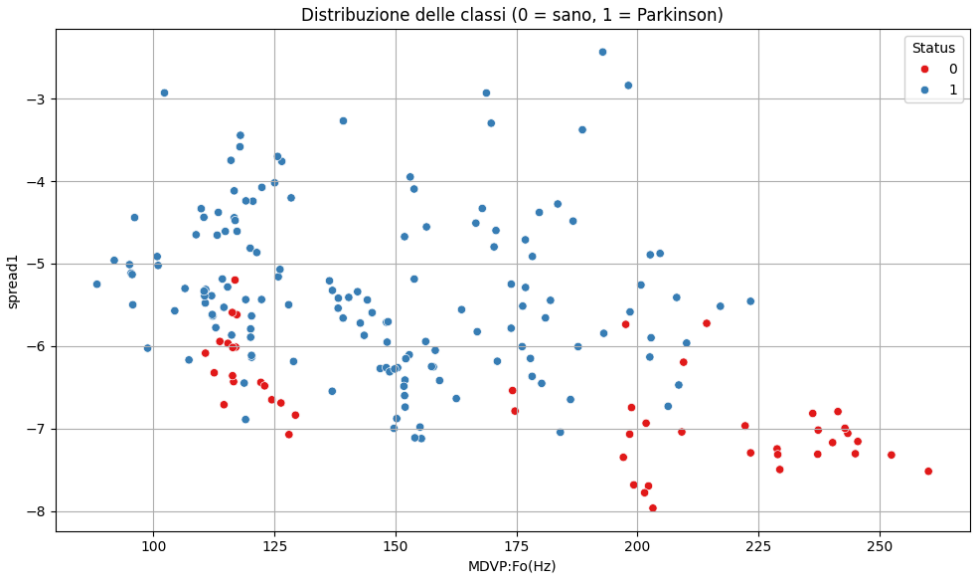
D2– Related dimension of vocal dynamics

PPE– Prediction entropy for voice

The form(None, 22)means:

- None→ variable number of samples (e.g. 1, 10, 100, etc.)
- 22→ each sample has 22 numeric values (one for each feature)

Scatter Plot



FUTURE WORKS

- **Balancing the dataset with SMOTE**
  SMOTE stands for Synthetic Minority Over-sampling Technique. It's a widely used technique in machine learning to balance unbalanced datasets.
  It analyzes the samples of the minority class (in this case, the "healthy" ones), for each point, calculates the most similar neighbors (e.g. k=5), generates new synthetic points between the real samples and their neighbors, therefore it does not copy the data, it creates new similar but not identical examples.

```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
)
```

- **Feature Importance (Random Forest)**
  When using a model like Random Forest, the model makes decisions based on certain columns (features) of the dataset.
  Feature importance tells you which columns had the greatest impact on the outcome. Simply put: "Which vocal characteristics were most useful in predicting whether a person has Parkinson's?"
  A Random Forest is composed of many decision trees. Each tree chooses the features to use to divide the data into nodes.
- For each feature, the model measures how much it improves the purity of the nodes (i.e., how much “clearer” it makes the classifications) each time that feature is used.

What is meant by “purity”?

- A node is "pure" if it contains only examples of the same class. Therefore, an important feature is the one that best separates the data.
- The final score is a weighted average of the importance of that feature across all the trees in the forest.

```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()
```

CONCLUSIONS

In this project, I created a machine learning system that classifies Parkinson's disease by analyzing people's voices. To do this, we chose a Random Forest Classifier, tuned with grid search (GridSearchCV) to maximize audio characteristics.

The numbers speak for themselves:
- Overall accuracy: 92%
- Recall for the positive class (Parkinson's): 97%
- AUC-ROC: 0.96, indicating an excellent balance between sensitivity and specificity.

The model is therefore able to almost always identify Parkinson's patients, suggesting it can truly help doctors with their diagnosis. By converting it to TensorFlow Lite, we were able to make it lightweight and fast, ready for smartphones or embedded boards and therefore for clinical scenarios in the field.

Link to the Google Colab:
https://colab.research.google.com/drive/1qGACQUX93sNMQaCItZWpgJcER7USNs5y?usp=sharing