CHAPTER 1

INTRODUCTION

1.1 Context

By audio extrapolation we mean finding or extending out the audio using the old audio samples when only a part of audio is being provided to the model. A naïve way of understanding audio extrapolation is by taking a very simple example. Suppose, a person is calling his friend in Mumbai over a telephone, due to a moderately congested network, the friend on the other side of the call is only able to hear 'beaut…', followed by a time interval of noise. Audio extrapolation would refer to extending this part and finding out the entire word 'beautiful'.

Formal definition: The extrapolation of a discrete signal section means calculation of new previously unknown signal samples extending the given signal.

1.2 Extrapolation vs Interpolation

When we talk about most of the machine learning regression examples, we mainly consider finding a pattern by training the model using the training set. The predicted values are formed in a close proximity of the actual regressor line. Conversely, in extrapolation we tend we move away from the dataset and tend to predict the values that lie at some distance from the actual dataset.

We can clearly understand the difference using the following diagram:

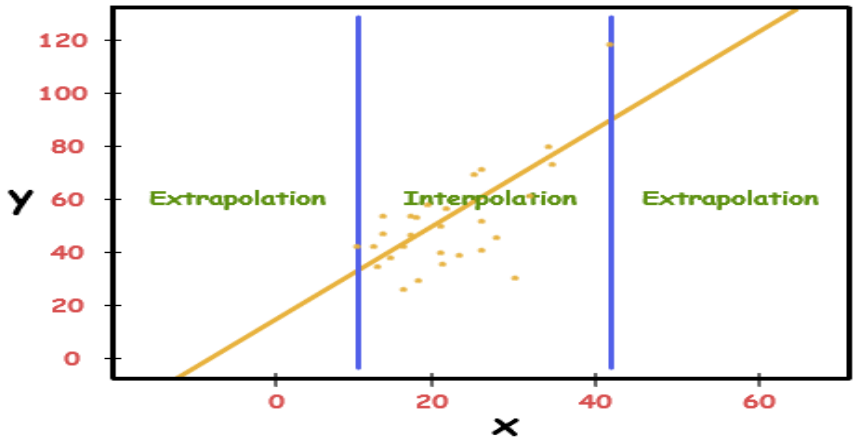

Fig 1.2: Interpolation vs Extrapolation

1.3 Motivation

Audio Classification

Audio classification is a fundamental problem in the field of audio processing. The task is essentially to extract features from the audio, and then identify which class the audio belongs to. Many useful applications pertaining to audio classification can be found in the wild – such as genre classification, instrument recognition and artist identification.

A common approach to solve an audio classification task is to pre-process the audio inputs to extract useful features, and then apply a classification algorithm on it. For example, in the case study below we are given a 5 second excerpt of a sound, and the task is to identify which class does it belong to – whether it is a dog barking or a drilling sound.

Audio Fingerprinting

The aim of audio fingerprinting is to determine the digital "summary" of the audio. This is done to identify the audio from an audio sample. Shazam is an excellent example of an application of audio fingerprinting. It recognises the music on the basis of the first two to five seconds of a song. However, there are still situations where the system fails, especially where there is a high amount of background noise.

Telecommunications

When one person is speaking to another person and in between the call some of the audio samples gets corrupted then by using the audio extrapolation audio samples can be recovered.

Finding in between samples

Missing audio samples can also be interpolated by using same machine learning techniques used in our project to find in between samples of any audio. K Neighbor regression is the best approach to interpolate the missing audio samples.

CHAPTER 2

LITERATURE SURVEY

The internet is increasingly utilized as the transport framework for Nowadays communication. The common technique for the transmission of speech, Voice over IP (VoIP), has been used for about20 years and replaced analog as well as ISDN telephony extensively. Also the spreading of musical content, called Audio over IP (AoIP), has been well-established. However, this trend mainly applies for broadcast scenarios but not for low-latency, bidirectional communication[4].
Voice over Internet Protocol is a category of hardware and software that enables people to use the Internet as the transmission medium for telephone calls by sending voice data in packets using IP rather than by traditional circuit transmissions of the PSTN.

One advantage of VoIP is that the telephone calls over the Internet do not incur a surcharge beyond what the user is paying for Internet access, much in the same way that the user doesn't pay for sending individual emails over the Internet.

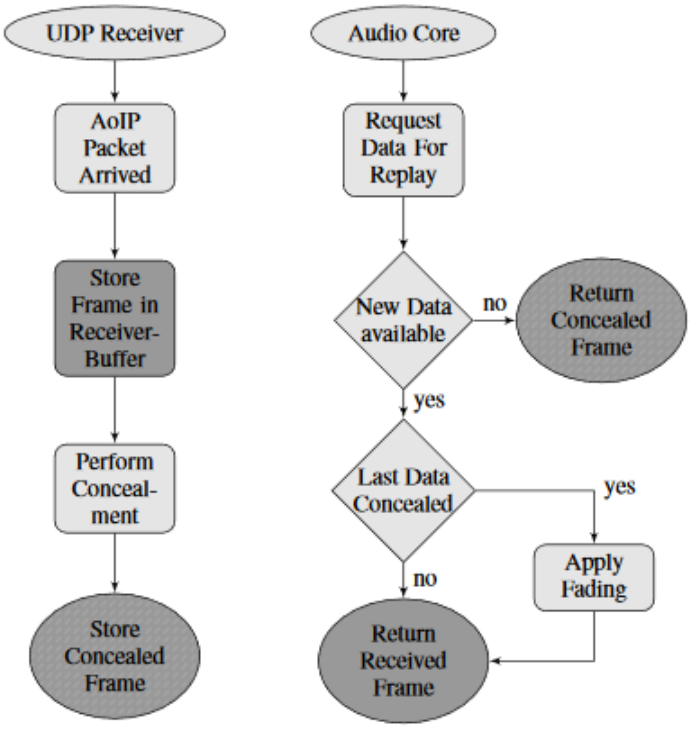

Fig 2.1 Flow Chart of concealment mechanism in AoIP software.

To allow the transmission of continuous, analog signals it is necessary to convert the signal into a digital representation, fragment it into blocks, and encapsulate it into an IP packet. Thereafter, the actual transport over the IP network can occur before a receiver can extract the audio segment from the packet, and convert it to the analogue domain again for the purpose of replaying it. All these steps introduce a certain amount of delay, conflicting with the requirement of low latency in many AoIP applications. Whenever an intact AoIP packet arrives at one of the UDP sockets, which is bound to a specific sender of another client, it is written to a free slot in the corresponding receiver buffer (marked in dark gray). The entire or just a fraction of the receiver buffer can then be used to perform the actual concealment using the techniques described above. The result is stored in a separate buffer (shaded dark gray). Note, that the extrapolated audio frame is longer than the actual audio frames of size N[4].

Every time the audio core gets activated, and the audio processing callback is triggered, a stereo mix of the current audio frames of all receivers is written to the output buffer of the sound card. Therefore, the audio core requests data from all receiver buffers. If a receiver buffer does not contain new data, it will return the last concealed audio frame truncated to length of N.
Sometimes while sending the audio signals some of the audio signals get lost, so to recover them we will make use of linear interpolation method.

In Linear interpolation, graph is plotted of those audio signals which have been received correctly, in order to determine the missing audio signal, choose two points which are closest to missing audio signal. Form two linear equations and parameters (a1 and a2) is obtained. By using these two parameters missing sample is obtained[1].

Audio bandwidth extrapolation discovered some hidden structure in music signals; in effect, to map a seemingly high dimensional space of audio signals into a lower dimensional space in a manner which is invertible to our perception. The combination of MDCT and linear/PCA based mappings was thought to be inappropriate for such a task[2].

The most serious weakness of this approach was that the majority of audio signals that we tested had frequency coefficients that were largely uncorrelated. This right away ruled out frequency-domain based linear estimation for most audio signals[2].

CHAPTER 3

Requirements

Anaconda package.

IDE : Spyder (from Anaconda package).

Programming language: Python 3.x

Python libraries like “os”, “numpy”, “scipy”, “pandas”, “winsound”, “matplotlib”, “seaborn”.

Minimum System Requirement: 4GB RAM (or higher), Graphics Card, i5 or higher processor.

3.1 Anaconda Package

With over 6 million users, the open source Anaconda Distribution is the easiest way to do Python data science and machine learning. It includes 250+ popular data science packages and the *conda* package and virtual environment manager for Windows, Linux, and MacOS. Conda makes it quick and easy to install, run, and upgrade complex data science and machine learning environments like Scikit-learn, TensorFlow, and SciPy. Anaconda Distribution is the foundation of millions of data science projects as well as Amazon Web Services' *Machine Learning AMIs* and *Anaconda for Microsoft* on Azure and Windows.

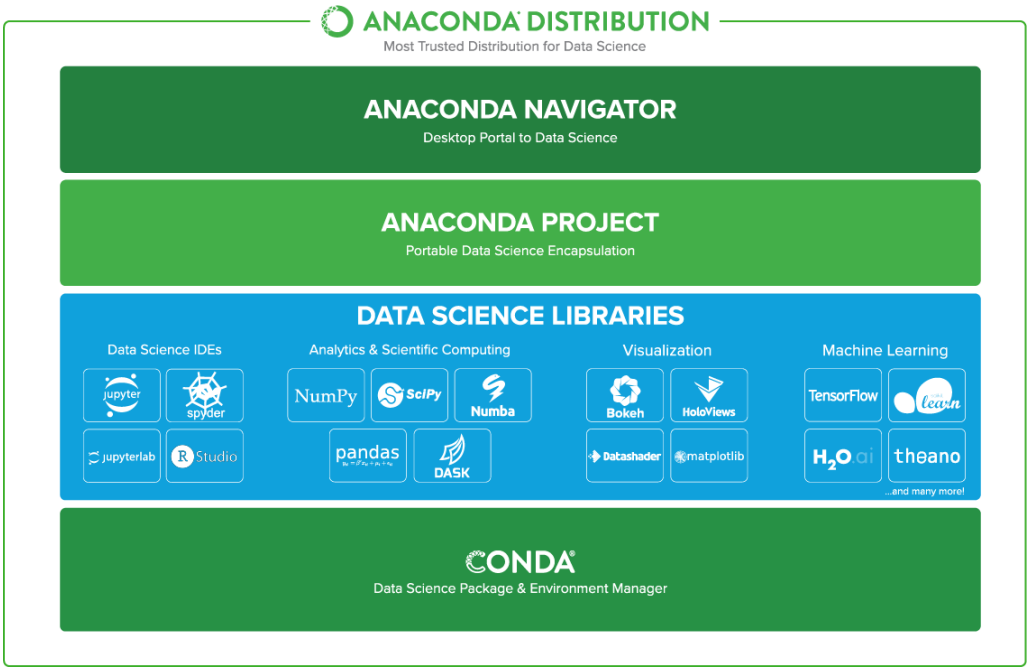

Fig 3.1: Anaconda Distribution

3.2 IDE (Spyder)

Spyder is the Scientific Python Development Environment:

- a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features
- and a numerical computing environment thanks to the support of *IPython* (enhanced interactive Python interpreter) and popular Python libraries such as *NumPy* (linear algebra), *SciPy* (signal and image processing) or *matplotlib* (interactive 2D/3D plotting).

Spyder may also be used as a library providing powerful console-related widgets for your PyQt-based applications – for example, it may be used to integrate a debugging console directly in the layout of your graphical user interface.

3.3 Programming Language (Python 3.x)

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later on many of its important features have been backported to be compatible with version 2.7.

3.4 Python Libraries

- "OS": The OS module in python provides a way of using OS dependent functionality. The function that the OS module provides allows you to interface with the underlying operating system that Python is running on.
- "winsound": The **winsound** module provides access to the basic sound-playing machinery provided by Windows platforms. It includes functions and several constants.

  winsound.**PlaySound**(*sound, flags*)

  Call the underlying **PlaySound()** function from the Platform API. The *sound* parameter may be a filename, audio data as a string, or None. Its interpretation depends on the value of *flags*, which can be a bitwise ORed combination of the constants described below. If

the *sound* parameter isNone, any currently playing waveform sound is stopped. If the system indicates an error, **RuntimeError** is raised.

- "Numpy"

  NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

  Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

- "Scipy"

  SciPy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling systems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

- "Pandas"

  Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

CHAPTER 4

METHODOLOGY

4.1 Models

4.1.1 Linear Regression

Linear Regression establishes a relationship between dependent variables(y) and one or more independent variables(x) using a best fit straight line (also known as regression line). In audio extrapolation, previous n audio samples can be used to predict the $(n+1)^{th}$ audio sample. The idea behind simple linear regression is to "fit" the observations of two variables into a linear relationship between them. Graphically, the task is to draw the line that is "best-fitting" or "closest" to the points (x_i, y_i) where x_i and y_i are observations of the two variables which are expected to depend linearly on each other[3].

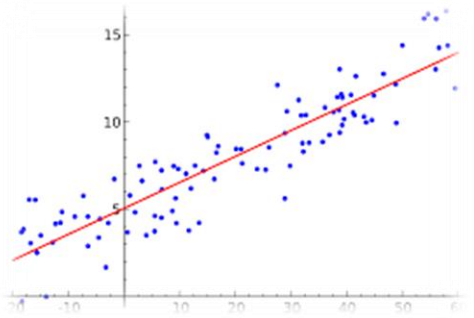

Fig 4.1.1 Linear Regression Example

How Linear Regression works?

$H(x) = \Theta 0(x) + \Theta 1(x)$

Where Θ is an intercept and $\Theta 1$ is the slope. The error between the predicted sample and actual sample is given by

$J(\Theta 0,\Theta 1) = 1/2m \sum (h(x)-y)^2$

Gradient descent approach is used for the minimization of the cost function.

Repeat until convergence

{

$\Theta j = \Theta j - \alpha * (d/d(\Theta j))\ [J(\Theta 0, \Theta 1)]$

}

Where j can be 0 and 1 and α is the learning rate which can be carefully chosen to strike out a balance fast and slow step size. If α is very large then it may overshoot the minima and if it is too small then convergence will take very long time.

Application of Linear Regression

Regression is a common process used in many applications of statistics in the real world. There are two main types of applications:

- **Predictions:** After a series of observations of variables, regression analysis gives a statistical model for the relationship between the variables. This model can be used to generate predictions: given two variables x and y the model can predict values of y given future observations of x. This idea is used to predict variables in countless situations, e.g. the outcome of political elections, the behavior of the stock market, or the performance of a professional athlete.
- **Correlation:** The model given by a regression analysis will often fit some kinds of data better than others. This can be used to analyze correlations between variables and to refine a statistical model to incorporate further inputs: if the model describes certain subsets of the data points very well, but is a poor predictor for other data points, it can be instructive to examine the differences between the different types of data points for a possible explanation. This type of application is common in scientific tests, e.g. of the effects of a proposed drug on the patients in a controlled study

Linear Regression in Audio Extrapolation

By using gradient descent approach, the best line (line having least error) can be fit into the data and the rest of the samples can be recovered by using the points lie on the line. In ideal case error

should be zero but to check how perfectly the predicted audio matches with actual audio, absolute mean error is used.

Absolute mean square error should be used to predict how much variation is present between predicted audio samples in comparison to original audio samples. In an ideal case absolute mean error should be zero.

4.1.2 K Neighbor Regression

It's a simple, effective and non-parametric regression algorithm which is used for an extrapolation of audio samples. It doesn't make any assumption about the probability distribution of the input. It's a lazy learning method that generalizes data in the training phase rather than the testing phase. Because of lazy learning it can quickly adapt to changes, since it's not expecting a generalized dataset.

Regression involves establishing a relationship between the input points and rest of the data. Determining a distance between different neighbors can be performed using different notion of distance. Here we are using Minkowski distance.

$D(x , x') = \sqrt{\sum}|xd – x'd|p$

Where p = 2 and it becomes Euclidean distance. The value returned is the average value of the input's k neighbors.

An extrapolation of audio samples can be performed by the average of k nearest audio samples.

PARAMETRIC SELECTION

To determine the number of neighbors to consider when running the algorithm(k), a common method involves choosing the optimal k for a validation set (the one that reduces the percentage of error) and using for the test set. In general, a higher k reduces noise and localized variables but allows for more error near decision boundary and vice versa.

Mean Square error is used to measure how well original audio samples matches with predicted audio samples.

4.1.3 Decision Tree Regression

Multi output Decision Tree regression

Multi output decision tree regression is a supervised learning algorithm with several outputs to predict, that is when y is a 2D array of size [n_input_audio_samples , n_output_audio_samples].

Multioutput regression can be categorized as

1 Problem Transformation Method

2 Algorithm Adaptation Method

1 Problem Transformation Method: It is also known as local method that transform the multi output problem into independent single output problem each solved using single output regression algorithm[5].

2 Algorithm Adaptation Method: It is also known as global or big bang method that adapt a specific single output method to directly handle multiple output datasets.

Adaptive Algorithm method has two advantages over problem transformation methods

a) A single multitarget regression tree is usually smaller than the total size of the individual single target trees for all variables, thus it requires a lower training time since only a single estimator is built.
b) Multitarget regression tree better identifies the dependencies between the different target variables. Hence the generalization accuracy of the resulting estimator may often be increased[5].

Mathematics behind Decision Tree Regressor

Given training vectors $x_i \in R^n$, i=1,…, l and a label vector $y \in R^l$, a decision tree recursively partitions the space such that the samples with the same labels are grouped together.

Let the data at node $m$be represented by Q. For each candidate split $\theta = (j, t_m)$consisting of a feature $j$and threshold t_m, partition the data into $Q_{left}(\theta)$and $Q_{right}(\theta)$subsets

$$Q_{left}(\theta) = (x, y)|x_j <= t_m$$
$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta)$$

The impurity at $m$is computed using an impurity function $H()$, the choice of which depends on the task being solved (classification or regression)

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$

Select the parameters that minimises the impurity

$$\theta^* = \text{argmin}_\theta \, G(Q, \theta)$$

Recurse for subsets $Q_{left}(\theta^*)$and $Q_{right}(\theta^*)$until the maximum allowable depth is reached, $N_m < \min_{samples}$or $N_m = 1$

Criteria for Regression

Output of an extrapolated audio is a continuous value, then for node m, representing a region $R_m$with $N_m$observations, common criteria to minimise as for determining locations for future splits are Mean Squared Error, which minimizes the L2 error using mean values at terminal nodes, and Mean Absolute Error, which minimizes the L1 error using median values at terminal nodes.

Mean Squared Error:

$$c_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$
$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - c_m)^2$$

Mean Absolute Error:

$$\bar{y_m} = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y_m}|$$

where $X_m$is the training data in node m

4.1.4 Random Forest Regression

It's an extension of decision trees. Here each tree in the ensemble is built from the ensemble drawn from a sample drawn with replacement (i.e bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among the random set of features . As a result of this randomness, the bias of the forest slightly increases (with respect to the bias of a single non random tree) but due to averaging its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

Algorithm:

(x1,y1) , (x2,y2) , … , (xn,yn) = dataset(d)

For I = 1, 2 , … , B:

{

Choose bootstrap sample Di from D

Construct tree Ti using D: such that

At each node , choose random subset of m features and create splitting on those features.

}

4.1.5 Support Vector Regression

The method of Support Vector Classification can be extended to solve regression problems. This method is called Support Vector Regression.

The model produced by support vector classification depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

Mathematics behind SVR

Given training vectors $x_i \in \mathbb{R}^p$, i=1,…, n, and a vector $y \in \mathbb{R}^n$ ε-SVR solves the following primal problem:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$

$$\text{subject to } y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i,$$
$$w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*,$$
$$\zeta_i, \zeta_i^* \geq 0, i = 1, ..., n$$

Its dual is

$$\min_{\alpha,\alpha^*} \frac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \varepsilon e^T (\alpha + \alpha^*) - y^T (\alpha - \alpha^*)$$

$$\text{subject to } e^T (\alpha - \alpha^*) = 0$$
$$0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, ..., n$$

where $e$is the vector of all ones, $C > 0$is the upper bound, $Q$is an $n$by $n$positive semidefinite matrix, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.

The decision function is:

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + \rho$$

These parameters can be accessed through the members `dual_coef_` which holds the difference $\alpha_i - \alpha_i^*$, `support_vectors_` which holds the support vectors, and `intercept_` which holds the independent term ρ [5].

4.2 Audio Visualization

4.2.1 using matplotlib library

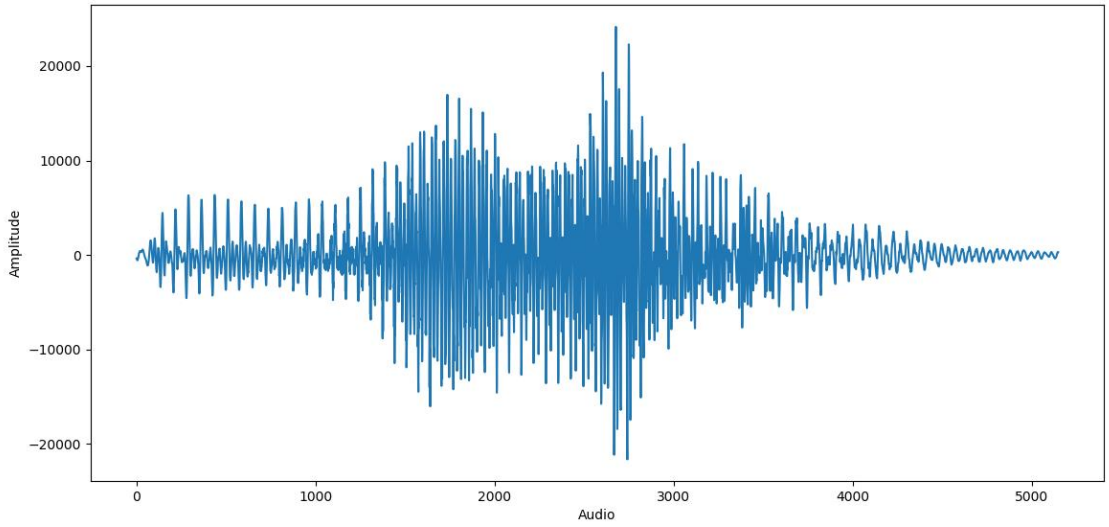

Fig 4.2.1 Visualization using matplotlib

Audio signal is viewed in the above diagram by using matplotlib.pyplot() library, showing amplitude of each audio sample within the specific audio.

4.2.2 using seaborn library

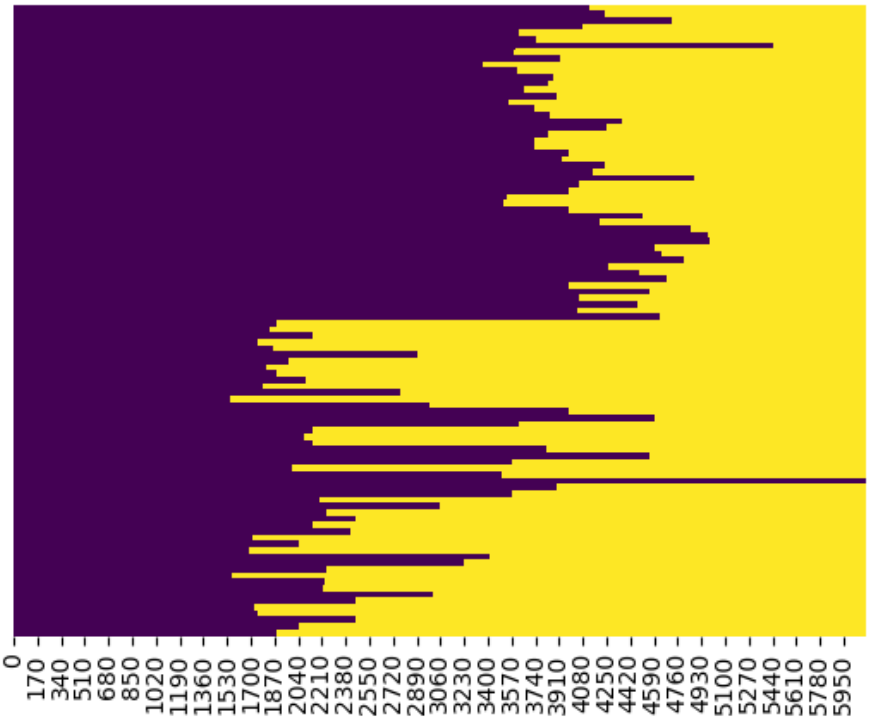

Fig 4.2.2 Visualization using seaborn

Audio of 3 people is visualized based upon the filled in audio samples. In the above map, the purple region specifies the filled in values and the yellow region specifies null values.

CHAPTER 5

IMPLEMENTATION

5.1 Workflow

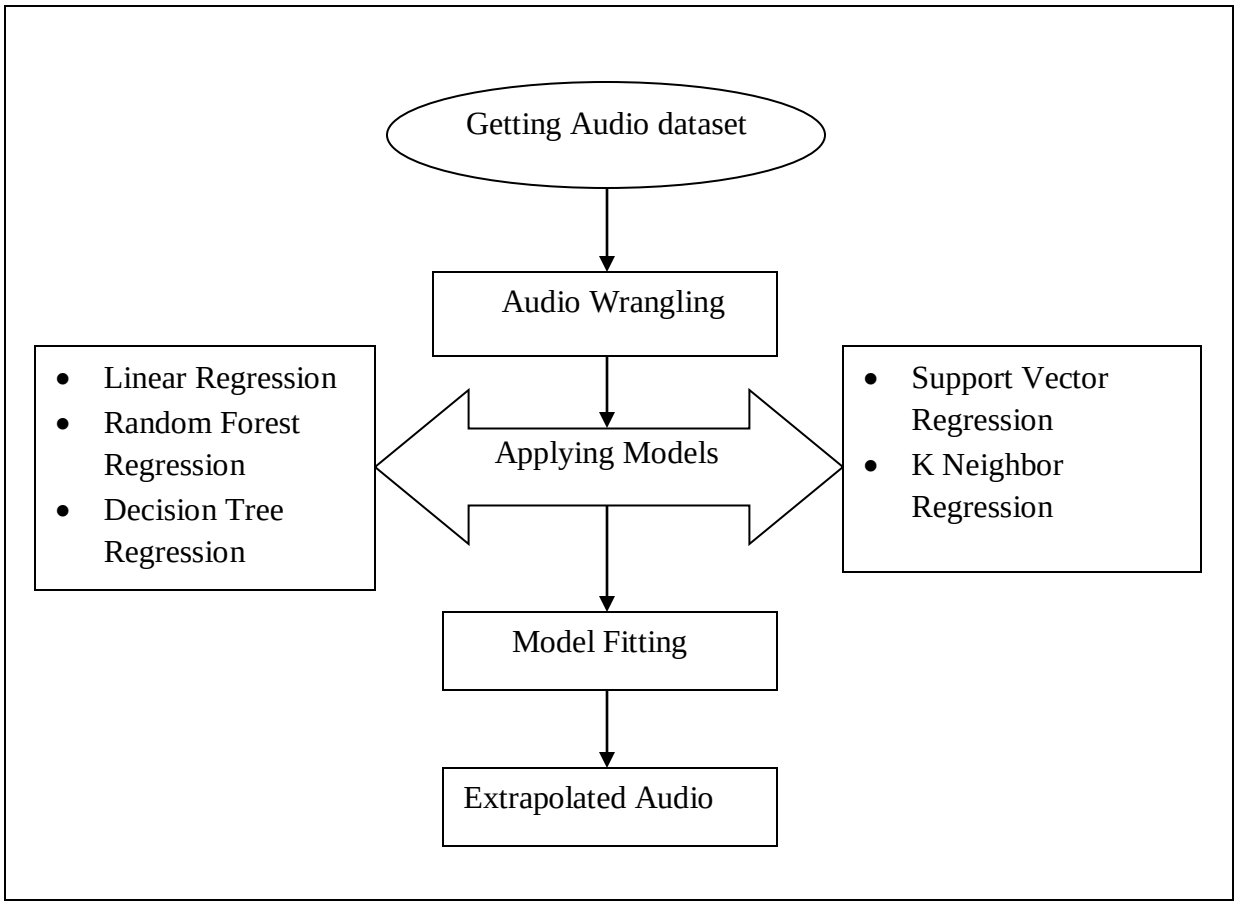

Fig 5.1 Workflow

5.2 Dataset

Dataset for our project is taken from the following link:

https://github.com/Jakobovski/free-spoken-digit-dataset/recordings

A simple audio/speech dataset consisting of recordings of spoken digits in wav files at 8kHz. The recordings are trimmed so that they have near minimal silence at the beginnings and ends.

- 150 audio samples of 3 persons named Nicola, Jackson and Theo has been taken.
- English pronunciations.

5.3 Data Preparation

Audio has been collected from the specified folders and it is in the analog form because of which it is not possible to use it with the regression models . It has to be discritized using python library named scipy.io.wavfile library which returns a tuple of sample rate of audio and the array of audio.

Now it has to be converted into a dataframe. In our project we have taken the audio of two people saying one for different intervals of time. Hence data wrangling is applied on it by using dropna() function of dataframe having a threshold of 2. Remaining null values has to be filled by using fillna() function taking the mean along the horizontal axis.

Data is sliced by taking 49 audio samples each of Jackson and Theo respectively and training can be done. 50^{th} sample of Jackson and 50^{th} sample of Theo is taken for testing the data.

5.4 Forming training and test splits

Input is taken from the user.

The samples within the audio are sliced using python slicing into train set and test set

X_train = train_data[: , : int(input_audio_percent * no_of_audio_samples)]

We have taken all the rows and only the percentage of samples provided by the user as training set. Similar is done for test set as well.

CHAPTER 6

RESULTS

6.1 With 20% input

6.1.1 Linear Regression

Absolute mean error: 1516

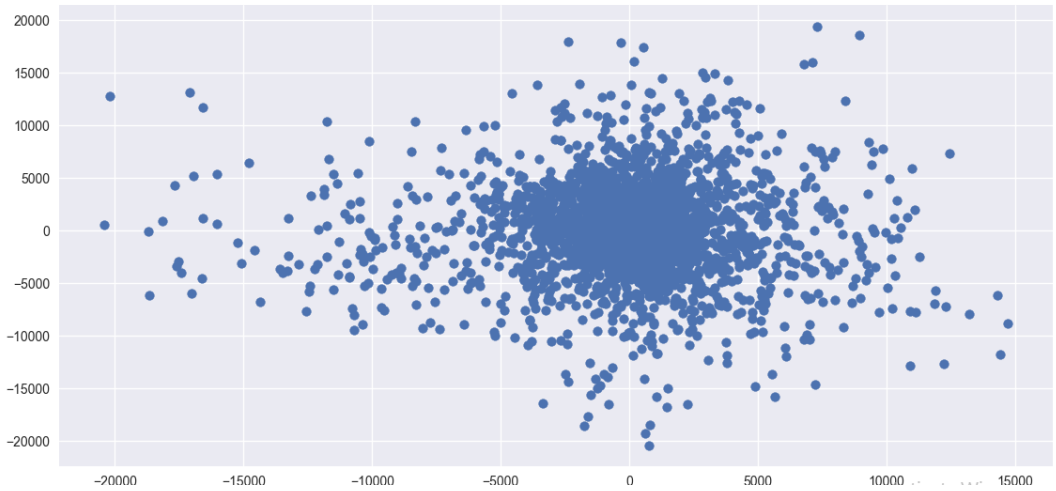

Fig 6.1.1.1: Linear regression with 20% input (scatterplot)

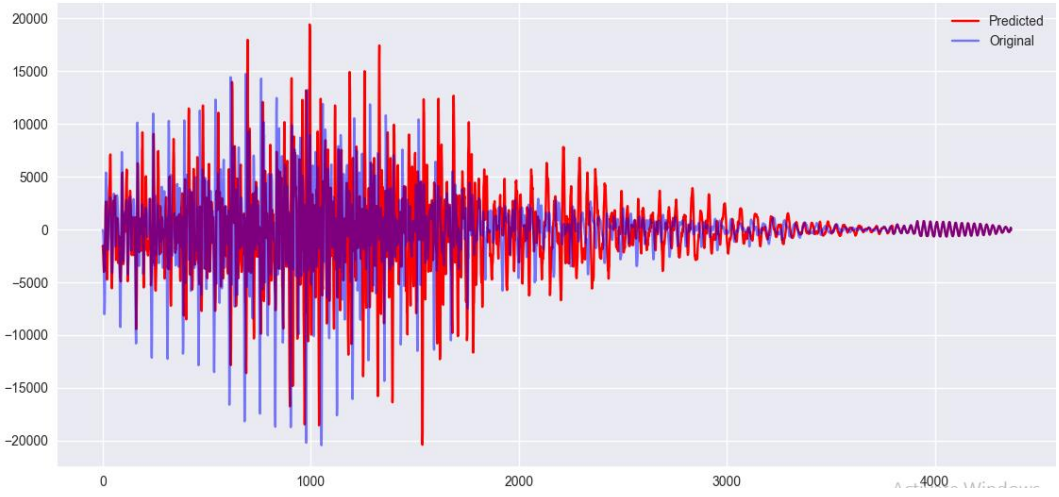

Fig 6.1.1.2: Linear regression with 20% input (heartbeat plot)

6.1.2 K Neighbor Regression

Absolute Mean error: 1003.38

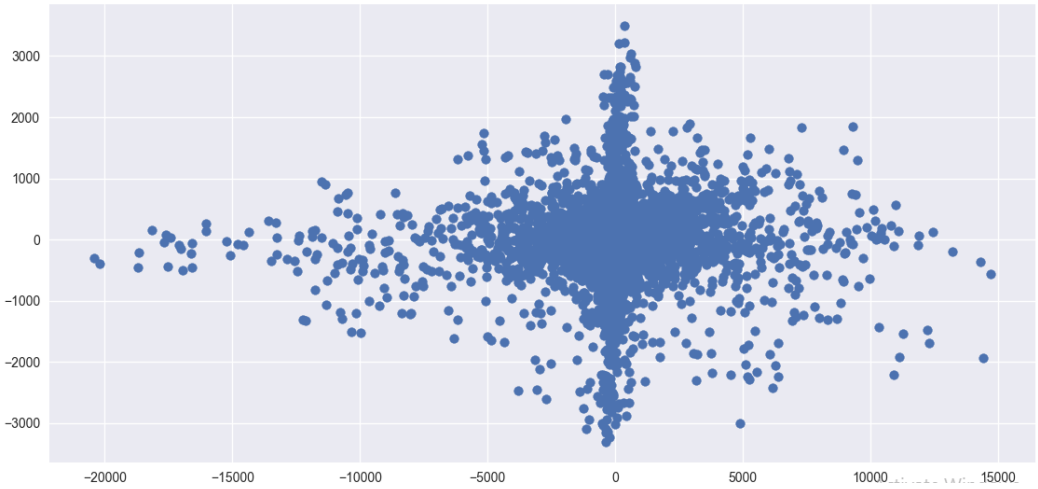

Fig 6.1.2.1.: K Neighbor regression with 20% input (scatterplot)

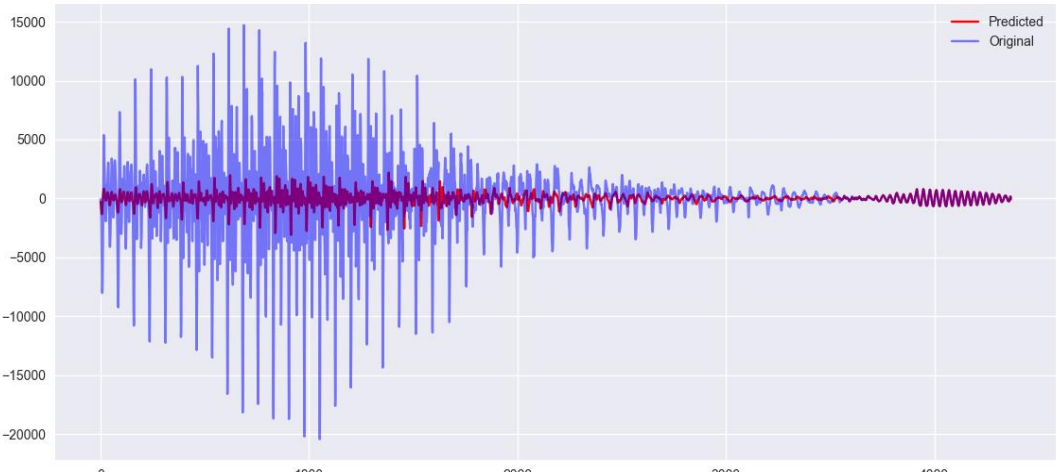

Fig 6.1.2.2: K Neighbor regression with 20% input (heartbeat plot)

6.1.3 Decision Tree Regression

Absolute mean error: 867.77

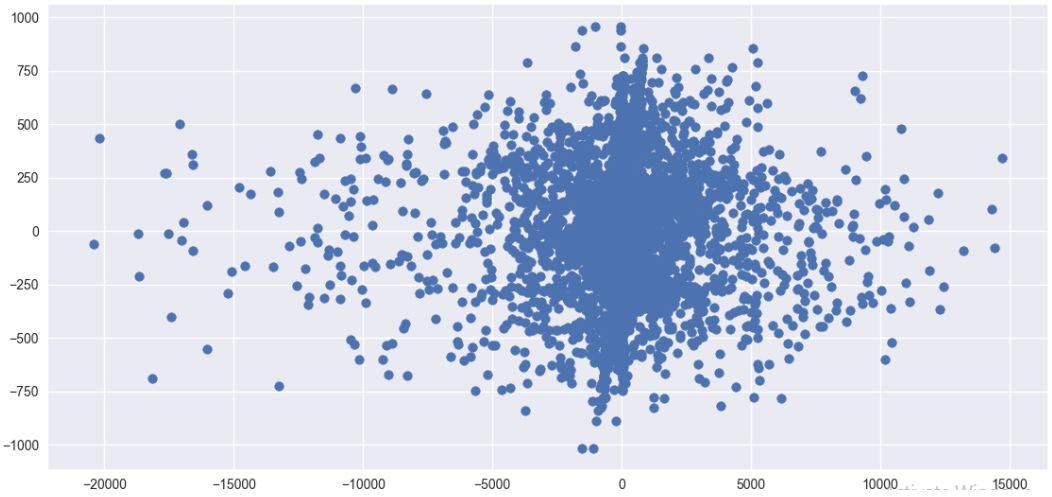

Fig 6.1.3.1: Decision Tree regression with 20% input (scatterplot)

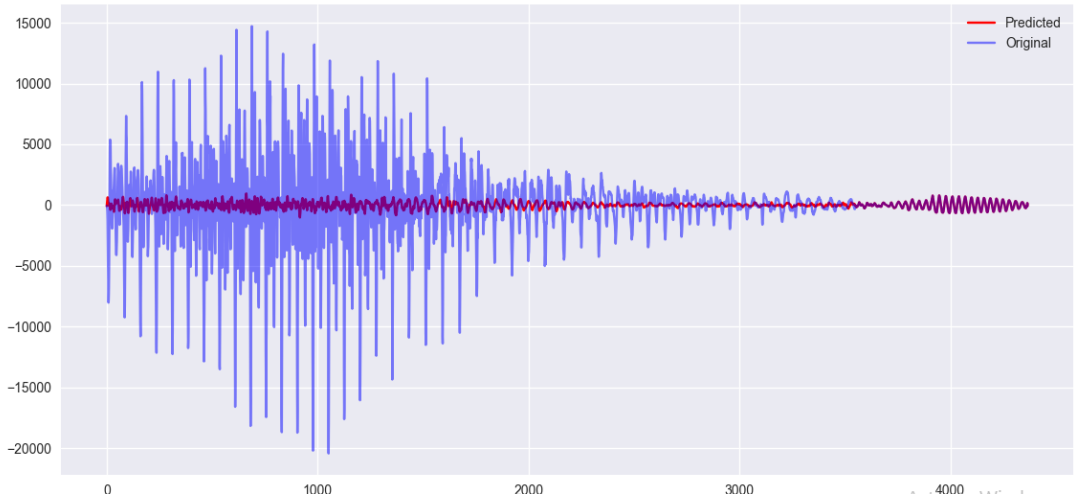

Fig 6.1.3.2: Decision Tree regression with 20% input (heartbeat plot)

6.1.4 Random Forest Regression

Absolute Mean Error: 866.196

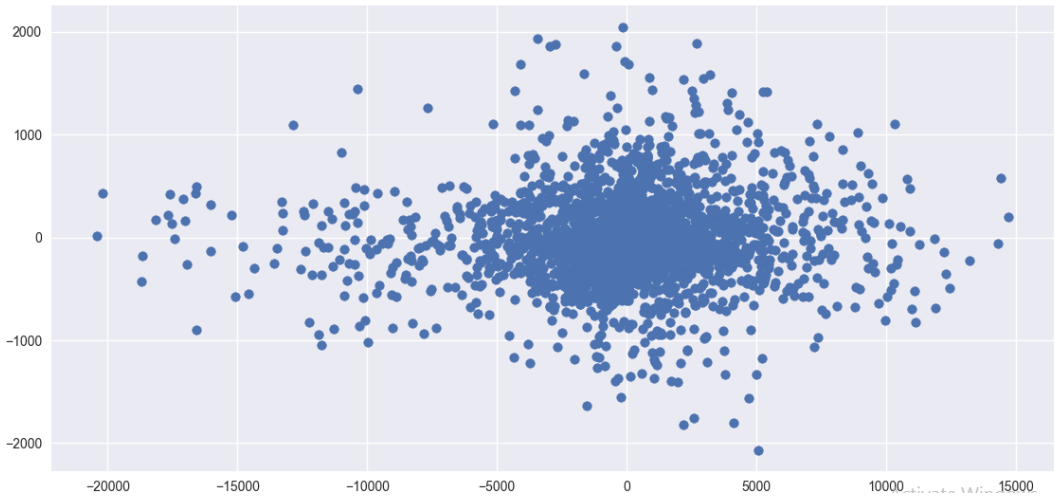

Fig 6.1.4.1: Random Forest regression with 20% input (scatterplot)

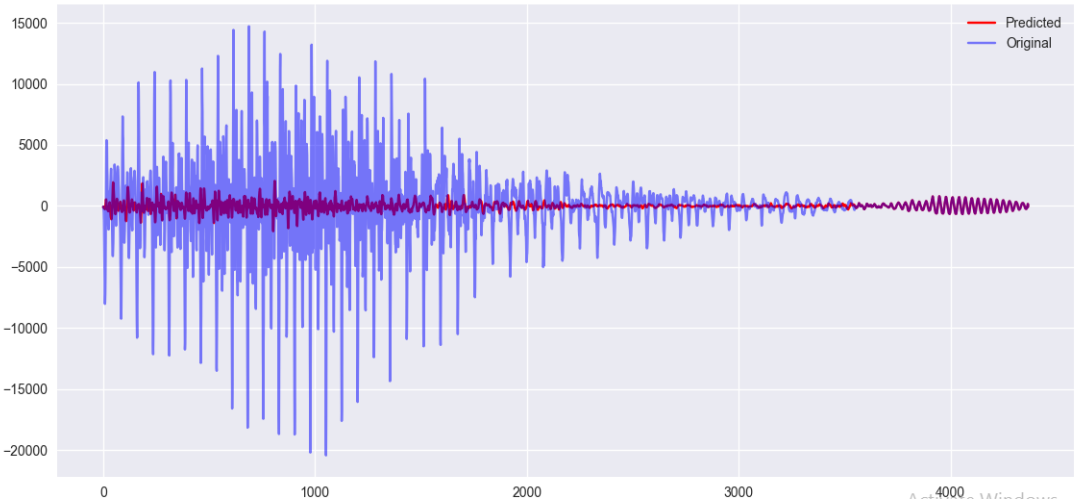

Fig 6.1.4.2: Random Forest regression with 20% input (heartbeat plot)

6.1.5 Support Vector Regression

Absolute Mean Error: 2450

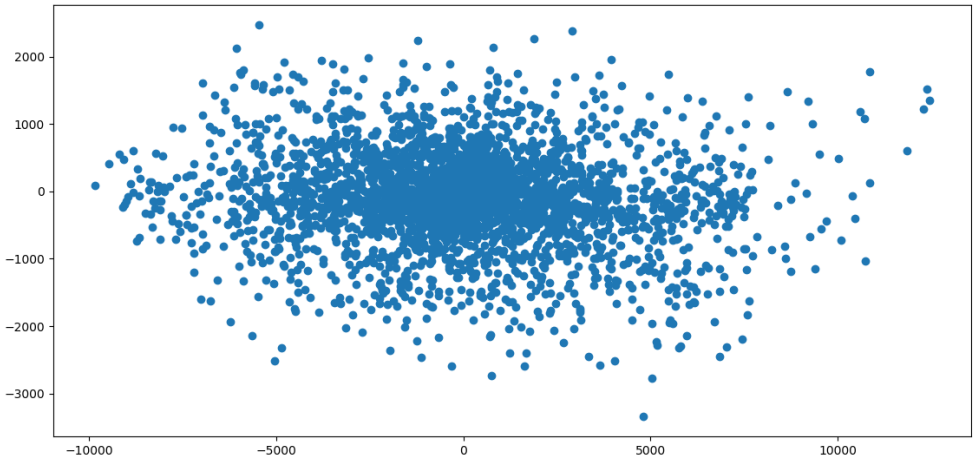

Fig 6.1.5.1: Support Vector regression with 20% input (scatterplot)

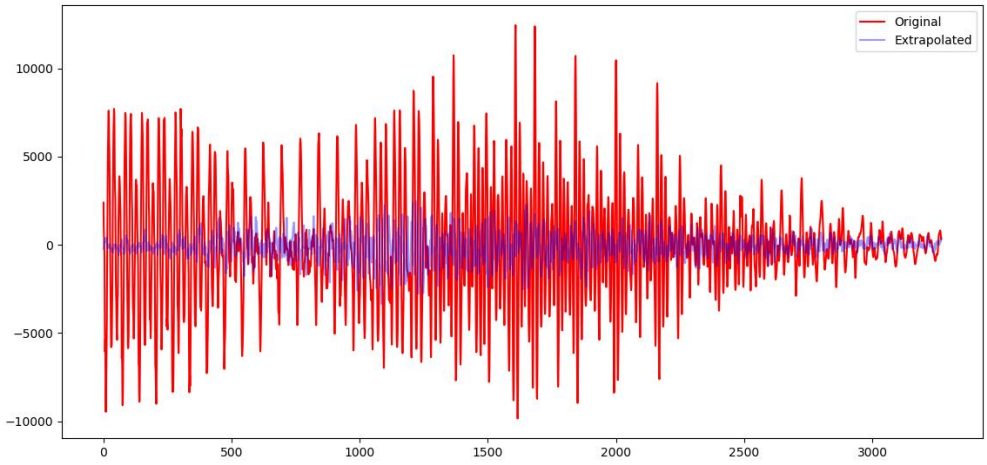

Fig 6.1.5.2: Support Vector regression with 20% input (heartbeat plot)

6.2 With 50% input

6.2.1 Linear Regression

Absolute Mean Error: 325.922

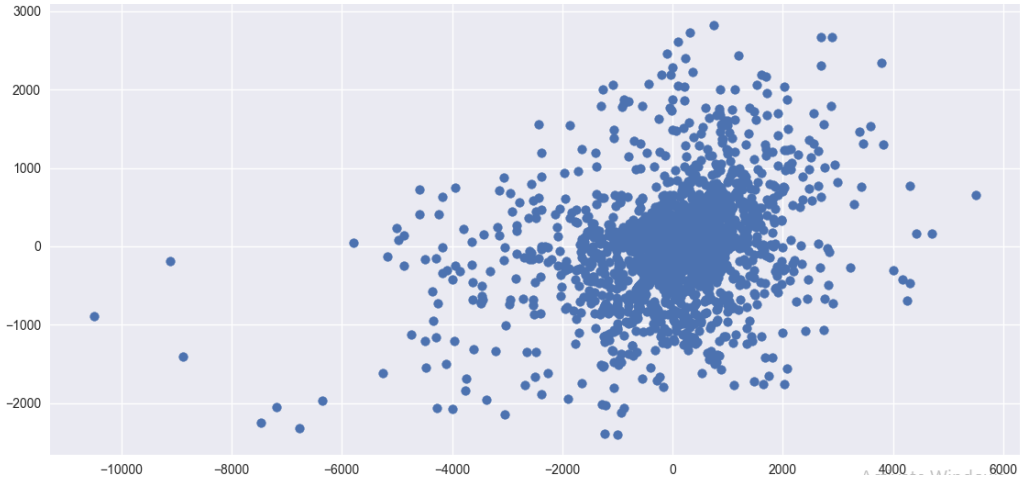

Fig 6.2.1.1: Linear Regression with 50% input (scatterplot)

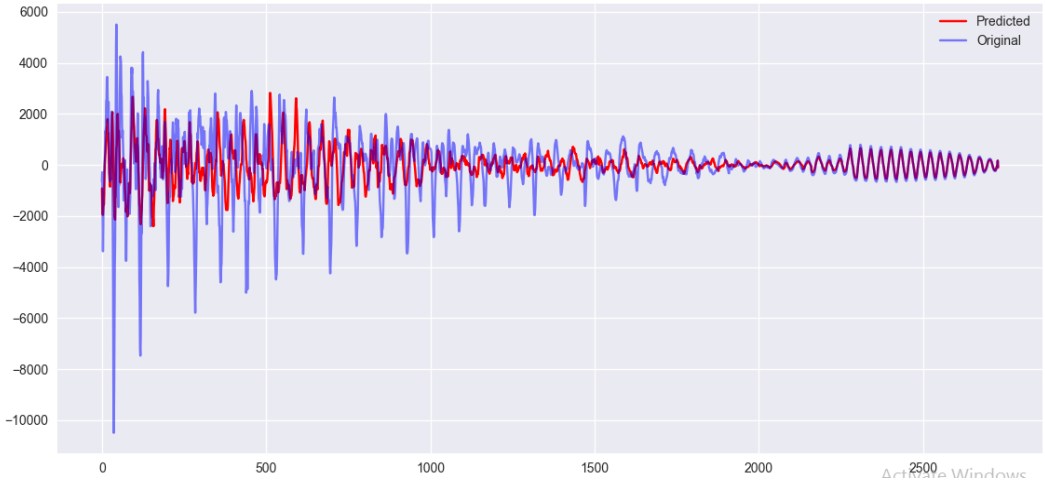

Fig 6.2.1.2: Linear Regression with 50% input (heartbeat plot)

6.2.2 K Neighbor Regression

Absolute Mean Error: 307.196

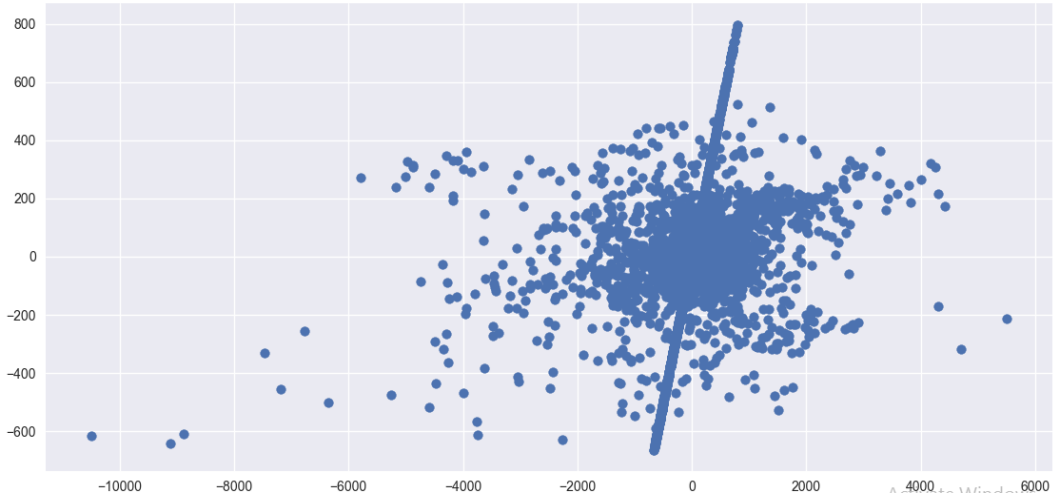

Fig 6.2.2.1: K Neighbor Regression with 50% input (scatter plot)

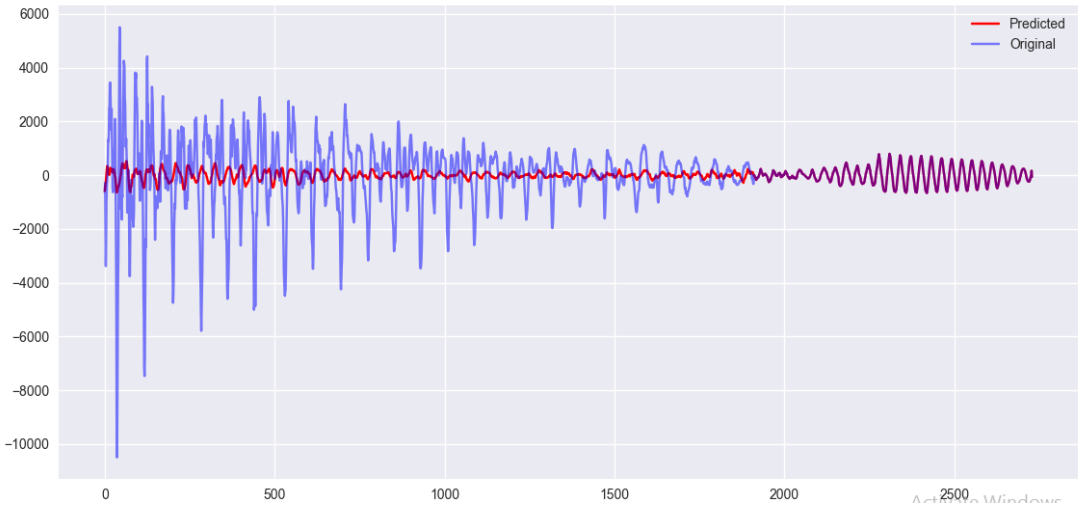

Fig 6.2.2.2: K Neighbor Regression with 50% input (heartbeat plot)

6.2.3 Decision Tree Regression

Absolute Mean regression: 333.826

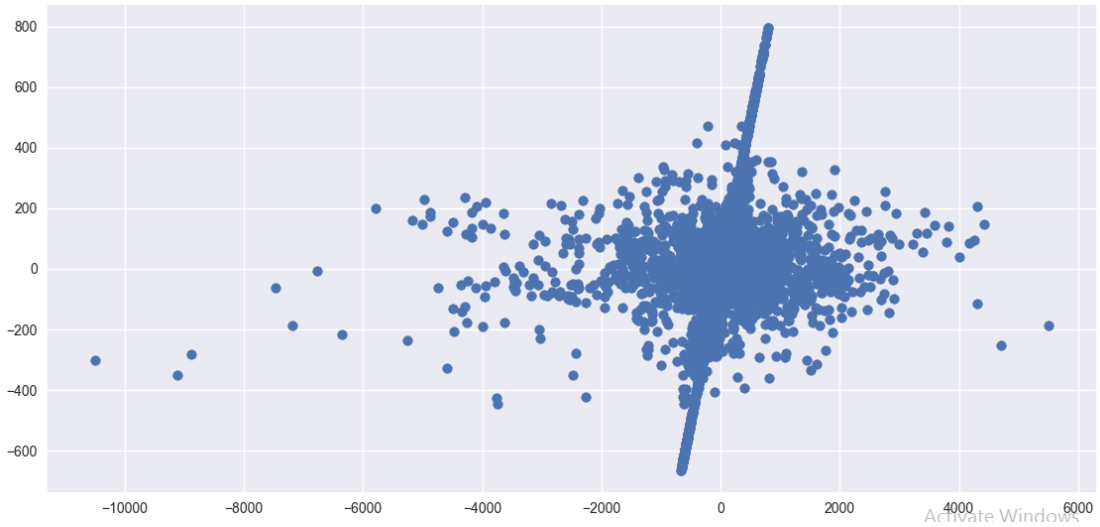

Fig 6.2.3.1: Decision Tree Regression with 50% input (scatterplot)

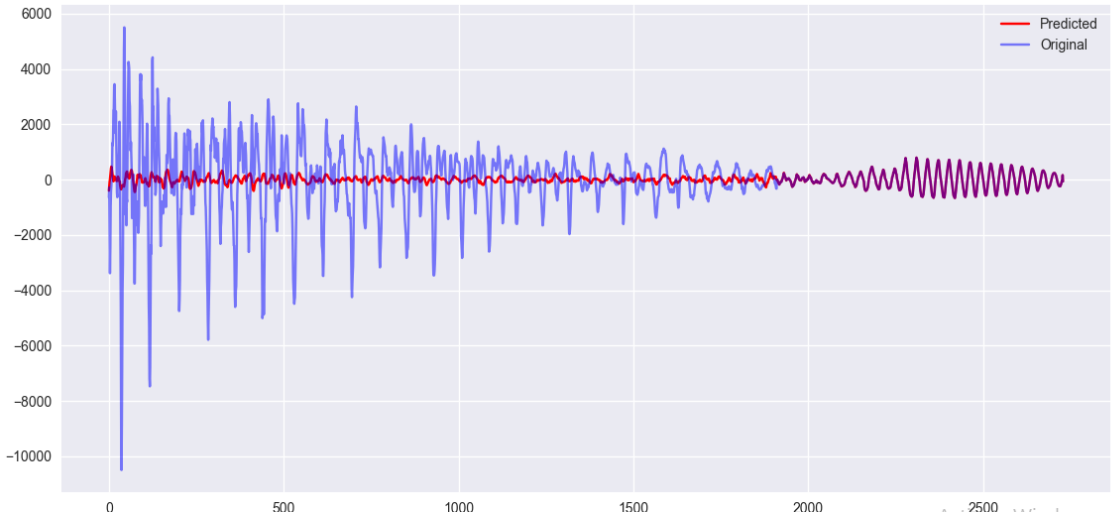

Fig 6.2.3.2: Decision Tree Regression with 50% input (heartbeat plot)

6.2.4 Random Forest Regression

Absolute Mean Error: 335.279

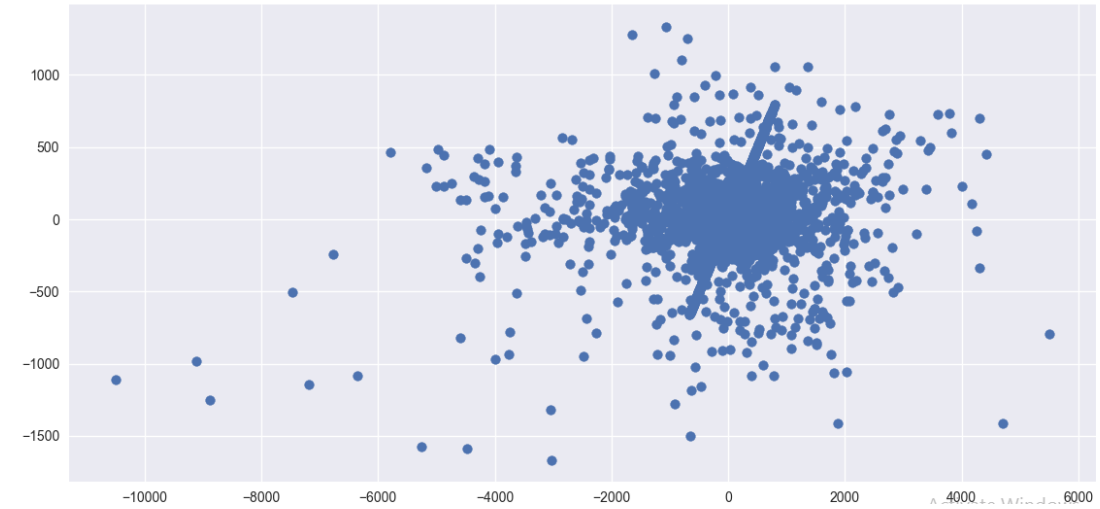

Fig 6.2.4.1: Random Forest Regression with 50% input (scatter plot)

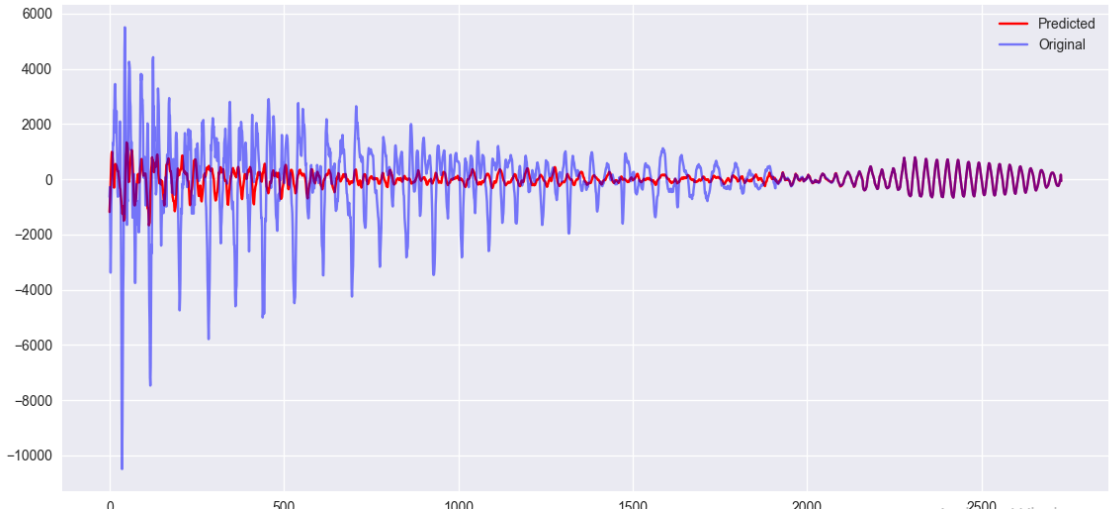

Fig 6.2.4.2: Random Forest Regression with 50% input (heartbeat plot)

6.2.5 Support Vector Regression

Absolute Mean error: 1128

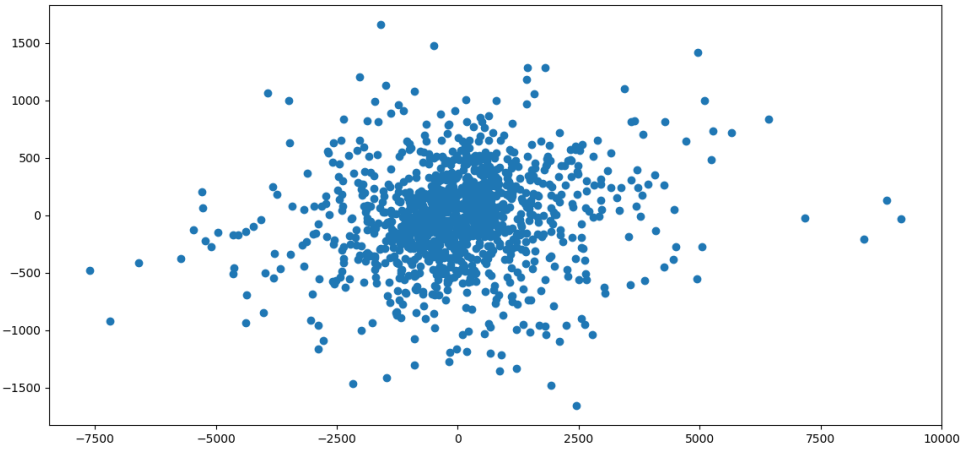

Fig 6.2.5.1: Support Vector Regression with 50% input (scatter plot)

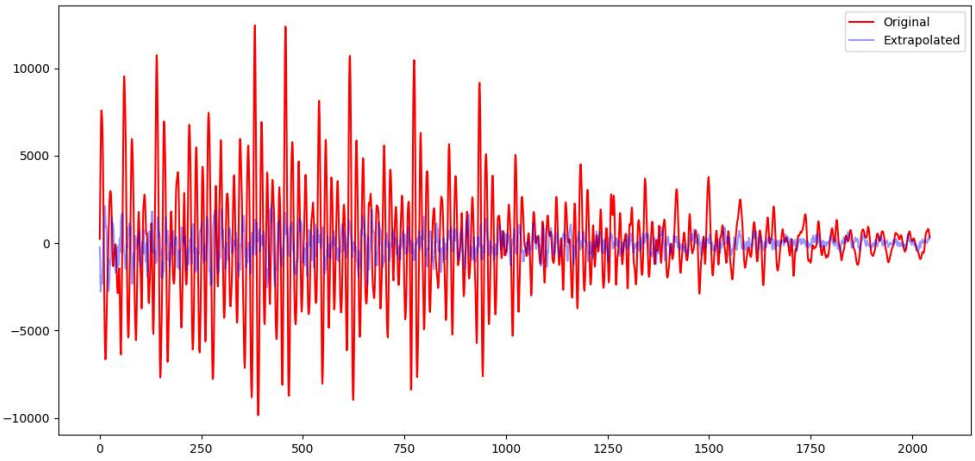

Fig 6.2.5.2: Support Vector Regression with 50% input (heartbeat plot)

CHAPTER 7

Conclusions and Future Work

7.1 Conclusions

In our project Audio Extrapolation, we have analyzed our results by using various machine learning algorithms like Linear Regression, K Neighbor Regression, Decision Tree Regression, Random Forest Regression and Support Vector Regression. Among them Random Forest Regression gives the least error with 20% input audio and K Neighbor Regression gives the least error with 50% input audio. The use of the proposed extrapolation method allows reconstruction of extremely long missing or damaged sections with none or barely audible side effects. Due to the low computational complexity, the extrapolation technique can be implemented into real time audio restoration applications.

Overall comparison (With 20% input)

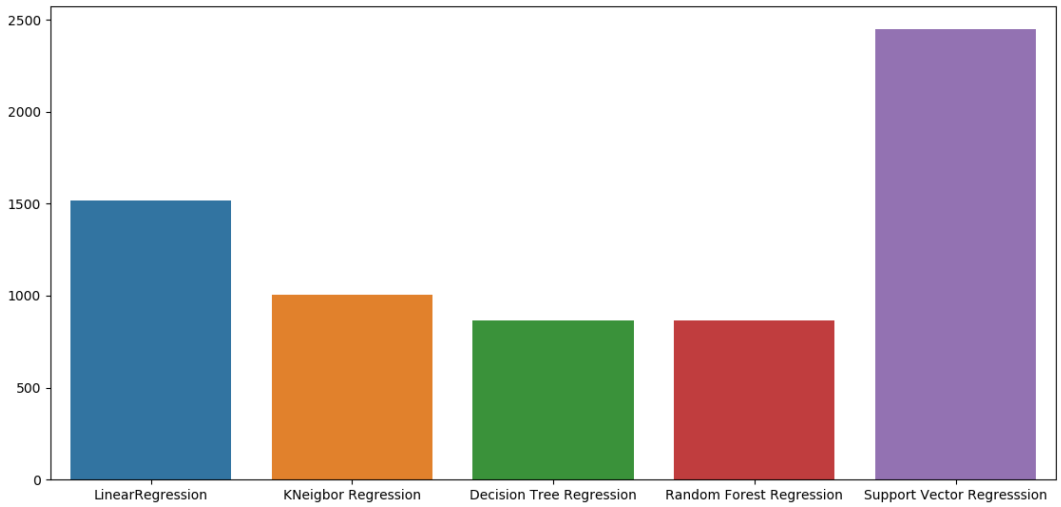

Fig 7.1.1: Overall comparison of models at 20%

It can be clearly seen that with 20% input Random Forest Regression works best followed by Decision Tree Regression and K Neighbor Regression.

Overall comparison (With 50% input)

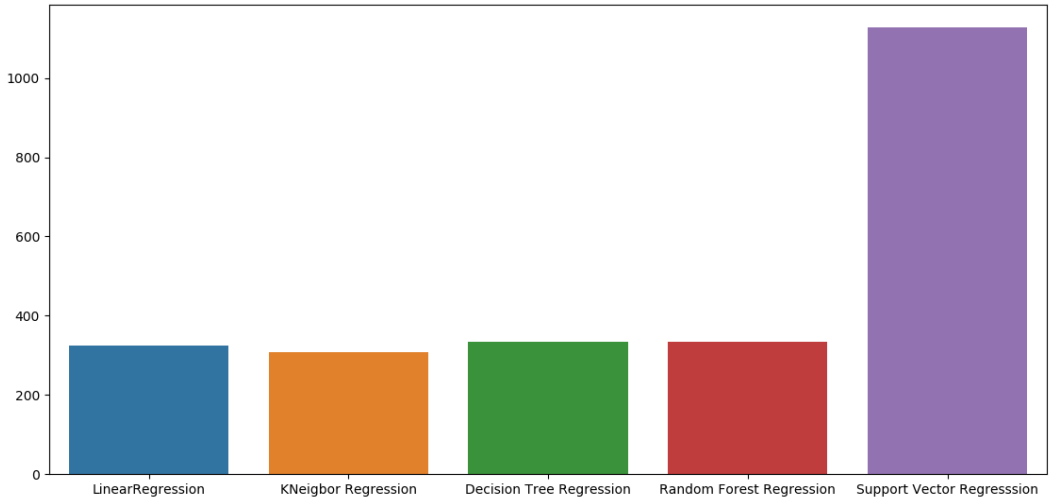

Fig 7.1.2: Overall comparison with 50% input

It can be clearly seen that with 50% input K Neighbor Regression works best followed by Random Forest Regression and Decision Tree Regression. But the error difference between all these three models is very less which concludes that as input samples increase nearly all the regression models give same results except Support vector Regression.

7.2 Future work

- This project is based on audio dataset which is specific to some particular audio and hence may not be used for extrapolation of all the audio samples. In order to take our work across all the audio samples, entire audio dictionary with words spoken by contrasting set of people should be used.
- Certain words having similar prefixes should be considered accurately for extrapolation. Like the words wonder, wonderful and wonderfully needs to be distinguished properly.
- Making this project work in the reverse direction where the suffixes are known but we need to extrapolate the start. Technically, can be referred to as reverse extrapolation.
- This project can be further extended to some other regression models and deep learning supervised means.

References

[1] “Long Interpolation of Audio signals using linear prediction in sinusoidal modelling”, Mathieu Lagrange.

[2] Bandwidth extrapolation for Audio signals by Sung-Won Yoon, David Choi.

[3] I. Kauppinen A and J. kauppinen,” A method for long extrapolation of audio signal”.

[4] “Comparison of various predictors of audio extrapolation” by Macro fink & Martin Holter.

[5] “A survey on multi-output regression” by Hanen Borchani, Gherardo Varando, Concha Bielza.