**Chapter 3 : Materials and Methods**

Machine Learning

Machine learning, a branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. Machine Learning is a scientific discipline that addresses the following question: ‘How can we program systems to automatically learn and to improve with experience?’ Learning in this context is not learning by heart but recognizing complex patterns and make intelligent decisions based on data. The difficulty lies in the fact that the set of all possible decisions given all possible inputs is too complex to describe. To tackle this problem, the field of Machine Learning develops algorithms that discover knowledge from specific data and experience, based on sound statistical and computational principles.

The field of Machine Learning integrates many distinct approaches such as probability theory, logic, combinatorial optimization, search, statistics, reinforcement learning and control theory. The developed methods are at the basis of many applications, ranging from vision to language processing, forecasting, pattern recognition, games, data mining, expert systems and robotics.

A learner can take advantage of examples (data) to capture characteristics of interest of their unknown underlying probability distribution. Data can be seen as examples that illustrate relations between observed variables. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new case. We have followed the same principle, where we have designed algorithms to generalize from the given examples and then produce a useful output. Since the amount of data available for the different languages was limited, we used KFold Cross Validation which makes efficient use of data available and avoids overfitting, which occurs when a model begins to "memorize" training data rather than "learning" to generalize from trend. Machine learning algorithms are described as either 'supervised' or 'unsupervised'.

Supervised Learning

Supervised learning is the machine learning task of inferring a function from supervised (labeled) training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. It is called a classifier (if the output is discrete) or a regression function (if the output is continuous). Supervised learning is when the data you feed your algorithm is "tagged" to help your logic make decisions.

Our classifying approach is a supervised learning method that uses labelled dataset of audio speech samples, using the features extracted from the frames as the basis of the classifier. Using some of the audio speech samples as the training dataset, classifier is applied on testing dataset to predict the spoken language. The testing method used is KFold Cross Validation which makes efficient use of the available data and avoids overfitting.

pyAudioAnalysis

pyAudioAnalysis is an open Python library that provides a wide range of audio-related functionalities focusing on feature extraction, classification, segmentation and visualization issues. The purpose of the pyAudioAnalysis library is to provide a wide range of audio analysis functionalities through an easy-to-use and comprehensive programming design.

pyAudioAnalysis implements the following functionalities:

* Feature extraction: several audio features both from the time and frequency domain are implemented in the library.
* Classification: supervised knowledge (i.e. annotated recordings) is used to train classifiers. A cross-validation procedure is also implemented in order to estimate the optimal classifier parameter. The output of this functionality is a classifier model which can be stored in a file.
* Regression: models that map audio features to real-valued variables can also be trained in a supervised context. Again, cross validation is implemented to estimate the best parameters of the regression models.
* Segmentation: the following supervised or unsupervised segmentation tasks are implemented in the library: fix-sized segmentation and classification, silence removal, speaker diarization and audio thumbnailing.
* Visualization: given a collection of audio recordings pyAudioAnalysis can be used to extract visualizations of content relationships between these recordings.

One of the best things of using pyAudioAnalysis was the ease of its use. It was very easy to extract the time and frequency domain features from an audio signal. However, since the time domain and the frequency domain features apart from the mfcc features did not provide any improvement in the results, the library was used only to extract the mfcc features. Inspite of providing very accurate classifiers it did not contain any DNN Classifiers, which was the reason it wasn’t used for classifying spoken audio samples. A drawback in this library is the large processing time it takes to extract all the features from an audio. It extracts all the features regardless of how many maybe required. It is advised to tweak the code for the library to extract only the required features. It was used both in the test and training phases to extract features from the audio speech samples.

LibROSA

LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It is built to ease the transition of music information retrieval (MIR) researchers into Python (and modern software development practices), and also to make core MIR techniques readily available to the broader community of scientists and Python programmers. It has a relatively flat package layout, and following scipy rely upon numpy data types and functions, rather than abstract class hierarchies. Librosa’s functions expose all relevant parameters to the caller. While this provides a great deal of flexibility and a consistent interface to process audio files by defining a set of general conventions and standardized default parameter values shared across many functions. LibROSA supports the following basic features among others:

* Compute mel spectrogram, MFCC, delta features, chroma features
* Locate beat events
* Compute beat-synchronous features
* Display features
* Save beat tracker output

We used LibROSA to compute the delta and delta delta coefficients from the obtained mfcc coefficients. The best thing about using librosa was the ease of its use. Both delta and delta delta functions could be calculated in a straightforward manner. Althouh librosa does provide functionality to extract mfcc coefficients from an audio signal, it lacks the versatility that pyAudioAnalysis provides and hence wasn’t used for feature extraction.

Deep Learning

**Deep learning** (also known as **deep structured learning**, **hierarchical learning** or **deep machine learning**) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations.Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. It is one of the fastest growing fields of Machine Learning - it’s true bleeding edge.

For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures which remove redundancy in representation. Deep learning is nothing but a neural network with a lot of hidden layers of nonlinear processing and supervised or unsupervised learning of feature representations in each layer, such that the layers form a hierarchy from low level to high level features. Deep learning is called ‘deep’ learning, as its algorithms transform their inputs through more layers than shallow learning algorithms. At each layer, the signal is transformed by a processing unit, like an artificial neuron, whose parameters are learned through training.

We have used DNNs because they have the ability to learn complicated feature representations and classifiers jointly. They learn much better models of data that lie on or near a non-linear manifold and their performance does not saturate with increase in training data.

Theano

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

* Tight integration with NumPy.
* Transparent use of a GPU
* Efficient symbolic differentiation
* Speed and stability optimizations dynamic C code generation
* Extensive unit-testing and self-verification

Theano defines a language to represent mathematical expressions and manipulate them, a compiler to create functions that can compute values for these expressions, and a library which will execute these functions when evaluated on numeric values.

The Keras software used in our project builds on the strengths of Theano, by providing a higher level user interface. Keras makes it easier to express the architecture of deep learning models, and training algorithms, as mathematical expressions to be evaluated by Theano.

Keras

**Keras** is an open source neural network library written in Python. It is capable of running on top of Deeplearning4j, Tensorflow or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being minimal, modular and extensible. *Being able to go from idea to result with the least possible delay is key to doing good research.* It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer.

Features of Keras:

* Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
* Supports both convolutional networks and recurrent networks, as well as combinations of the two.
* Runs seamlessly on CPU and GPU.

We used Keras to design and implement the DNNs at both the initial stage as well as the binary classification stage. The main advantage of using Keras was how easy it was to ignore the difficult mathematical details of the underlying neural network and focus only on optimizing the performance of the net. Keras provided a smooth interface to change every parameter of the neural network including the number of layers, type of connections, activation functions, error function, number of neurons in each layer, weight initialization, dropout, regularization and performance metrics among others. It even has an option of loading and saving a model which enables fast learning. Keras is strongly advised for anyone who wishes to analyse the performance of DNN on a problem in a short amount of time.

scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance.

scikit-learn provides the following functionalities:

* Classification: Identifying to which category an object belongs to.

Algorithms: SVM, nearest neighbors, random forest

* Regression: Predicting a continuous-valued attribute associated with an object.

Algorithms: SVR, ridge regression, Lasso

* Clustering: Automatic grouping of similar objects into sets.

Algorithms: k-Means, spectral clustering, mean-shift

* Dimensionality reduction: Reducing the number of random variables to consider.

Algorithms: PCA, feature selection, non-negative matrix factorization

* Model selection: Comparing, validating and choosing parameters and models.

Modules: grid search, cross validation, metrics

* Preprocessing: Feature extraction and normalization.

Modules: preprocessing, feature extraction.

We used scikit-learn to build the SGD Classifier which was used as a baseline for comparison with our proposed Model. The library was used to perform KFold Cross Validation which makes efficient use of the limited data available and avoids overfitting. It was also used to preprocess the inputs before feeding it to the neural network to convert the output labels into one hot vectors and to shuffle the input before training.