Chase Howard
2022-03-13
CSS 584 Multimedia Database Systems

Assignment 4: Machine Learning Classification of Audio Files

How to run?

1) Unzip the app folder to the desired location – verify package contents
    a. Folders: audio, data, models
    b. Python file: AudioSegment, controller, FeatureExtraction, model, ModelEvaluation, view
2) Open a terminal and navigate to the location of the unzipped package
3) Type the following command:
    a. python.exe ./controller.py
4) Wait for the GUI to display

How to operate?

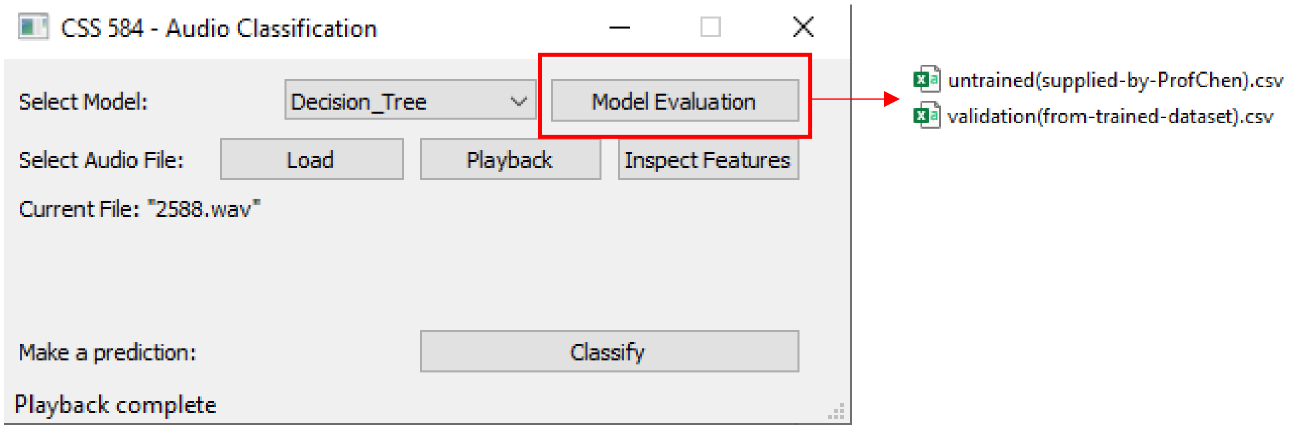
Caution: if loading from the validation set (see description below), there will be a significant delay in responsiveness of the GUI until the entirety of the file is loaded (often 7-12 min audio files). When the "Playback" button is selected, it will require the user to listen to the entirety of the audio file prior to processing the next command.

It is suggested for the user to test the features on the 'untrained' dataset, or the data provided for the purpose of the assignment to avoid excessing wait times.

Selecting and Evaluating a Model:
At the top of the GUI, the user has the option to load a set of 5 pretrained models by selecting the drop-down menu on the Combo Box. All models were trained on the same set of data; however, this was different than the supplied audio.

The user also can select "Model Evaluation" which will then prompt the user to select 1 of 2 data sets (or import their own dataset) to evaluate each models performance.

- *Validation.csv* is a subset of the same dataset used to train the models, this will most accurately represent the performance of the model
- *Untrained.csv* includes the features extracted from the dataset provided for this assignment; however, none of the models were trained on any data retrieved from the audio files provided.



Current File: "2588.wav"

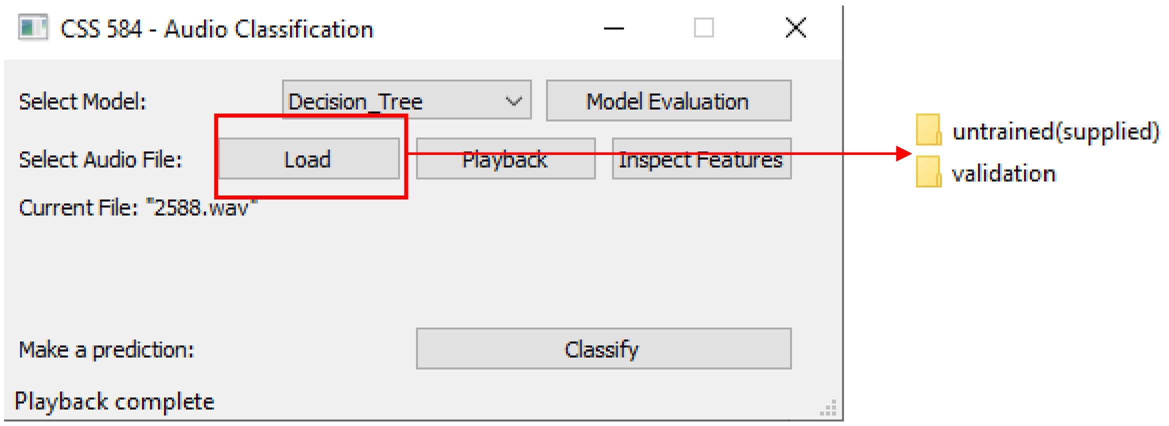
Playback complete

Loading, Playing, and Inspecting an Audio File:

A user can load any audio file from the file explorer. Note that once a file is loaded, the program will subsequently pull features from the audio file. This may take awhile on larger files and may require some patience. The GUI will default to a directory allowing the user to select audio from either the validation dataset or the supplied audio from Professor Chen.

Additionally, the user can listen to or 'Playback' the audio file by selecting the playback button. Note this is a blocking call and will not allow any user interaction with the GUI until the completion of the audio file. This is especially important for long audio files.

Finally, the user can inspect the audio features that have been extracted as well as display the amplitude over time and the magnitude of the frequencies by selecting the 'Inspect Features' button. This will open a second window to display the items of interest.

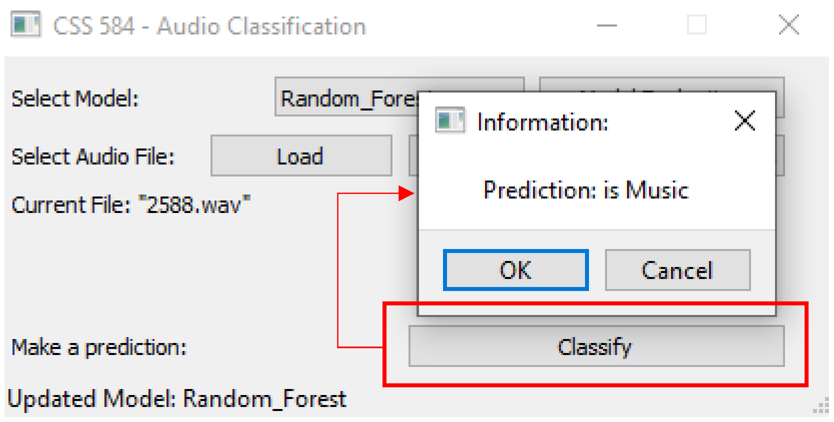


Current File: "2588.wav"

Playback complete

Classifying an audio file:

Once an audio file has been loaded, the user may choose to make a prediction using the model of interest by selecting the 'Classify' button on the bottom of the GUI. This will display a secondary window informing the user of the model's prediction.



Current File: "2588.wav"

Prediction: is Music

Updated Model: Random_Forest

Functionalities of program files

AudioSegment.py

This file contains the AudioSegement class which holds information relating to an entire audio file or segment of file. Components include

- Path to file
- Duration of file
- Sample Rate
- Amplitude Data
- Time Data
- Frequency Data

This file is used to load audio file and extract and store digitized information describing the audio. It also allows for sub-setting of audio files for more fine grained features as opposed to averaging features across an entire file.

FeatureExtraction.py

This file is self-explanatory – it is used to extract the various features from an Audio segment. It contains functions to load an entire directory of audio file and extract the relevant features. Available features to extract include:

- 'Bandwidth'
- 'ZeroCrossingRate'
- 'SilenceRatio'
- 'RMS'
- 'Energy'
- 'EnergyDistribution'
- 'SpectralCentroid'
- 'SpectralBandwidth'
- 'SpectralContrast'
- 'SpectralFlatness'

ModelEvaluation.py

This file is also self-explanatory – it is used to evaluate various Machine Learning models against a set of extracted features. This allows for the loading of feature data from file or importing of feature data in the form of a pandas DataFrame. Currently, there is the ability to evaluate up to 5 different models as follows:

- Logistic Regression
- Niave Bayes
- Decision Tree
- Random Forest
- XGBoost

There are also utilities for evaluating metrics of the model such as generating a confusion matrix, predicting against a feature set, as well as evaluate a cross_validation score of varying metrics. Models can either be loaded from a pickled file, or trained from existing feature sets.

Files for GUI operation

The remaining files are all used for use of the GUI to interface with audio files and the machine learning models which are predicting on them. They generally follow the Model-View-Controller format.

controller.py

This is the main driver for the program and the user will start the GUI from running this script. This file is used to handle all user requests, parse responses, format messages appropriately, as well as boot and shutdown the GUI. This serves as the query to the 'model' or backend and routes messages to the user interface or 'view'

model.py

This file is used to store all the data being accessed by the 'view' or display to the user. It is queried by the controller and returns relevant data.

view.py

This file contains all the code required to display the relevant information to the user. This was built on PyQt5 and will generate commands to be sent to and handled by the controller.

Libraries, Tools, and Techniques

Below is a list of useful libraries I found during this project and a brief description regarding their particular use in this instance

- Librosa – loading and extracting features from audio files
- Sklearn – vast, powerful machine learning database for python used for all machine learning models and much of the model evaluation
- PyQt5 / matplotlib.plotly – GUI development and display
- Numpy / pandas – general data handling and large scale computations
- Pickle – storing data in binary format and reloading

Some general techniquest that were used for developing the application included the use of the architectural pattern Model-View-Controller as well as implementing one of the Gang-of-Four design pattern referred to as the Command Structure for passing different commands to and from different classes. This allowed for increased modularization and flexibility when developing.

Additionally, I split the audio files into subsets to preserve some of the features and avoid having them washed out when averaged across an entire audio file.

Datasets and Example Features:

Note: Full extracted feature set can be found in the application directory under "./data/trained.csv". Additionally, a list of available features can be found in the above section under the title "FeatureExtraction.py"

Speech Dataset: https://www.kaggle.com/vjcalling/speaker-recognition-audio-dataset
Music Dataset: https://www.kaggle.com/imsparsh/musicnet-dataset

Many of the features were extracted using the 'librosa' library; however, there were a few that were written as custom functions to extract features.

Training vs Testing Data
There was a total of 50 hours of speech data and 50 hours of music data was taken for the entirety of the training. There were two methods by which the testing and training data was split.

For training of the models, data was split into 66% training data and 33% test data. The models were trained on the 66% and then exported in the form of a pickled file for use in later applications.

For reporting of metrics, 80 of the 100 hours were used with a 10-fold cross validation. This means the models were trained 10 different times where different subsets of data were withheld from training the model during each iteration. The output metrics were averaged across each of the different folds.

The final 20 hours not used in the above step were excluded for a final 'Validation Set', of which a smaller subset was submitted with this report for testing int eh GUI. A total of 40 files was submitted from the validation set.

Ground Truth Comparisons

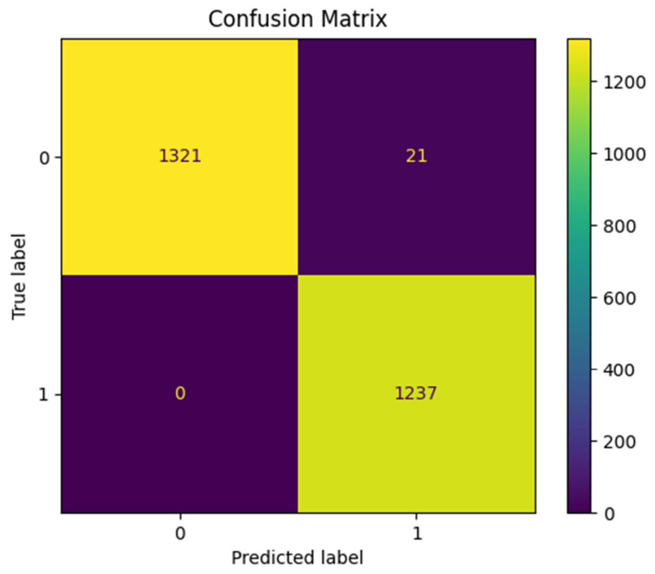
Note: Ground Truth comparisons are provided for both a subset of the dataset which was withheld during training (Validation Set) as well as a foreign set of data which the models were never exposed to (untrained Set -> 40 files) which was provided by the professor. Both are included in the following pages.

Metric Collection: All reported metrics in the table below were gathered by a 10-fold cross-validation score. The Confusion matrix results are reported from a single prediction from a pre-trained model when compared to the ground truth.

Summary:
Each model performed extremely well when confronted with data like the data it was trained on; however, when presented with data that it wasn't exposed to in training, it performed significantly worse. This hints towards a lack of generalizability either from the features selected or lack of diversity in training data.

Logistic Regression

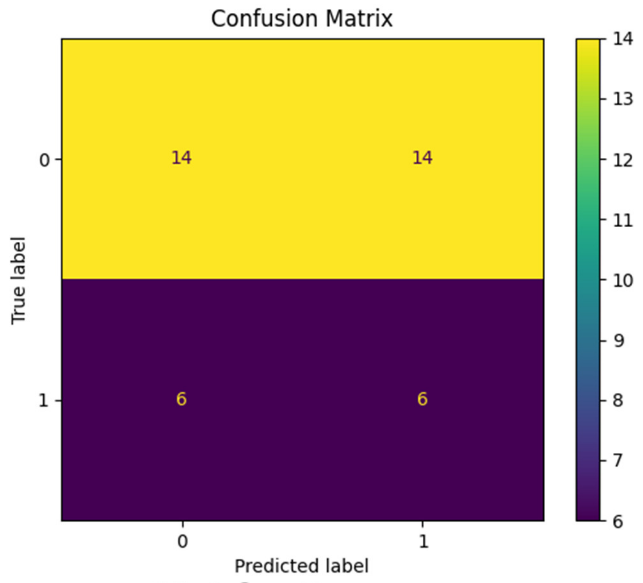
Validation Set:



Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.9895	1.0	0.9783	0.989	0.9996

Untrained Set:

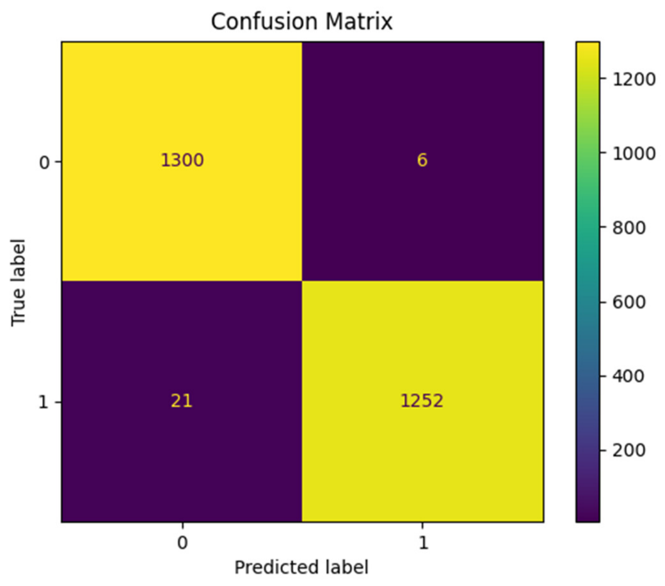


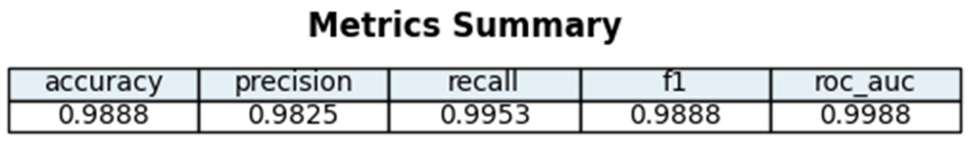
Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.7	0.65	0.7833	0.6633	0.8583

Naive Bayes

Validation Set:

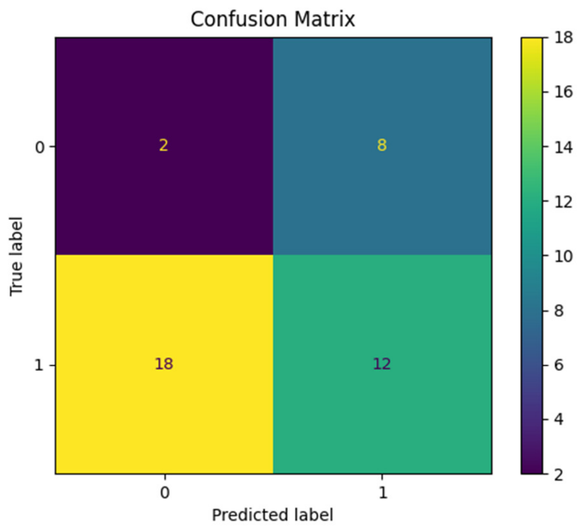


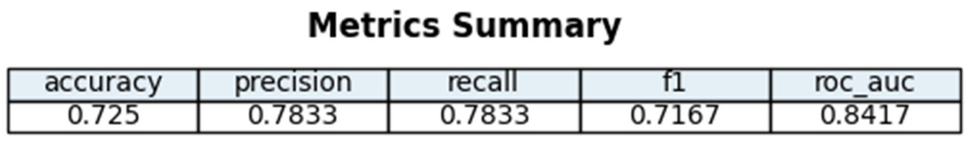


Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.9888	0.9825	0.9953	0.9888	0.9988

Untrained Set:



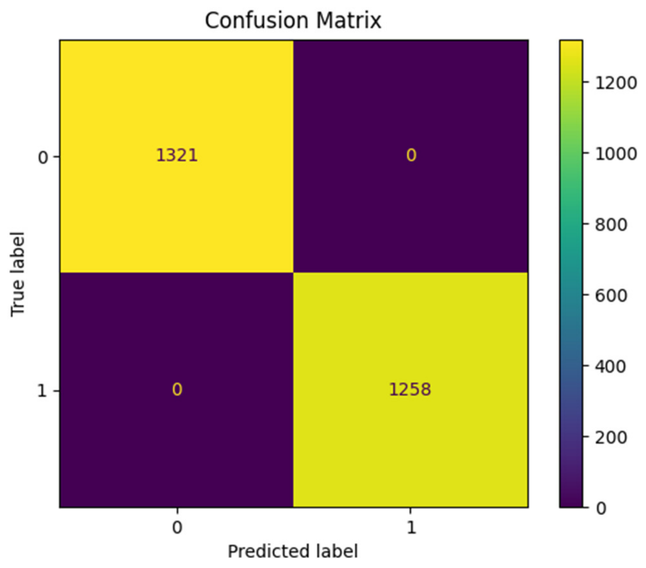


Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.725	0.7833	0.7833	0.7167	0.8417

Decision Tree

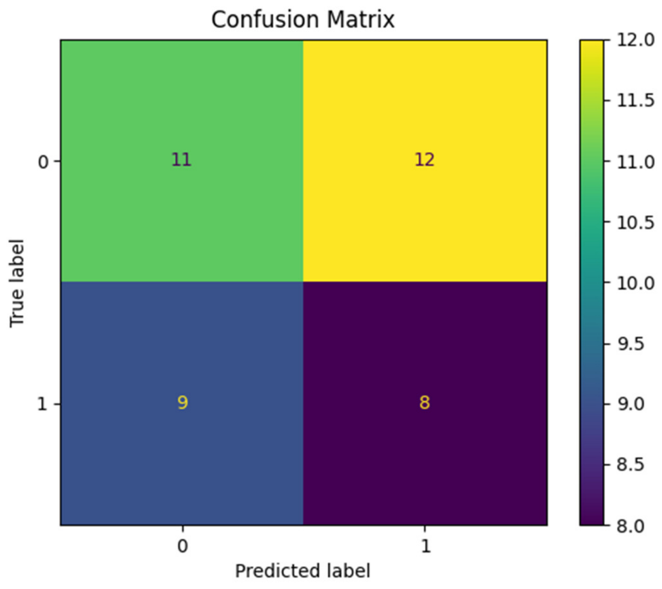
Validation Set:



Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.9996	0.9993	1.0	0.9996	0.9996

Untrained Set:

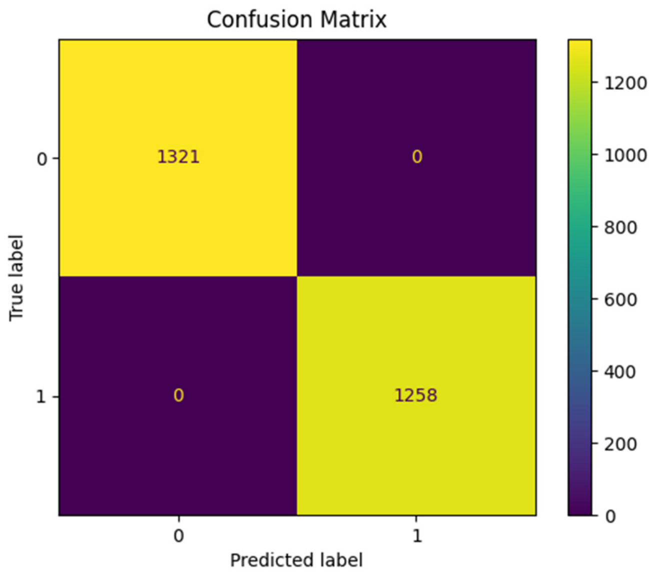


Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.725	0.6667	0.7333	0.6467	0.675

Random Forest (ensemble)

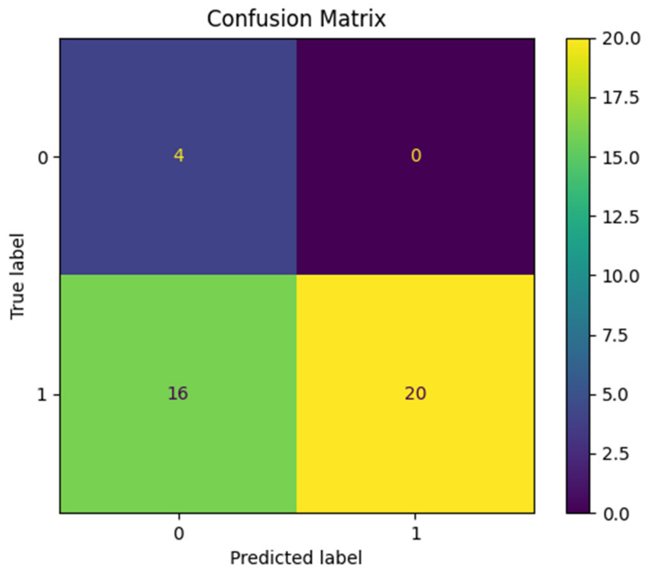
Validation Set:



Metrics Summary

accuracy	precision	recall	f1	roc_auc
1.0	1.0	1.0	1.0	1.0

Untrained Set:

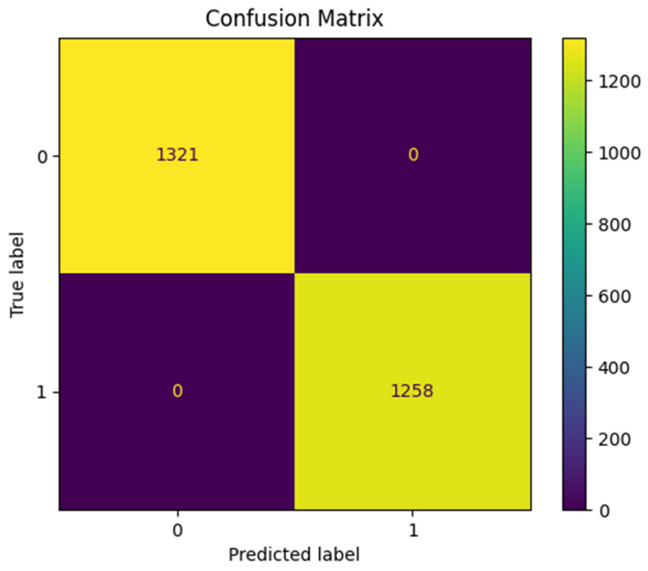


Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.725	0.6667	0.7833	0.5433	0.95

XGBoost (ensemble)

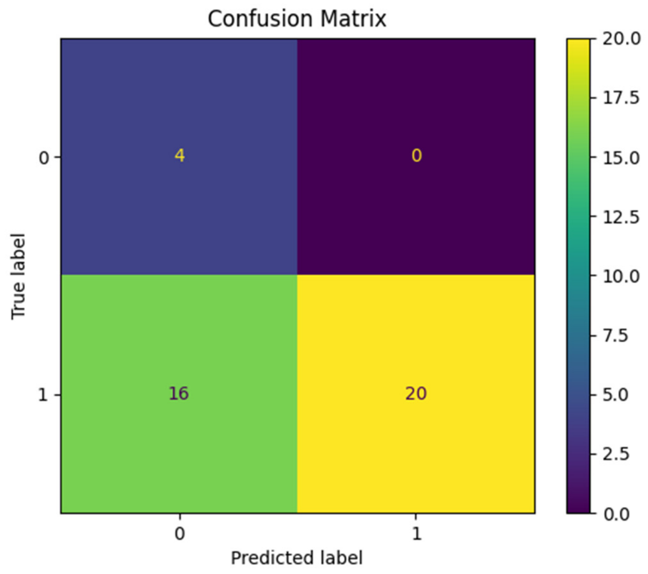
Validation Set:



Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.9996	0.9993	1.0	0.9996	0.9999

Untrained Set:



Metrics Summary

accuracy	precision	recall	f1	roc_auc
0.7	0.7	0.7	0.6533	0.8667