**PROFESSIONAL TRAINING REPORT**

**at**

**Sathyabama Institute of Science and Technology**

**(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering Degree in Computer Science and Engineering

By

**SAGI SANDEEP**

**REG.NO: 39110865**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

***JEPPIAAR NAGAR, RAJIV GANDHI SALA*I,**

***CHENNAI – 600119, TAMIL NADU***

**NOVEMBER 2021**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

### ***(DEEMED TO BE UNIVERSITY)***

***Accredited with Grade “A” by NAAC***

*(Established under Section 3 of UGC Act, 1956)*

*Jeppiaar nagar, Rajiv Gandhi salai*

*Chennai– 600119*

*[www.sathyabama.ac.in](http://www.sathyabama.ac.in)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to inform you that this Project Report is the bonafide work of **Sagi Sandeep (Reg.No: 39110865)** who carried out the project entitled "**AirFoil Self Noise**" under my supervision from September 2021 - November 2021.

**Internal Guide**

**Mr.Babu**

**Head of the Department**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted for Viva voce Examination held on** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Internal Examiner External Examiner**

**DECLARATION**

I, **Sagi Sandeep** hereby declare that the project report entitled "**AirFoil Self Noise based on Machine Learning**" is done by me under the guidance of **Mr. Babu** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

**DATE:**

**PLACE: Signature of the Candidate**

**ACKNOWLEDGMENT**

I am pleased to acknowledge my sincere thanks to the Board **of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr Babu,** for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

**TRAINING CERTIFICATE**



**ABSTRACT**

This project aims to predict the scaled sound pressure levels in decibels, based on the aerodynamics and acoustic related attributes. Each attribute can be regarded as a potential feature. The problem is how to predict the sound pressure level accurately based on these features. This project describes the approach using Random Forest Regression algorithm and other optimization algorithms used for better predictions. The comparative results and analysis are also provided.

The noise generated by an aircraft is an efficient and environmental matter for the aerospace industry. A vital component of the total air-frame noise is the airfoil self-noise, which is due to the interaction between an airfoil blade and the turbulence produced in its boundary layer and near wake.

This self-noise mechanism are due to boundary-layer phenomena, that is, the boundary-layer turbulence passing the trailing edge,separated-boundary-layer and stalled flow over an airfoil, vortex shedding due to laminar boundary layer instabilities, vortex shedding from blunt trailing edges, and the turbulent vortex flow existing near the tip of lifting blades.

In this project I will be demonstrating the method of approach made through the Random Forest Regression algorithm and graphs to predict better scaled sound pressure values. The behaviour of airfoils can be understood by performance optimization and using this we can make designs with minimal or reduced noise.

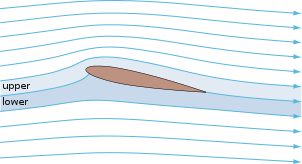
1. **INTRODUCTION**
   1. ***About Airfoil***

An airfoil is the cross-sectional shape of an object whose motion through

a gas is capable of generating significant lift, such as a wing, a snail,or the blades of propeller, rotor, or turbine.

The wings and stabilizers of fixed-wing aircraft, as well as rotor blades, are built with airfoil-shaped cross sections.Airfoils are also found in propellers,fans, compression and turbines.

A visual representation of airfoil is given below:-

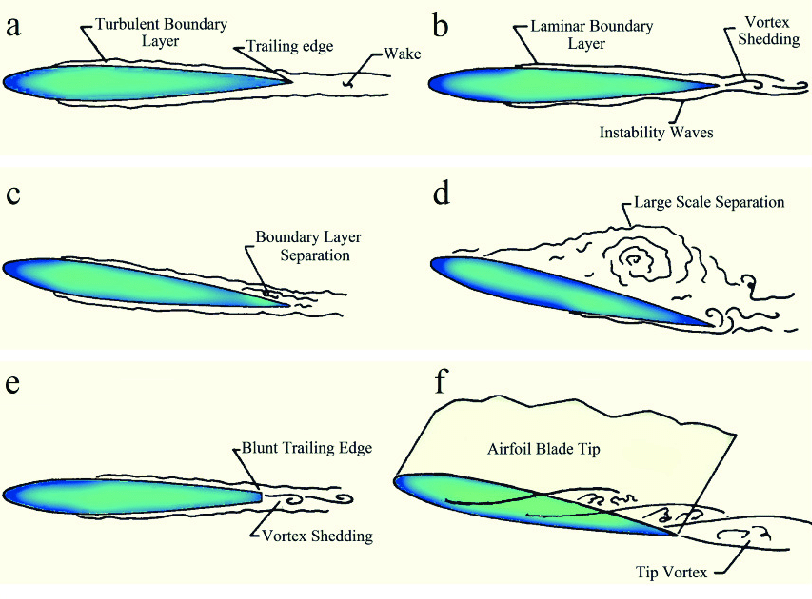


Examples of aerofoil in nature and in or on various vehicles. The dolphin flipper at bottom left obeys the same principles in a different fluid medium; the figure is shown below;-



***1.2 Airfoil self-noise Reason, Sources and Background***

The interaction between an airfoil blade and the turbulence produced in its own boundary layer and near wake is the reason behind airfoil self-noise. It is the total noise produced when an airfoil encounters smooth non-turbulent inflow. There are five mechanisms of generation of self noise as shown in Fig 1.1.



***Fig 1.1: (a) Turbulent Boundary Layer-Trailing Edge Noise; (b) Laminar Boundary Layer-Vortex-Shedding Noise; (c) Separation-Stall Noise at low angle of attack; (d) Separation-stall Noise at high angle of attack; (e) Trailing Edge Bluntness-Vortex-Shedding Noise; (f) Tip Vortex Formation Noise.***

At high Reynolds number(Rc), Turbulent Boundary Layers(TBL) develop over most of the airfoil, noise is generated as the turbulence passes over the Trailing Edge(TE).

At low Rc, largely Laminar Boundary Layers(LBL) develop, whose instabilities result in Vortex Shedding(VS) and associated noise from the TE. The various angles of attack results in noise generation by either shed turbulent velocity or causing the airfoil to radiate at low-frequency. Another noise source is VS occurring in the small separated flow region aft of a blunt TE.

***1.3 PREDICTION INVOLVED***

Different techniques have evolved over the years to estimate TE noise which uses the airfoil geometry and flow condition (e.g. Mach number, RC , etc.) as inputs to compute the Sound Pressure Level(SPL), as a measure of the noise generated.

These techniques have been classified into three categories which are named as empirical, direct and hybrid methods. The empirical methods are derived directly from the analysis of wind tunnel results. Noise estimation is done by calculating the turbulent flow characteristics and the noise generated in a single step in the direct method. The hybrid method uses a two-step approach to the problem; in the first step turbulent flow field properties are estimated, which are used as the source in the second step to estimate the sound field.

Schinkler and Amiet developed a model for helicopter rotors, which was later used to predict the noise of wind turbine blades very accurately. Other empirical methods of noise prediction were developed based on fluctuating surface pressure. "Correct estimation of turbulence properties is the key to accurate TE noise prediction."

***1.4 OVERVIEW OF THE REPORT***

This report describes the attempt made to design the model precise enough to make the prediction with greater accuracy.

The rest of the report is divided into 5 chapter. The 2nd chapter gives a clear idea of what we are going to work on and how the project is going to unfold as we go ahead. The 3rd chapter gives a brief introduction on the Random Forest algorithm and its Regression models which were used to model and predict the self noise and also describes in detail the data set used for training and testing of the model proposed. The 4th chapter details the designing aspect of the regressors along with result analysis and furthermore the chapter discusses and critically analyses the results obtained.The 5th and the final chapter concludes the study undertaken with a brief summary of whatever has taken place so far.

1. **AIM AND SCOPE OF THE PROJECT**
   1. ***AIM***

Our aim is to make a highly effective and accurate machine learning model,to predict with high level of accuracy, the higher the level of accuracy and robustness of this model will play a vital role in optimizing the noise generated by various airfoil based devices.

* 1. ***SCOPE OF INVESTIGATION***

The first task we have on hand is to choose a relabel programming environment .Then we get our hands on the data set and understand what the data set and its entries describe based on the problem statement.

Next, is to understand the category of data in the data set, because we don’t want any null values present in the data-set which will affect the prediction of the model. So we have to clean our data and filter out all the invalid and null values.We are also going import necessary libraries. Then jumping into the data prep work for the model training and testing, here for the model we have decided to use the Random Forest algorithm for the model which we hoped to produce greater accuracy for the prediction values.

Then we will analyze the results using plots and various graphs to have a better understanding of the results when compared with the actual entities.

***3.* ENVIORNMENT, MATERIALS AND METHODS, ALGORITHMS USED**

***3.1 PROGRAMMING LANGUAGE AND ENVIORNMENT TO BE USED***

We are using Python programming language thought this project .Python is a programming language used across the globe to perform data cleaning, analysis, and using different machine learning models in-order to predict the data accurately.

**Python** is the most used language for Machine Learning (which lives under the umbrella of AI). One of the main reasons Python is so popular within AI development is that it was created as a powerful data analysis tool and has always been popular within the field of big data.

We are going to use Jupyter notebook as our coding environment.The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

***3.2 DESCRIPTION OF DATA SET***

We will work on the self-noise data set that will be obtained at: <https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise>

I saved the above data set into an excel file and moving forward will be using that file in the project.

The self-noise data set used was compiled by extensive aerodynamic and acoustic testing on airfoil blade sections, conducted in an anechoic wind tunnel by NASA in 1989, referred hereinafter as the self-noise data set.

The self-noise data set consists of different size NACA0012 airfoils with a constant span to various wind tunnel speeds at varying angles of attack. The observer position remained the same in all of the experiments. The acoustic measurements were used to determine spectra for self-noise from airfoils encountering smooth airflow. This data was suitably processed to remove extraneous contributions and the results were presented as ⅓- octav spectra.

The first step is to prepare the [data set](https://www.neuraldesigner.com/learning/tutorials/data-set), which is the source of information for the approximation problem. It is composed of:

* Data source.
* Variables.
* Instances.

The self-noise dataset contains 1503 entries and consists of the following variables:

Frequency (Hz)

* Angle of Attack (deg.)
* Chord length (m)
* Free-stream velocity (ms-1)
* Suction side displacement thickness (m)
* Scaled sound pressure level, SPL1/3 (db)

The suction side displacement thickness was determined using an expression from boundary layer experimental data.

***3.3 IMPORTING NECESSARY LIBRARIES***

We will generally be working on this project using the standard libraries such as:

* Sklearn
* Pandas
* Numpy
* Matplotlib
* Seaborn

Sklearn is the root important library from which we are going to import and make use of the algorithms or the metrics or the train\_test\_split for making predictions using the model generated using thee.

Pandas is the standard library one would be dealing with while doing a machine learning project. As I have mentioned above that I will be using the data set which is stored in an excel file, we can use pandas to read the excel file.

Similarly we will use Numpy to make use of the numpy arrays, while we are calculating the errors or handling the data of some kind.

Matplotlib is one of the major necessities as we are going to use it for creating plots and to analyze our predictions based on that.

Seaborn is also a library used for visualizing the data using the plots. It was my personal opinion to choose this library out of other visualizing modules and libraries present. Moving forward there may be some other libraries that are used offhand but all serve for the same use.

We will break down more about the libraries and their usage once we start using them. I will try to describe that particular usage and its overall description too once we get into the coding part.

As we have seen that the above are the necessary libraries which we will import as full but for some we will only import the necessary modules (just to have a quick knowledge of exactly what modules we will be using).

We will import a few specific functions and algorithms from modules of Sklearn library, they are namely:

* train\_test\_split from model\_selection module of sklearn
* RandomForestRegressor from ensemble module of sklearn
* metrics module of sklearn

Model\_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.

train\_test\_split is a function in sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

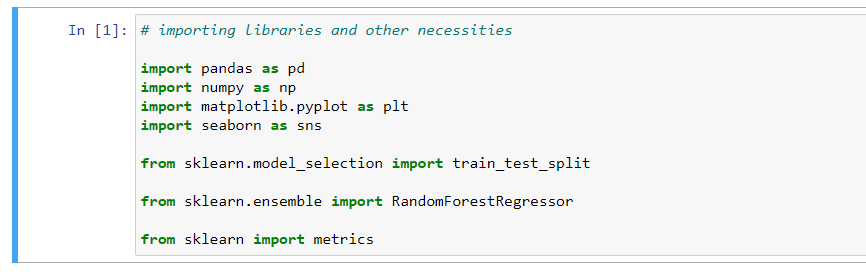
With this function you don’t need to divide the dataset manually. By default, sklearn train\_test\_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

Ensemble methods' goal is to combine the predictions of several estimators built with a given learning algorithm in-order to improve generalizability / robustness over a single estimator.

In random forests (see [RandomForestClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier) and [RandomForestRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor) classes), each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

The sklearn.metrics module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on [Classification metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics), [Multilabel ranking metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#multilabel-ranking-metrics), [Regression metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics) and [Clustering metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#clustering-metrics).

Fig 3.1 describes the code to import all the mentioned necessities.

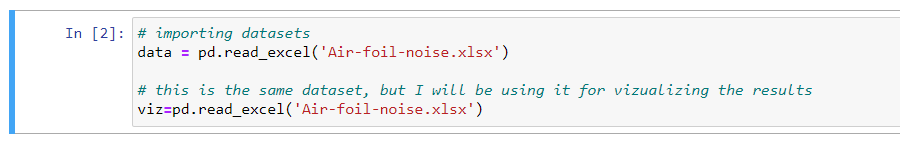


***Fig 3.1: code for importing libraries***

We will discuss more about the uses of all the libraries and modules mentioned above as we go through the coding phase

***3.4 IMPORTING THE DATA-SET FROM .xlsx FILE***

As we discussed earlier we will import the data set which was saved in excel file (.xlsx extension). We will use pandas to do it, I will use two different data-frame to import data set into. One for the data-set used for training, testing, model and predictions and the other for visualization. Fig 3.2 will show the code to import an excel file into a pandas data-frame.

***Fig 3.2: code for importing data set***

***3.5 DATA PRE-PROCESSING***

Data processing or data handling is one of the important steps needed to be taken before using the data front he given data set. Data processing is the task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative. Using Machine learning algorithms, mathematical modeling, and statistical knowledge, this entire process can be automated. The output of the process can be in any desired form like graphs, videos, charts, tables, images, and many more, depending on the task we are performing and the requirements of the machine.

The classic steps for data processing are as follows:

* Collection
* Preparation
* Input
* Processing
* Output
* Storage

Collection:

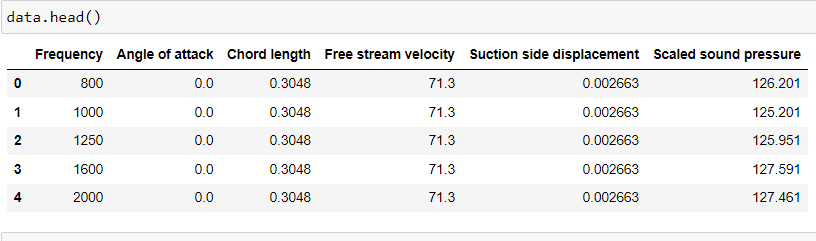
The most crucial step is to have the data of good quality and accuracy. The data we will be using is collected from UCL dataset for airfoil self-noise. Good data ensures that the results of the model are valid and can be trusted upon.

Preparation:

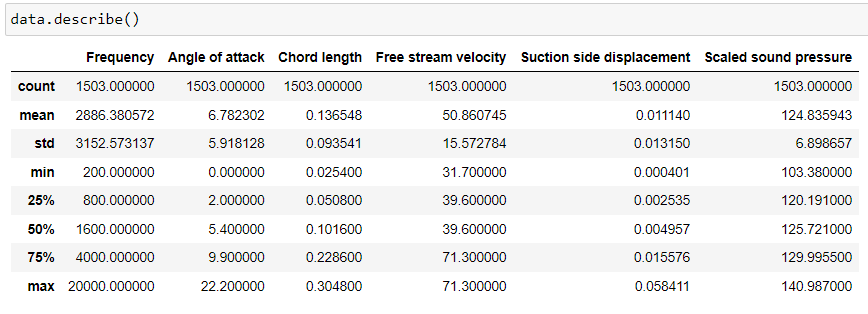
The collected data can be in a raw form which can’t be directly fed to the machine. So, this is a process of collecting data sets from different sources, analyzing the data sets and then constructing a new data set for further processing and exploration. This preparation can be performed either manually or from the automatic approach. Data can also be prepared in numeric forms also which would fasten the model’s learning.

For this preparation we have to first check for null values, for our self-noise data set we are lucky that we do not have any null values.

We can also see the data frame in a more detailed way in two different ways but he commands which is shown in fig 3.3 and 3.4, such as data.head() - which will just give us the generic top values of every column in the data set and also data.describe() - which is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.



***Fig 3.3: result of data.head()***

***Fig 3.4: result of data.describe()***

Example for preparing the data if you had null values or other irrelevant data entries in the data set, fig 3.5 and 3.6 describes the code for the data prep work for null values:

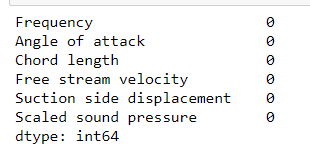
I have encountered a scenario for another data set where they had to remove the null values, or there were some other values entered in the data instead of null values like “?” and we had to replace those with null values and finally remove those null values. We can replace the continuous variables into some integer type if those variables are finite, like in the below example fig 3.4, for the race column there were finite number of variables such as ‘black’, ‘asian’, ‘other’ etc, they were replaced with finite integers 0, 1, 2, 3, etc respectively.

***Fig 3.5 & 3.6: Example code for the prep-work of data set with null values***

The above images describe the code for the example data set which will have

irrelevant entries. The “?” or the repeating values are converted into integer type values, as the repeating entries are in char and if they were in respective int type format then it will be easier to understand and handle the data.

The above was the case of an example data set, but our current data set doesn’t have any of these complexities and was very simple to handle. To check if there are null values a simple function can be used: data.isna().sum(), this will print the sum of the null values in every column, as we can see in fig 3.7, there are no null values so we can proceed with the preparation.



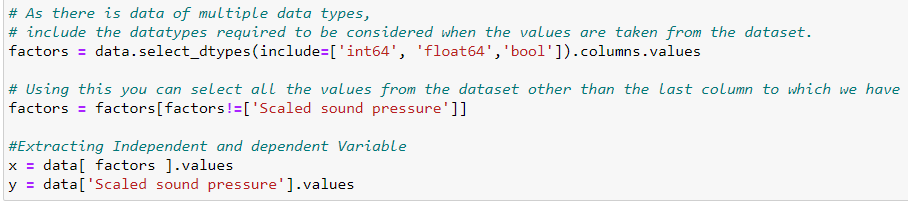
***Fig 3.7: result of the print statement “data.isna().sum()” for checking null values***

Input:

Now the prepared data can be in the form that may not be machine-readable, so to convert this data to the readable form, some conversion algorithms are needed. For this task to be executed, high computation and accuracy is needed.

For our self-noise data set we will now divide the dataframe into two parts, one with all the columns we know of the data to the other with the column of values for which we have to make the prediction. As stated in chapter 3.1 there are 6 columns in which the first five describe the means of the noise production and for the last column we have to predict the values.

So as my data set contains data of data types such as int64, float64, bool, I made the code as in fig 3.8, to obtain the column values of the respective data types I have manually given and also separate them from the last column values.

***Fig 3.8: code for splitting dataframe into two parts based on the column values***

Processing:

In this stage, results are procured by the machine in a meaningful manner which can be inferred easily by the user. Output can be in the form of reports, graphs, videos, etc. We will go through this phase in the next part.

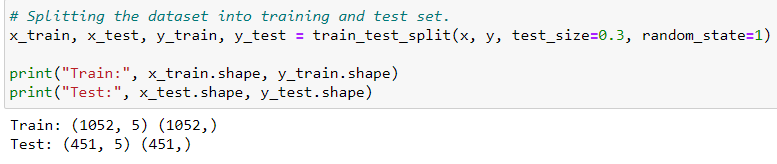
***3.6 SPLITTING DATA FOR TRAINING AND TESTING.***

As already discussed in chapter 3.3, we will use train\_test\_split that is imported from the model\_selection of sklearn library. We have already discussed about the model\_selection module and its use, now we will understand about the train\_test\_split function:

Sklearn train\_test\_split has several parameters. A basic example of syntax will look like: [ train\_test\_split(X, y, train\_size = 0.\*, test\_size = 0.\*, random\_state = \*) ].

* X, y - it is the data-set you are selecting to use.
* train\_size - this parameter sets the size of the training data set. There are three options: None - which is default, Int - which requires exact number of samples and Float - which ranges from 0.1 to 1.0.
* test\_size - this parameter specifies the size of the testing data set. The default state suits the training size. It will be set to 0.25 if the training size is set to default.
* random\_state - the default mode performs a random split using np.random. Alternatively, you can add an integer using an exact number.

As shown in fig 3.9, I have preferred to keep the train\_size to be default and the test\_size to be 0.3, so that it will split the data set into that ratio so that I can get accurate predictions and also preferred to keep the random\_state to 1. You can see the shapes of the train and test data set in the result of the fig 3.9

***Fig 3.9: code for splitting the dataset into training and testing.***

***3.7 RANDOM FOREST REGRESSION***

We will make the predictions by using the model made upon a random forest algorithm, so let’s understand about the algorithm before we move any further.

The Random Forest algorithm proposed by Brieman is a collection of tree predictions where the trees are formulated on the basis of various random features, hence the name “Random Forest”. It is an example of “ensemble learner” wherein each base learner is a Classification and Regression Tree (CART). Random vectors are generated to depict the growth of the trees; the trees are never pruned.

A random combination of features is selected at every node to perform splitting. Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size “max\_features”.

The purpose of two sources of randomness is to decrease the variance of forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yields decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

*Lets understand about the history and mathematical process that occurs while training a data set using Random Forest :*

The training process consists of generation of K number of Bootstrap sample data sets, with replacement, for K number of trees. Let the training data set be D = {(X1,Y1), (X2,Y2),...., (XN,YN)} and each feature vector is Xp = {xp1, xp2,….., xpn) where, X ∈ D ∈ Rn. For the growth of each tree with respective Bootstrap sample data used for training, Independently and Identically Distributed (IID) random set of vectors {Φ1, Φ2,...., ΦK} are generated; each tree predictor is denoted by h(X, Φ) and the random forest is the collection of such predictors h1, h2,...., hk.

Inorder to select random features and monitor error, a technique called Bootstrap Aggregating or Bagging was incorporated. The bagging method claims to increase accuracy of the Random Forest algorithm by reducing the ongoing generalization error for the ensemble trees owing to the use of random features.

This generalization error estimates are done using Out Of Bag (OOB) method. In this method, the trees which didn’t use a particular data (Xp, Yp) for training are used to predict the value of the same data (Xp, Yp) and average predicted value is calculated. Hence it follows that if a third of the data of the entire training data set is left out in each bootstrap sample data set, then the OOB estimates are based on one third of the total classifiers present in the main combination; in order to decrease the OOB error, number of combinations must be increased, and at the same time to get unbiased OOB estimates the test error must converge to a certain finite value. It has been proven that the generalization error almost certainly converges when the number of trees grown tends to infinity. The mean square generalization error for regression analysis, the main focus of this model, is given by Exy(Y - h(X))2.

*Let’s check the algorithm of how we can implement the basic working model of Random Forest.*

*Algorithm : Random Forest*

* Let the number of instances be N and the number of features be n
* The number of features at a node of the decision tree is determined to be m (m < n).
* The following steps are repeated for each decision tree:
  + A subset of training data is set with replacement that represents the N instances and the rest of the data is used to measure the error of the tree
  + The following step is repeated for each of node of this tree

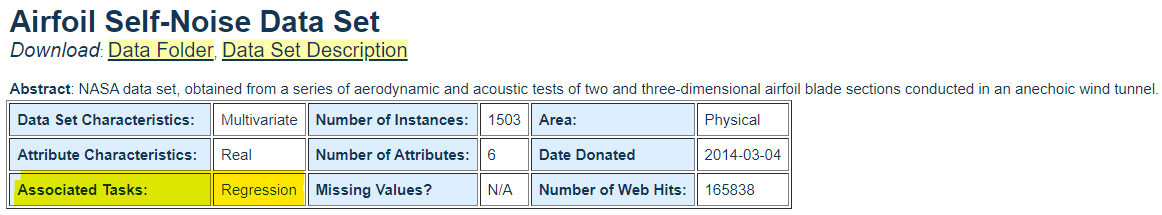
In order to determine the decision at this node and calculate the best split accordingly, m number of features is selected randomly.

* End.

*Now let’s get back to the case of our model and choose on classifier or regression for Random Forest algorithm :*

As we have discussed about the data set in chapter 3.1, the data set is given from the uni archives of airfoil self-noise, and as shown in fig 3.10, in the data set description it was said that associated tasks with the data set are to be Regression and also the data-set characteristics being multivariate, we cannot use it for prediction in a model based on classifier algorithms.

The image given below is also applicable to the data set description in chapter 3.1.



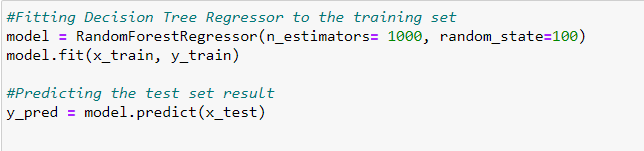
***Fig 3.10: data set description stating tasks to be Regression***

So, I will be using Random Forest Regressor algorithm based model for value predictions as we move forward.

***3.8 FITTING THE ALGORITHM AND MAKING PREDICTION***

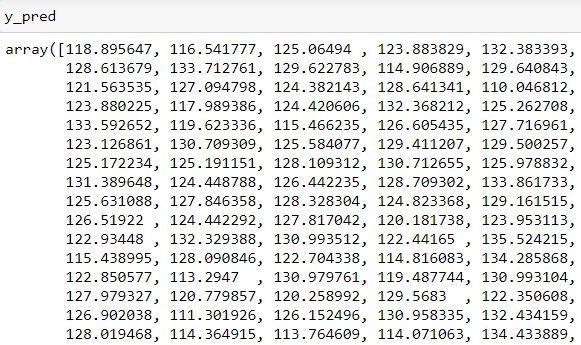
The Fig 3.11 shows how the Random Forest algorithm is fitted and to make predictions. The RandomForest Regressor has many parameters in which I will be only passing two: n\_estimators = 1000 and random\_state = 100, by default n\_estimators(the number of trees in the forest) = 100 and randomn\_state controls both the randomness of the bootstrapping of the samples used when building trees.

To fit the RandomForestRegressor algorithm to the model we can simply do as shown in fig 3.11, and similarly to make the prediction using the model, do as shown in the same fig.

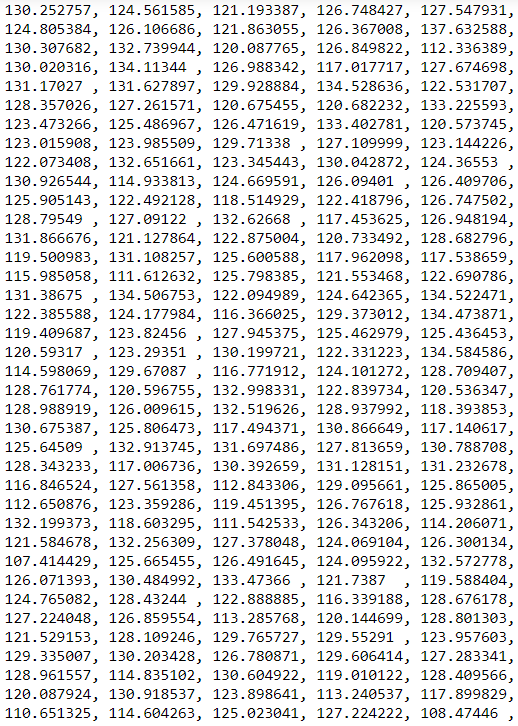


***Fig 3.11: code to demonstrate fitting the algorithm to the model.***

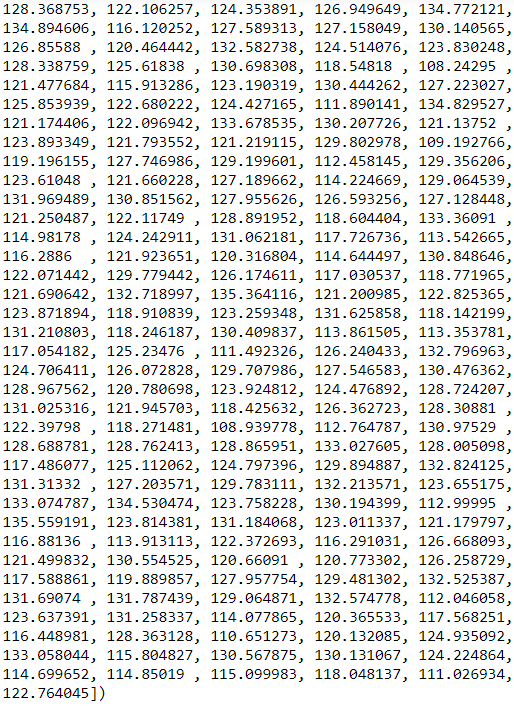
As instructed above the resulting array of the code for values prediction is shown in fig 3.12, 3.13, 3.14.



***Fig 3.12: resulting array of predictions made***



***Fig 3.13: resulting array of predictions made continuation.***



***Fig 3.14: resulting array of predictions made continuation.***

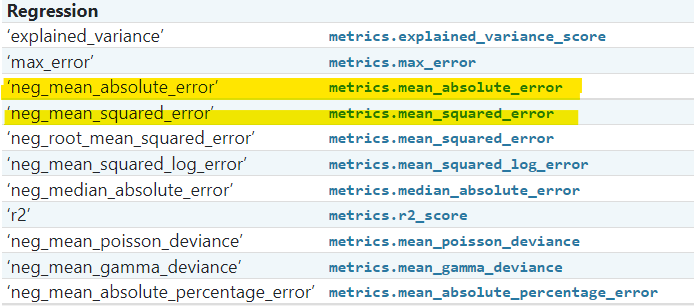
***3.9 CHECKING FOR THE ERRORS***

Errors are one of the ways to get an idea of how we have performed till now when compared to the initial values.

The error rate of RF mainly depends on the degree of correlation between any two trees and the prediction accuracy of an individual tree. The error rate decreases as the trees become increasingly uncorrelated and the prediction accuracy of each tree increases. Ease of parallelization, robustness and indifference to noise and outliers in most of the data set are the basic advantages of this method. The method is said to avoid overfitting due to its unbiased nature of execution.

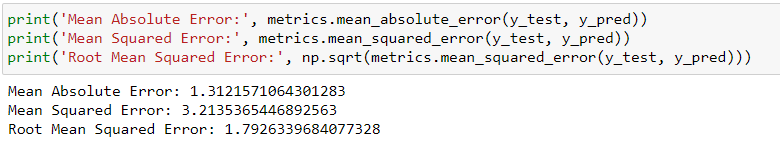
As part of the project we were asked to find out Mean Absolute Error, Mean Squared Error, Root Mean Squared Error. And these three can be found easily using the metrics module which is imported from sklearn library.

Here as shown in fig 3.15, we can see from the documentation which function to be used to find Mean Absolute Error and Mean Squared Error respectively.



***Fig 3.15: functions for errors***

But for Root Mean Squared Error we can simply use sqrt function on Mean Squared Error function using numpy. We can see the results of all three errors in fig 3.16 shown below.

***Fig 3.16: code for checking errors***

We are checking for the errors by passing the y\_test and y\_pred ad parameters in these functions, as here y\_test is the data set generated after the initial y data set (the column to which we have to predict) was gone through train\_test\_split and y\_pred is the data set that was created from the model as we wanted to predict the values for y\_test based on both x\_test and y\_test.

So when we compare both the initial data set (y\_test) and final data set obtained (y\_pred) to calculate errors, in a similar way we can calculate the accuracy of the machine.

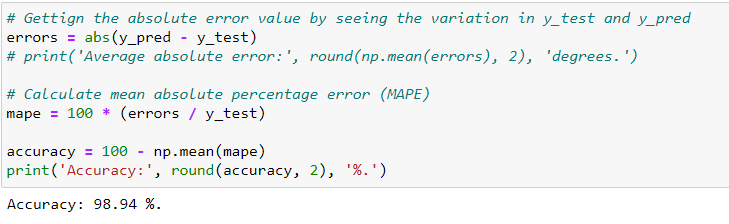
***3.9 CALCULATING THE ACCURACY***

The airfoil self-noise data set we are working on has data which has both multivariate and continuous characteristics, so I came across the “accuracy\_score” from the metrics module but it didn’t work as it has two different types of characteristics.

So I calculated the accuracy score not based on the pre-defined function from the metrics module but from calculating through errors.

The accuracy calculation process which will be explained is shown in fig 3.17, which is :

Find the absolute error by just getting the absolute value of the difference of y\_test and y\_pred, and then we have to find mean absolute percentage error (MAPE) by 100 \* (result / y\_test). Now the MAPE is the error that results in our prediction, so to attain the accuracy based on that we have to subtract the np.mean(MAPE) from 100, then we will get the accuracy percentage of the machine or of the predictions.



***Fig 3.17: code for accuracy calculation***

Here we have obtained 98.94% accuracy which is considered as a very good precision value, also we have to state that data set and data splitting(test and train data) plays a huge role in making good predictions.

Why does a model improve with more data? The best way to answer this is to think in terms of how humans learn. We increase our knowledge of the world through experiences, and the more times we practice a skill, the better we get.

A machine learning model also “learns from experience” in the sense that each time it looks at another training data point, it learns a little more about the relationships between the features and labels. Assuming that there are relationships in the data, giving the model more data will allow it to better understand how to map a set of features to a label.

For our case, as the model sees more observations of factors that cause airfoil self-noise, it better understands how to take those observations and predict the scaled sound pressure level. Practice improves human abilities and machine learning model performance alike.

**4. RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS**

***4.1 Results and discussion***

There are 4 results and they are listed below

1. ***Result for Splitting data for training and testing***

We will use train\_test\_split that is imported from the model\_selection of sklearn library. We have already discussed about the model\_selection module and its use.

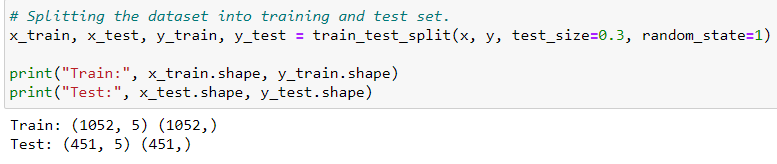
train\_test\_split is a function in sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

With this function you don’t need to divide the data set manually. By default, sklearn train\_test\_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

So, we got the results of ***train\_test\_split:***

***Train: (1052, 5) (1052,)***

***Test:(451, 5) (451,)***



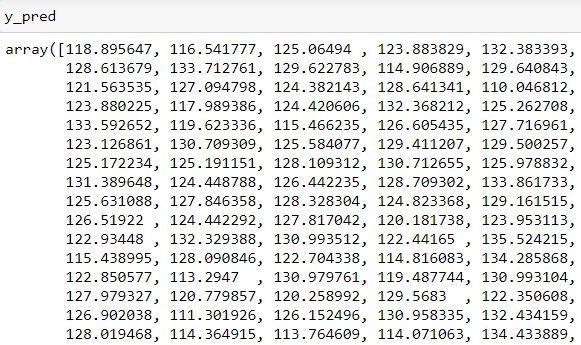
***Fig 4.1: Result of splitting data for training and testing***

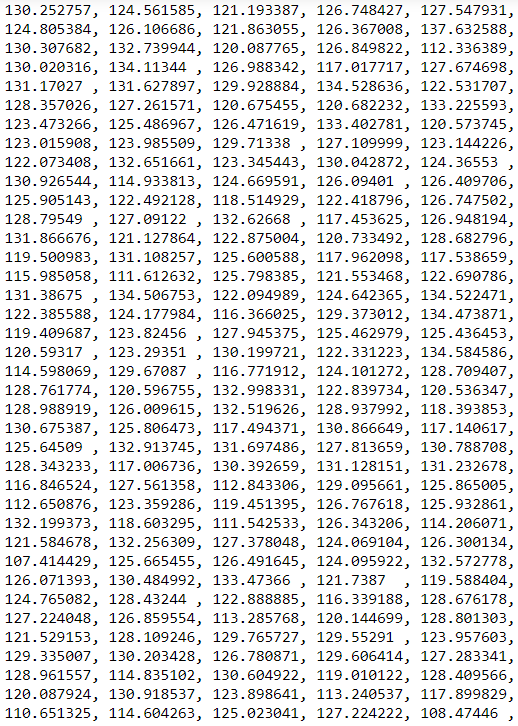
**2.Result for Fitting the algorithm and making predictions**

Random Forest algorithm is fitted and to make predictions. The RandomForest Regressor has many parameters in which I will be only passing two: n\_estimators = 1000 and random\_state = 100, by default n\_estimators(the number of trees in the forest) = 100 and random\_state controls both the randomness of the bootstrapping of the samples used when building trees.

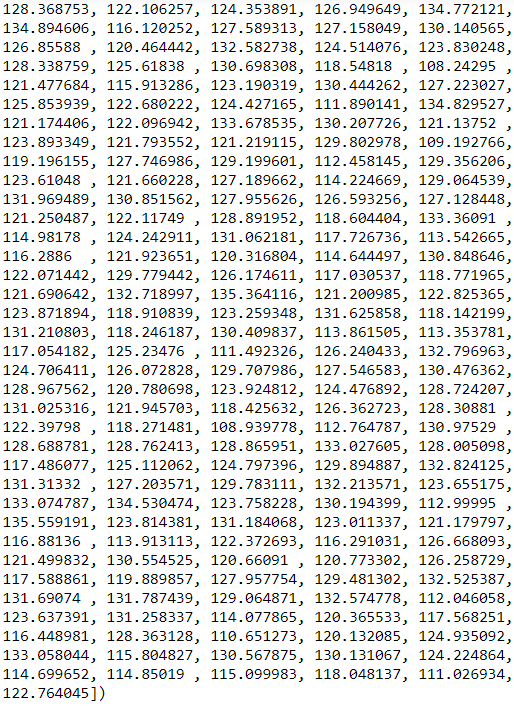
Following are the Resulting array of the code for values prediction is shown in fig 4.2 ,

4.3, 4.4.

***Fig 4.2: resulting array of the prediction made***



***Fig 4.3: resulting array of predictions made***



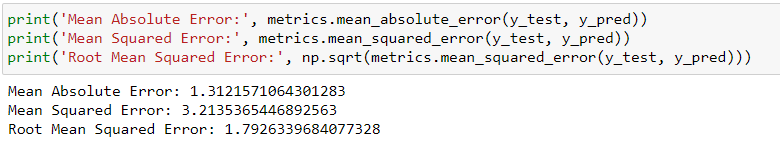
***Fig 4.4: resulting array of predictions made***

## ***3. Result for errors and other metrics***

We are checking for the errors by passing the y\_test and y\_pred ad parameters in these functions, as here y\_test is the data set generated after the initial y data set (the column to which we have to predict) was gone through train\_test\_split and y\_pred is the data set that was created from the model as we wanted to predict the values for y\_test based on both x\_test and y\_test.

So when we compare both the initial data set (y\_test) and final data set obtained (y\_pred) to calculate errors, in a similar way we can calculate the accuracy of the machine.

But for Root Mean Squared Error we can simply use square root function on Mean Squared Error function using numpy. We can see the results of all three errors in ***fig 4.5***



***fig 4.5: Results of the metrics***

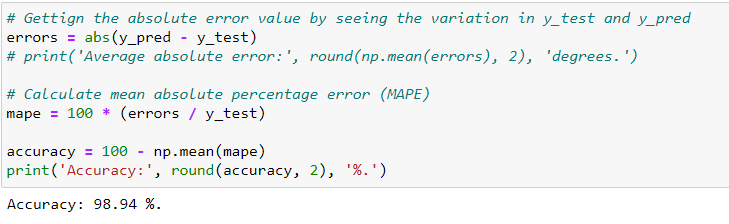
## **4.*Result for Calculating the accuracy of the prediction***

The airfoil self-noise data set we are working on has data which has both multivariate and continuous characteristics, so I came across the “accuracy\_score” from the metrics module but it didn’t work as it has two different types of characteristics.

So I calculated the accuracy score not based on the pre-defined function from the metrics module but from calculating through errors.

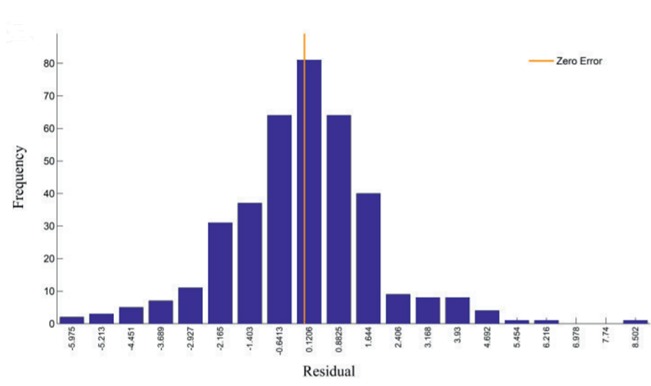
Find the absolute error by just getting the absolute value of the difference of y\_test and y\_pred, and then we have to find mean absolute percentage error (MAPE) by 100 \* (result / y\_test). Now the MAPE is the error that results in our prediction, so to attain the accuracy based on that we have to subtract the np.mean(MAPE) from 100, then we will get the accuracy percentage of the machine or of the predictions.

Here we have obtained 98.94% accuracy which is considered as a very good precision value, also we have to state that data set and data splitting(test and train data) plays a huge role in making good predictions.

So , we got the result for: ***Accuracy as 98.94%.***

***Fig 4.6: Result of the accuracy***

***4.2 Performance and Analysis***

******

**To understand more about the system design of the model, the figure shown is a Histogram of error distribution using Random Forest Prediction model on the test data set.**

***5.*SUMMARY AND CONCLUSIONS**

NASA data set, obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel.

An overall prediction method has been developed for the self-generated noise of an airfoil blade encountering smooth flow. Prediction methods for individual self-noise mechanisms are semi empirical and are based on previous theoretical studies and the most comprehensive self-noise data set available. The specially processed data set, most of which is newly presented in this report, is from a series of aerodynamic and acoustic tests of two- and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. Five self-noise mechanisms due to specific boundary-layer phenomena have been identified and modeled: boundary-layer turbulence passing the trailing edge, separated-boundary-layer and stalled-airfoil flow, vortex shedding due to laminar-boundary-layer instabilities, vortex shedding from blunt trailing edges, and the turbulent vortex flow existing near the tips of lifting blades.

We predicted the upcoming value and by using the Random-forest Regression method and successfully got the accuracy 98.94% for next value.So,by doing this we can resolve or diminish the airfoil self noise produced by the aerofoil.

By this method of approach made through the Random Forest algorithm to predict scaled sound pressure values with better accuracy. Performance Optimization can be applied to understand the behaviour of airfoils and make designs with reduced noise.

**REFERENCES**

* NASA.gov:

https://ntrs.nasa.gov/api/citations/19890016302/downloads/19890016302.pdf

[19890016302.pdf (nasa.gov)](https://ntrs.nasa.gov/api/citations/19890016302/downloads/19890016302.pdf)

* ResearchGate:

[(PDF) Random Forest and Stochastic Gradient Tree Boosting Based Approach for the Prediction of Airfoil Self-noise (researchgate.net)](https://www.researchgate.net/publication/275366771_Random_Forest_and_Stochastic_Gradient_Tree_Boosting_Based_Approach_for_the_Prediction_of_Airfoil_Self-noise)

* Neuraldesigner:

[Airfoil self-noise prediction machine learning example (neuraldesigner.com)](https://www.neuraldesigner.com/learning/examples/airfoil-self-noise-prediction)

**Relevant papers:**

* T.F.Brooks, D.S.Pope, and A.M.Marcolini.Airfoil self-noise and prediction.

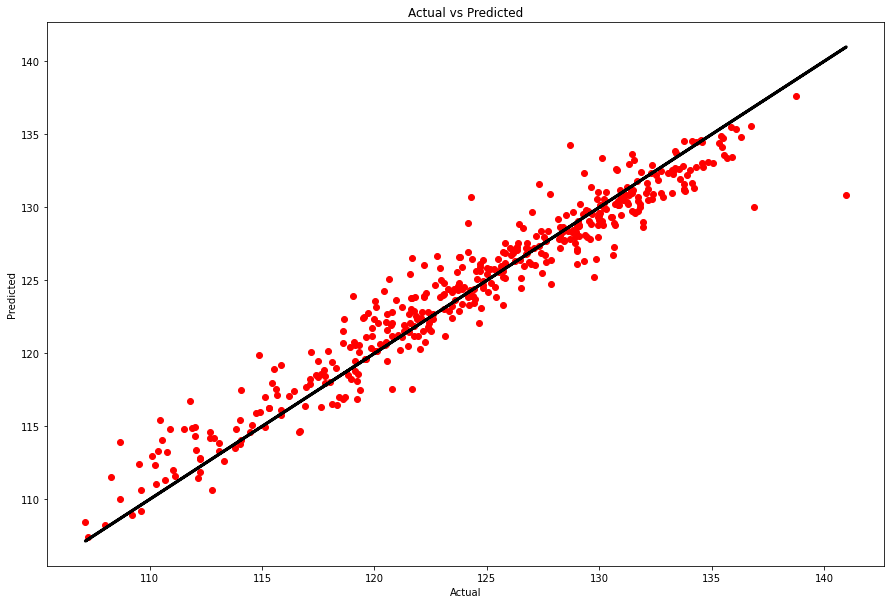
Technical report, NASA RP-1218, July 1989.

**APPENDIX**

**SCREENSHOTS:**

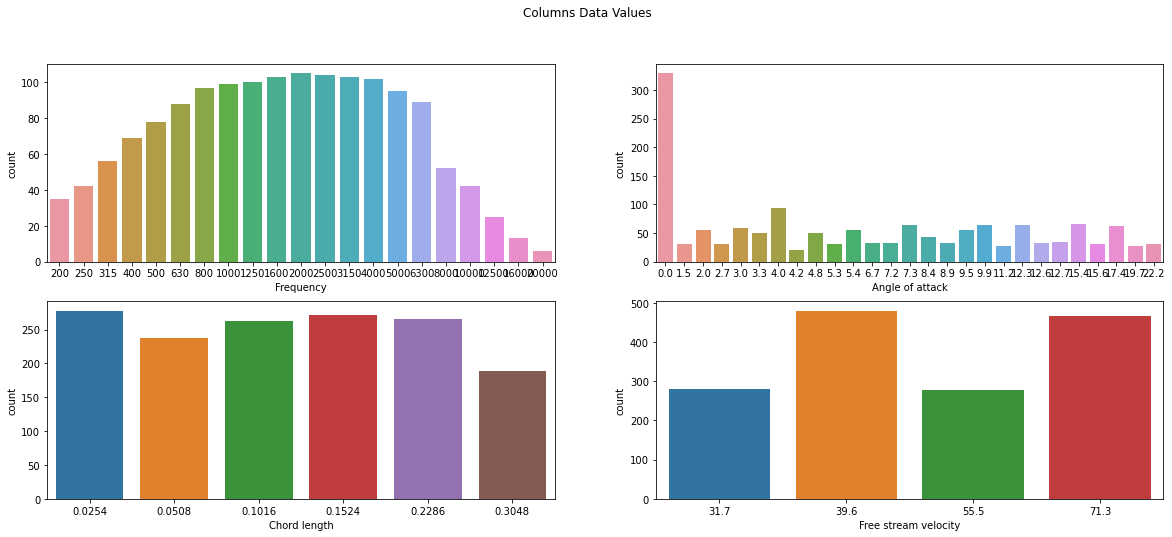
### ***VISUALISATION GRAPHS***

### **1.visualization using scatter plot to check the accuracy between the actual and predicted values**

******

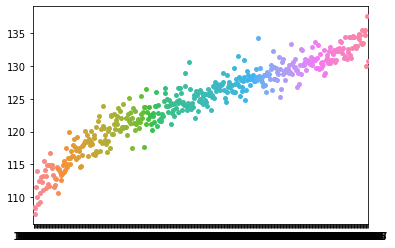
***Fig 4.7: scatter plot***

### **2.Using count plot to visualize the column's values**

****

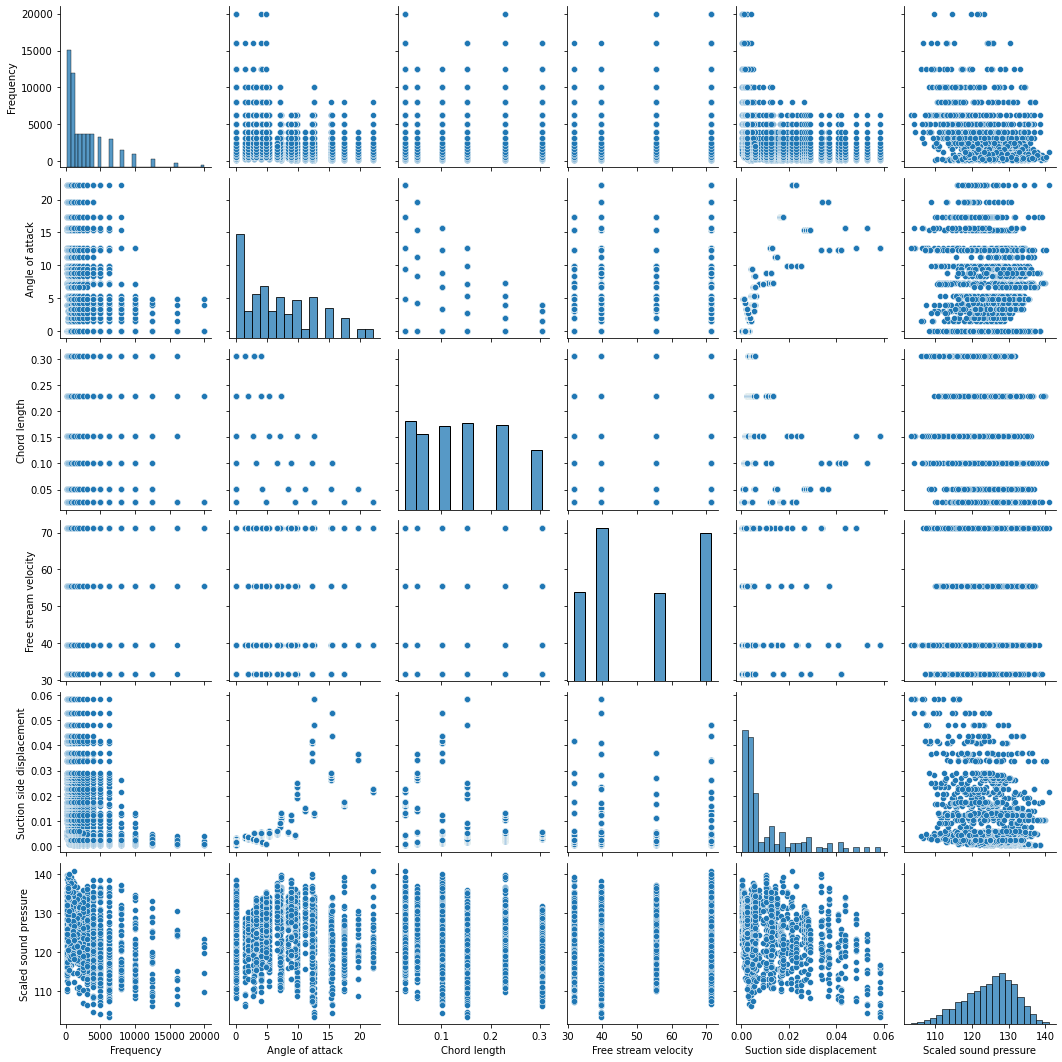
***Fig 4.8:count plot***

### **3.Using Strip plot to visualize y\_test and y\_pred as Categorical scatter plots.**



***Fig 4.9:strip plot***

### **4.Using Pair plot to finally display the relationship of data between each column and row in the data set**



***Fig 4.10:Pair plot***

***SOURCE CODE***



