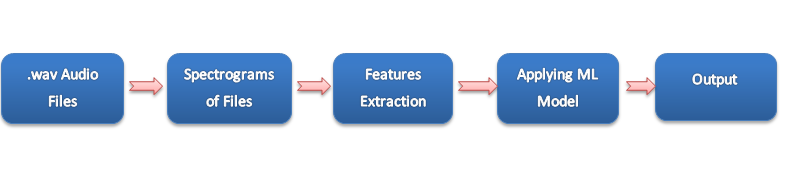
**Introduction**

# Who’s the Songster:

**Speaker recognition** is the identification of a person from characteristics of his/her voice is an important human trait most take for granted in natural human-to-human interaction/communication. It is also called **voice recognition**. There is a difference between speaker recognition (recognizing **who** is speaking) and [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition) (recognizing **what** is being said). These two terms are frequently confused, and “voice recognition” can be used for both.

Building a deep learning model that can recognize the voice of an artist (also known as **dynamic**voice identifier) for a given song with minimal training data. To create such a system, naturally the tool of choice would be an image classifier. Basically, this is the high-level view of what I am going to do in building this recognition task.



So, in the first step I am going to collect the audio files of different songsters(artists) in .wav format and then convert all audio files into a particular spectrogram (image representation) and after that extract features from images using CNN and then apply ML ensemble model gradient descent boosting.

As there was no publicly available data set for voice classification in songs so we have to create our own in order to train a model. So, for that our first attempt is to collect at least 40 songs each for your favourite artists in any format using any music app.

Downloaded at least 500 **solo** songs of artists in a different folder like Arijit, Atif, Arman, Shreya, Sunidhi, Somu Nigam etc (at least 40 each). In .TS format. For that we will use use **[ffmpeg](https://www.labnol.org/internet/useful-ffmpeg-commands/28490/" \t "_blank)**(most powerful and versatile command line tool for converting audio and video files) in python.

**The project aims to:**

* Classify singers for their particular songs.
* Achieve a wide range of users in certain region.
* Smooth, fast, efficient, reliable and easy to deploy python-based tool.
* Providing a user friendly and good visualization capability.

**SDLC Methodology**

This document plays a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process. SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models. As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far.

* Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project. The steps for Spiral Model can be generalized as follows:
* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled down system, and represents an approximation of the characteristics of the final product.

**A second prototype is evolved by a fourfold procedure:**

* Evaluating the first prototype in terms of its strengths, weakness, and risks.
* Defining the requirements of the second prototype.
* Planning a designing the second prototype.
* Constructing and testing the second prototype.
* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**Performance Requirements:**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system the existing system is completely dependent on the user to perform all the duties.

**Methodology:**

Following steps should be taken for analysis: -

* Data Integration
* Data Spectrogram Generation
* Data Cleaning

* Data Selection
* Model Training
* Pattern Evaluation
* Data Prediction

**External Interface Requirements:**

**SOFTWARE REQUIREMENTS:**

* Operating System Platform Used
* Windows 7 and above
* Python 3.6 and it’s various libraries

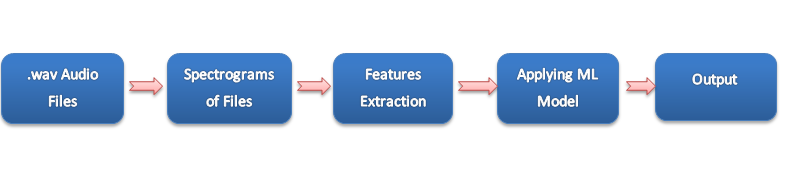
**HARDWARE REQUIREMENTS:**

* Hardware : Intel Dual core 2.2 GHz or more
* RAM : 1GB or more
* Hard disk : 40GB or more

**Requirement Analysis:**

System requirement is the ability of the system to meet the condition desired by the users. System requirement analysis is performed by grouping the needs into functional requirements and non-functional requirements.

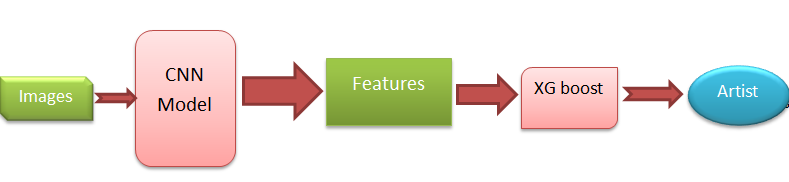
**Flow Chart:**



**Project Methodology:**

# Methodology / Planning of Work:

* Dataset i.e. audio files with wav extension.
* Generation of spectrogram for every audio file.
* Saving spectrogram images.
* **Using CNN Architecture.**
* CNN by modifying an existing VGG-16 and train it on spectrograms from 10 unique speakers.
* [**VGG-16**](https://www.quora.com/What-is-the-VGG-neural-network)(16-layers Convolutional Neural Network) CNN model in order to extract features from the spectrogram images.



**Data Exploration:**

* The total number of singers = 5.
* The number of songs per singer = 40.
* The number of spectrograms per song = 5.
* Total spectrograms = 1000.

**Implementation: Data Pre-processing:**

From the table in Exploring the Data above, we can see there are several features for each record that are non-numeric. Typically, learning algorithms expect input to be numeric, which requires that non-numeric features (called categorical variables) be converted. One popular way to convert categorical variables is by using the one-hot encoding scheme. One-hot encoding creates a "dummy" variable for each possible category of each non-numeric feature. For example, assume some Feature has three possible entries: A, B, or C. We then encode this feature into someFeature\_A, someFeature\_B and someFeature\_C.

Additionally, as with the non-numeric features, we need to convert the non-numeric target label, 'income' to numerical values for the learning algorithm to work. Since there are only two possible categories for this label ("<=50K" and ">50K"), we can avoid using one-hot encoding and simply encode these two categories as 0 and 1, respectively. In code cell below, you will need to implement the following:

* Use pandas.get\_dummies() to perform one-hot encoding on the 'features\_log\_minmax\_transform' data.
* Convert the target label 'income\_raw' to numerical entries.
* Set records with "<=50K" to 0 and records with ">50K" to 1.

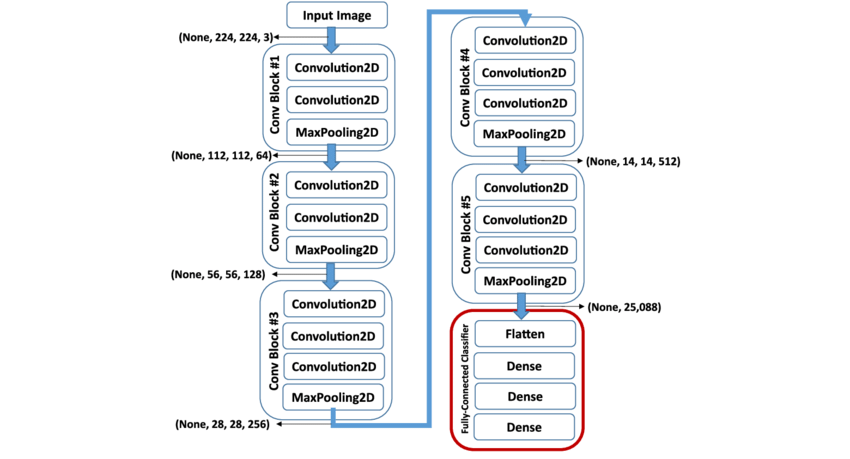
**Shuffle and Split Data:**

Now all categorical variables have been converted into numerical features, and all numerical features have been normalized. As always, we will now split the data (both features and their labels) into training and test sets. 80% of the data will be used for training and 20% for testing.

**Methodology:**

**VGGNet** consists of **16 convolutional layers** and is very appealing because of its very uniform architecture. Similar to AlexNet, only 3x3 convolutions, but lots of filters. Trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor.





**Modules/phases of the project:**

* Matplotlib
* NumPy
* Pandas
* Seaborn
* CNN
* Librosa

**Modules/phases covered**

**Feature:**

In Machine Learning feature means a property of your training data. Or you can say a column name in your training dataset.

**Label**: The output you get from your model after training it, is called label.

**Matplotlib:**

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc.

**VGG16:**

The VGG16 architecture consists of twelve convolutional layers, some of which are followed by maximum pooling layers and then four fully-connected layers and finally a 1000-way SoftMax classifier.

Simplified VGG16 Architecture

**First and Second Layers:**  
The input for AlexNet is a 224x224x3 RGB image which passes through first and second convolutional layers with 64 feature maps or filters having size 3×3 and same pooling with a stride of 14. The image dimensions changes to 224x224x64.   
Then the VGG16 applies maximum pooling layer or sub-sampling layer with a filter size 3×3 and a stride of two. The resulting image dimensions will be reduced to 112x112x64.

**Third and Fourth Layer:**  
Next, there are two convolutional layers with 128 feature maps having size 3×3 and a stride of 1.   
Then there is again a maximum pooling layer with filter size 3×3 and a stride of 2. This layer is same as previous pooling layer except it has 128 feature maps so the output will be reduced to 56x56x128.

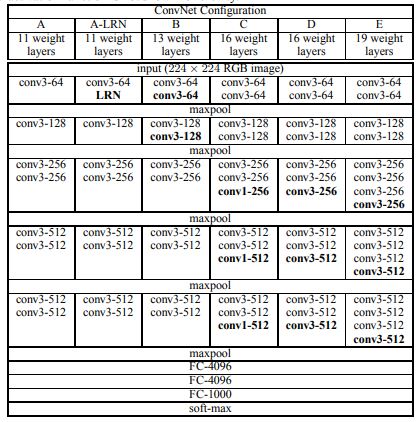
**Fifth and Sixth Layers:**  
The fifth and sixth layers are convolutional layers with filter size 3×3 and a stride of one. Both used 256 feature maps.  
The two convolutional layers are followed by a maximum pooling layer with filter size 3×3, a stride of 2 and have 256 feature maps.

**Seventh to Twelfth Layer:**  
Next are the two sets of 3 convolutional layers followed by a maximum pooling layer. All convolutional layers have 512 filters of size 3×3 and a stride of one. The final size will be reduced to 7x7x512.

**Thirteenth Layer:**  
The convolutional layer output is flattening through a fully connected layer with 25088 feature maps each of size 1×1.

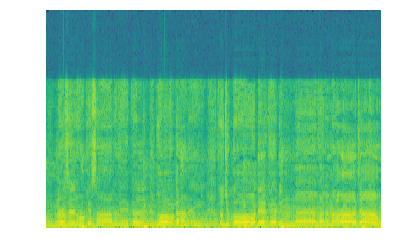
**Fourteenth and Fifteenth Layers:**  
Next is again two fully connected layers with 4096 units.

**Output Layer:**  
Finally, there is a SoftMax output layer ŷ with 1000 possible values.

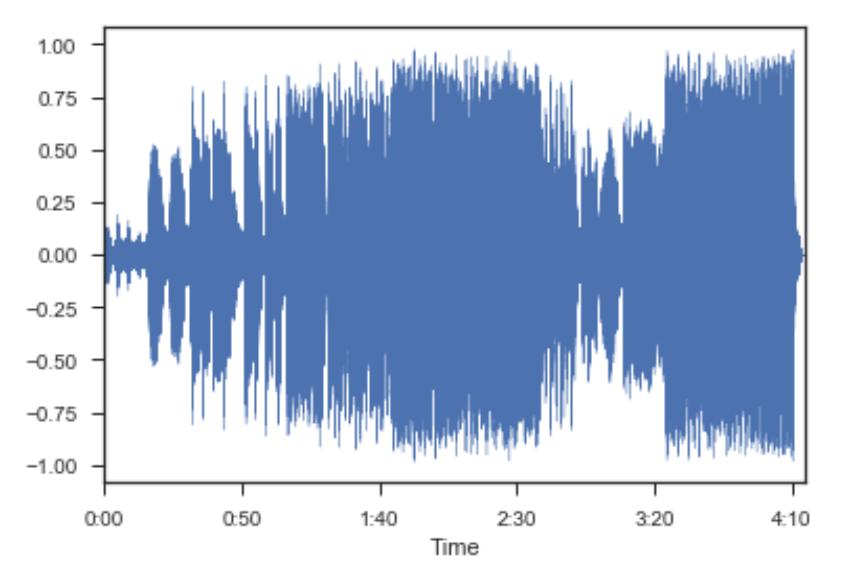


**Screenshots**

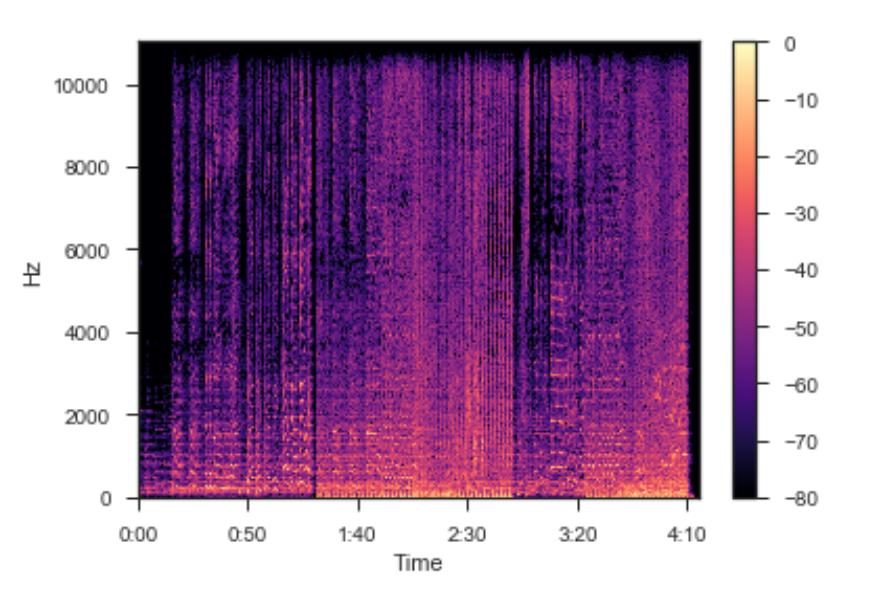
**Spectrogram:**



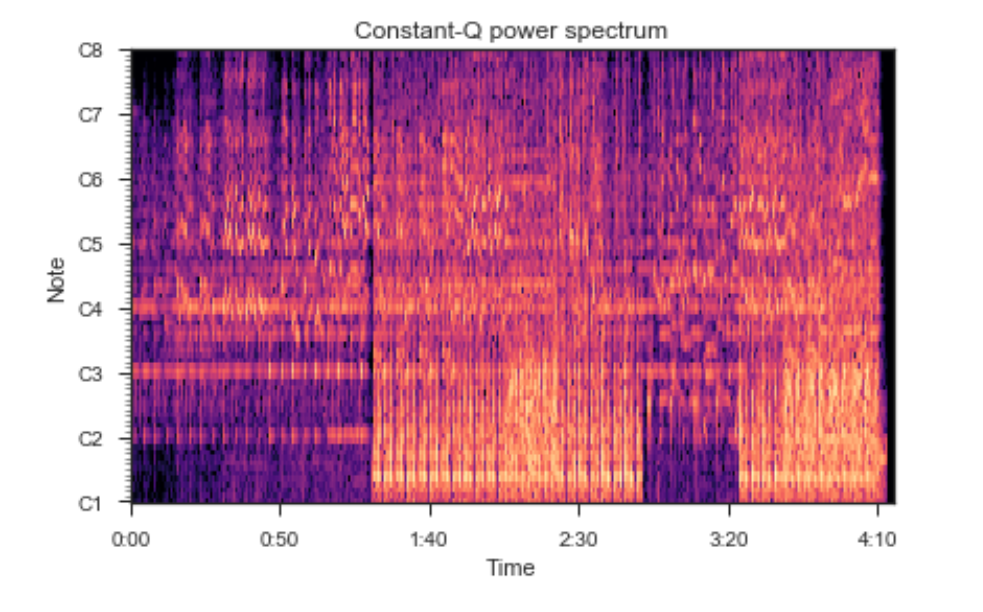
**Wave Plot:**



**Spectrogram with Heat Map:**



**Constant-Q power spectrum:**



**Harmonic + Percussive:**

