 A blue logo with a sun and text

Description automatically generated

***NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY (NUST)***

***SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING (SMME)***

**FUNDAMENTALS OF PROGRAMMING II (CS – 116)**

**MID – SEMESTER PROJECT**

**“NASA AIRFOIL SELF-NOISE PREDICTION”**

**PROGRAM:** BE AEROSPACE **SEMESTER:** 3RD

**SESSION:** FALL 2024 **COURSE CODE:** CS – 116

***NAME: Ateeq-ur-Rehman***

***CMS ID: 465502***

**TABLE OF CONTENTS**

[INTRODUCTION 3](#_Toc181200040)

[THEORY 3](#_Toc181200041)

[Self-Noise 3](#_Toc181200042)

[Aeroacoustics 4](#_Toc181200043)

[Scaled Sound Pressure Level (SSPL) 5](#_Toc181200044)

[Boundary Layer 6](#_Toc181200045)

[Angle of Attack 6](#_Toc181200046)

[Bagging Regressor & Random Forest 7](#_Toc181200047)

[DATASET OVERVIEW 8](#_Toc181200048)

[PHYSICAL SIGNIFICANCE OF PARAMETERS 8](#_Toc181200049)

[Frequency (f) 8](#_Toc181200050)

[Angle of Attack (α) 8](#_Toc181200051)

[Chord Length (c) 8](#_Toc181200052)

[Free-stream Velocity (U∞) 8](#_Toc181200053)

[Boundary Layer Thickness (Δ) 9](#_Toc181200054)

[MACHINE LEARNING MODEL ANALYSIS 10](#_Toc181200055)

[Residual Analysis 10](#_Toc181200056)

[Actual vs. Predicted SSPL Analysis 11](#_Toc181200057)

[New Predictions Analysis 12](#_Toc181200058)

[ENGINEERING AND PHYSICS IMPLICATIONS/APPLICATIONS 13](#_Toc181200059)

[CONCLUSION 13](#_Toc181200060)

[PYTHON CODE 14](#_Toc181200061)

[Code Explanation 16](#_Toc181200062)

[REFERENCES 20](#_Toc181200063)

**TABLE OF FIGURES**

[Figure 1: NACA 0012 Airfoil 3](#_Toc181201213)

[Figure 2: Airfoil Self-Noise 4](#_Toc181201214)

[Figure 3: Boundary Layers 6](#_Toc181201215)

[Figure 4: Angle of Attack & Chord Line 7](#_Toc181201216)

[Figure 5: Feature Importances 9](#_Toc181201217)

[Figure 6: Distribution of Residuals 10](#_Toc181201218)

[Figure 7: Residuals vs Predicted SSPL 11](#_Toc181201219)

[Figure 8: Actual vs Predicted SSPL 11](#_Toc181201220)

[Figure 9: Actual vs New Predictions 12](#_Toc181201221)

## **INTRODUCTION**

This project (report) presents an analysis of the NASA (National Aeronautics and Space Administration, formerly NACA – National Advisory Committee for Aeronautics) Airfoil Self-Noise Dataset, and the corresponding machine learning model developed to predict the Scaled Sound Pressure Level (SSPL) of airfoils. The dataset, obtained from aerodynamic and acoustic tests conducted in an anechoic wind tunnel, offers valuable insights into the noise generation characteristics of NACA airfoils, specifically NACA 0012 airfoil, under various conditions. The [code](#_PYTHON_CODE) (written in Python), along with its [explanation](#_Code_Explanation) is given at the end of the report (just before the [References](#_REFERENCES)).

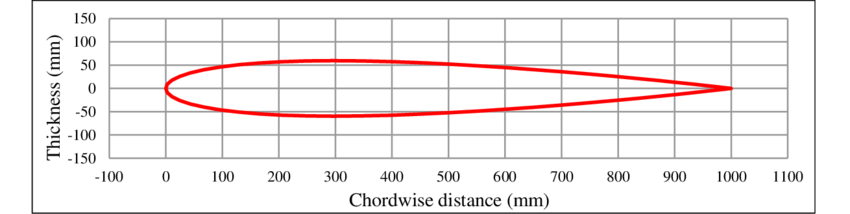


Figure : NACA 0012 Airfoil

## **THEORY**

The project delves into the field of aeroacoustics, which is an integral part of the whole aerospace/aeronautical engineering and industry, since it is related to noise and vibrations of aero vehicles which have an important effect on overall flight performance, stability and dynamics. Some terms which form the basis of this project and are directly and/or indirectly related and/or referenced to, are explained in this section.

### **Self-Noise**

Airfoil self-noise refers to the noise generated by the interaction of airflow with the surface and edges of an airfoil, such as an airplane wing or turbine blade, without external noise sources. This noise is a critical concern in aerospace and wind turbine engineering as it contributes significantly to overall noise pollution and affects the performance of aerodynamic devices.

The self-noise of an airfoil is generated by several mechanisms, each related to various aspects of airflow and boundary layer behavior:

* Boundary-Layer Turbulence Passing the Trailing Edge: Turbulent eddies interact with the trailing edge, producing sound.
* Separated-Boundary-Layer and Stalled Flow: When airflow separates from the surface of the airfoil, it forms a separated boundary layer, leading to noise from unsteady flow and vortex shedding.
* Vortex Shedding Due to Laminar Boundary Layer Instabilities: Instabilities in the laminar boundary layer cause vortices to shed, generating noise.
* Vortex Shedding from Blunt Trailing Edges: More frequent vortex shedding from blunt trailing edges increases noise.
* Turbulent Vortex Flow Near the Tip of Lifting Blades: Complex interactions of vortices near blade tips generate noise.

Several factors influence the production and magnitude of airfoil self-noise. Some of them are:

* Airfoil Shape and Thickness: Thicker airfoils and sharp trailing edges tend to generate more noise. The shape, camber, and edge design significantly affect noise levels.
* Angle of Attack: Higher angles of attack can increase turbulence and flow separation, thereby elevating noise levels.
* Reynolds Number and Mach Number: These dimensionless numbers characterize the flow conditions, influencing the transition from laminar to turbulent flow and affecting compressibility effects, respectively.
* Surface Roughness: Surface imperfections or roughness increase boundary layer turbulence, leading to higher noise levels.
* Flow Velocity: Increased speed intensifies turbulence and interactions within the boundary layer, raising noise levels.
* Wingtip Shape and Size: Variations in the shape and size of the wingtips affect the formation of tip vortices and subsequent noise generation.
* Flow Turbulence Intensity: The level of turbulence in the airflow contributes to the intensity of noise generated.

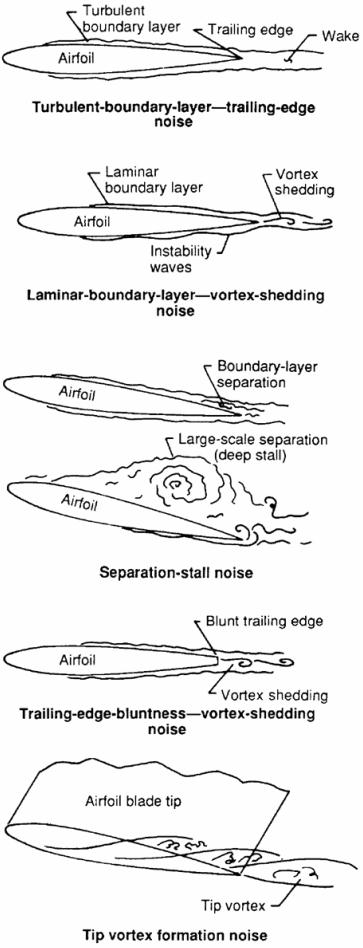


Figure 2: Airfoil Self-Noise

### **Aeroacoustics**

Aeroacoustics is a multidisciplinary field that examines the sound generated by fluid flow, integrating principles from aerodynamics, acoustics, and flight performance. This explores noise sources, propagation, and reduction techniques, significantly impacting aircraft design, efficiency, and passenger comfort. The scope of aeroacoustics includes the generation of sound by turbulent flows, aerodynamic noise from airfoils, wings, and engines, the propagation of sound through fluids, acoustic-flow interactions, and noise control and reduction methods.

* Relation with Aerodynamics:

Aeroacoustics and aerodynamics are inherently connected, as the noise generated by airflow is linked to aerodynamic forces and characteristics. Turbulent flows, flow separation, vortex shedding, and shear layers generate sound, affecting aerodynamic performance by influencing lift and drag. The shape of the airfoil, its angle of attack, and the Reynolds number are crucial factors in determining noise levels. Optimizing aerodynamic shapes helps control noise and improve performance, as turbulence, vortices, and flow separation increase drag and reduce efficiency.

* Relation with Flight Performance and Stability:

Noise generated by aerodynamic phenomena can significantly impact flight performance and stability. Aeroacoustic noise affects aircraft performance by influencing fuel efficiency and range. It can also impact flight stability, particularly through buffeting, which disrupts stable flight. Noise generation influences aircraft design, requiring considerations for shape and material to reduce noise. Additionally, noise reduction enhances passenger comfort and safety, contributing to a better flight experience. Managing aeroacoustic effects during critical flight phases, such as takeoff and landing, is crucial for safety and maneuverability.

* Connection between all three:

Aeroacoustics connects aerodynamics and flight performance through the study of noise generated by turbulent flows, aerodynamic forces, and moments. Airfoil design and shape optimization, boundary layer behavior, turbulence, flow separation, and vortex shedding are critical areas where aeroacoustics intersects with aerodynamic performance. Improved understanding of these factors leads to better noise management and enhanced flight efficiency.

The study of aeroacoustics profoundly impacts aircraft design and operations. Noise reduction enhances passenger comfort and safety, while improved aerodynamic efficiency leads to better fuel economy. Enhanced flight stability and control, reduced structural vibrations and fatigue, and optimized aircraft designs for quieter operation, are all benefits derived from a thorough understanding of aeroacoustics. By focusing on noise control and reduction, engineers can develop aircraft that are not only efficient but also environmentally friendly and comfortable for passengers.

To fully grasp aeroacoustics, it is essential to understand several core concepts, including Sound Pressure Level (SPL), frequency spectrum, directivity, acoustic intensity, and sound absorption. These concepts help in analyzing and controlling noise generation and propagation, leading to quieter and more efficient aircraft designs.

### **Scaled Sound Pressure Level (SSPL)**

Scaled Sound Pressure Level (SSPL) is a standardized measure used in aeroacoustics to compare the sound pressure levels of different airfoils or aerodynamic surfaces under controlled conditions. SSPL is essentially a normalized version of the Sound Pressure Level (SPL), which is a logarithmic measure of the pressure of sound waves relative to a reference pressure, typically the threshold of human hearing (20 µPa).

* Sound Pressure Level (SPL): SPL is a metric used to quantify the loudness of sound, determined by the pressure variations of sound waves in the air. Measured in decibels (dB), SPL uses a logarithmic scale where a 10 dB increase represents a tenfold increase in sound pressure.
* Scaling SPL: SSPL involves adjusting the measured SPL values to a common reference condition. This normalization process enables a fair and consistent comparison of the acoustic performance of different airfoils. The standard reference typically includes specific parameters such as frequency range (1 Hz to 40 kHz), distance from the airfoil (1 meter), Mach number (M = 0.2), and Reynolds number (Re = 1.6 × 106).

SSPL is particularly valuable in aeroacoustic research and engineering for evaluating and comparing the noise generated by different airfoil designs. By normalizing the sound pressure data, SSPL allows engineers to assess which airfoil designs produce less noise and are more suitable for applications where noise reduction is critical, such as in aviation, automotive, and wind energy sectors.

SSPL is dependent on several factors such as:

* Airfoil Shape and Camber: Different shapes and camber profiles affect noise levels.
* Angle of Attack: Higher angles of attack can increase turbulence and noise.
* Mach Number and Reynolds Number: Variations in these parameters influence the noise generated.
* Trailing Edge Geometry: The design of the trailing edge affects vortex shedding and noise.
* Surface Roughness: Rough surfaces increase turbulence and noise.
* Boundary Layer Turbulence: The level of turbulence in the boundary layer impacts noise generation.

SPL can be calculated using the following formula:

Where:

* = root-mean-square sound pressure
* = reference sound pressure (typically 20µPa in air)

### **Boundary Layer**

A boundary layer is a thin region of fluid, such as air or water, adjacent to a solid surface where the effects of viscosity are significant. This layer plays a crucial role in fluid dynamics as it significantly influences the flow characteristics near the surface. It is defined as the region of fluid flow close to a solid boundary where the velocity gradient is significant due to the effects of viscosity. Within this layer, the flow is influenced by friction between the fluid and the surface, creating a velocity profile that ranges from zero at the boundary (due to the no-slip condition) to the free stream velocity away from the surface. There are 3 (three) main types of boundary layers:

1. Laminar
2. Turbulent
3. Transitional

The thickness of the boundary layer increases with distance from the leading edge of the surface. It is typically characterized by the point where the velocity reaches 99% of the free stream velocity. Boundary layer thickness depends on factors such as fluid viscosity, flow velocity, and surface roughness. It is affected by the following factors:

* Reynolds number (Re)
* Surface Roughness
* Flow speed and direction
* Pressure gradient

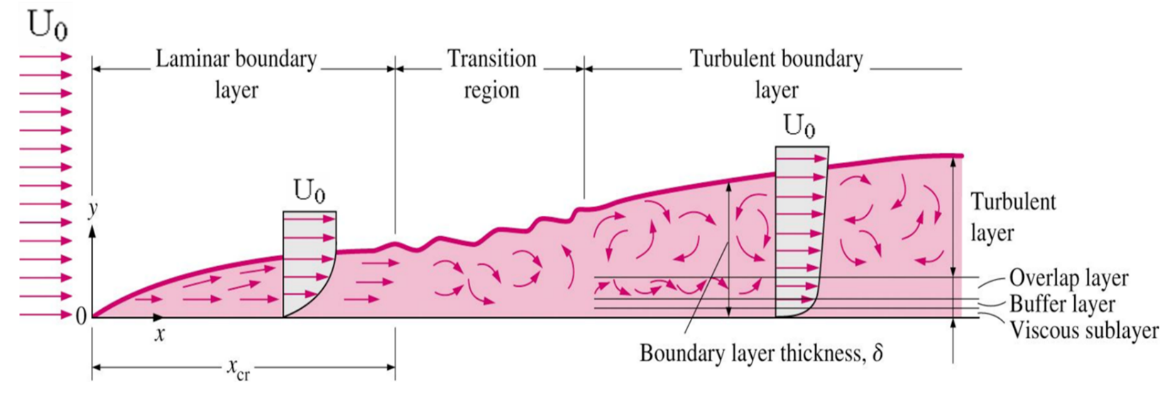


Figure 3: Boundary Layers

### **Angle of Attack**

Angle of Attack (AoA) is a fundamental concept in aerodynamics, defining the angle between the chord line of an airfoil (such as a wing or a blade) and the direction of the oncoming airflow (relative wind). It is a critical parameter that influences the aerodynamic forces acting on an airfoil, particularly lift and drag. Mathematically, it is expressed as:

Where:

* α is the Angle of Attack.
* θ is the pitch angle (orientation of the airfoil relative to a fixed reference).
* β is the angle of the relative wind to the horizontal.

The Angle of Attack is crucial because it directly affects the lift and drag forces on an airfoil. As the Angle of Attack increases, the lift also increases, up to a certain point known as the critical angle. Beyond this critical angle, the airfoil experiences a stall, where lift dramatically decreases, and drag increases significantly.

The AoA is also affected by some parameters, which include:

* Airfoil Shape
* Flow Conditions
* Aircraft Attitude
* Control Inputs

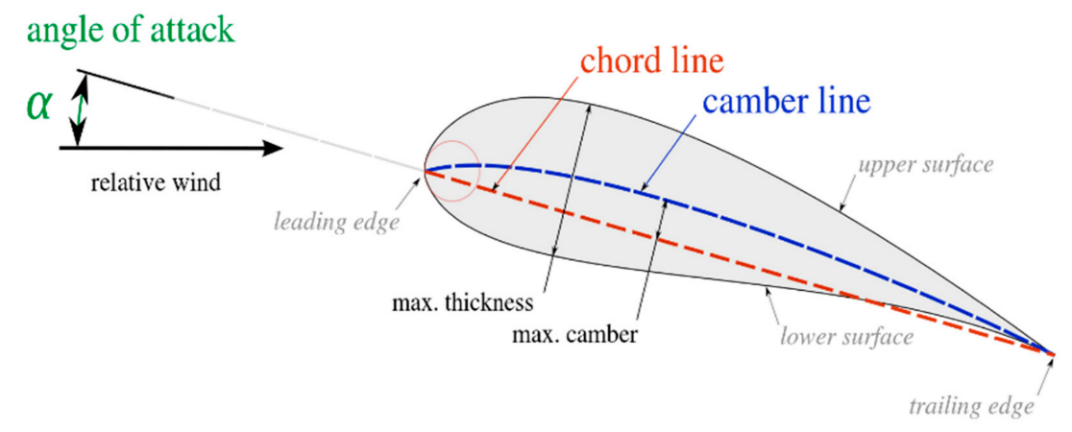


Figure 4: Angle of Attack & Chord Line

### **Bagging Regressor & Random Forest**

Bagging Regressor is an ensemble learning technique aimed at enhancing the accuracy and robustness of regression models. "Bagging," short for Bootstrap Aggregating, involves training multiple instances of the same model on different subsets of the training data created through bootstrap sampling (random sampling with replacement). The primary goal of bagging is to reduce variance and prevent overfitting by averaging the predictions of these multiple models.

Random Forest, an extension of the bagging method specifically designed for decision trees, adds an extra layer of randomness to further improve performance. In Random Forest, not only are bootstrap samples used, but each node in the decision trees considers a random subset of features when splitting. This additional randomness helps in reducing correlation among individual trees, leading to more robust and accurate predictions.

The working mechanism of both of these are as follows:

Bagging Regressor:

* Bootstrap Sampling: Generate multiple random subsets of the original dataset with replacement.
* Training Multiple Models: Train an individual regression model (base estimator) on each bootstrap sample.
* Aggregation: Aggregate the predictions from all models, typically by averaging, to obtain the final prediction.

Random Forest:

* Bootstrap Sampling: Create multiple bootstrap samples from the dataset.
* Random Feature Selection: At each node, a random subset of features is selected for splitting the data.
* Training Multiple Decision Trees: Train decision trees on each bootstrap sample.
* Aggregation: Combine the predictions of all trees by averaging (regression) or voting (classification).

## **DATASET OVERVIEW**

The NASA Airfoil Self-Noise Dataset comprises measurements from tests performed on NACA 0012 airfoils of different sizes, subjected to various wind tunnel speeds and angles of attack. The dataset includes the following parameters:

1. f: Frequency (Hz)
2. α (alpha): Angle of attack (degrees)
3. c: Chord length (meters)
4. U∞ (U\_infinity)**:** Free-stream velocity (m/s)
5. Δ (delta): Boundary layer thickness (meters)
6. SSPL: Scaled Sound Pressure Level (dB)

These parameters capture the key factors influencing airfoil self-noise, allowing for a comprehensive study of the aeroacoustic phenomena involved.

## **PHYSICAL SIGNIFICANCE OF PARAMETERS**

### **Frequency (f)**

The frequency parameter is crucial in understanding the spectral characteristics of airfoil noise. Different noise generation mechanisms dominate at various frequency ranges:

* Low frequencies: Associated with large-scale vortex shedding and flow separation
* Mid frequencies: Often related to turbulent boundary layer noise
* High frequencies: Typically dominated by small-scale turbulence and trailing edge noise

### **Angle of Attack (α)**

The angle of attack significantly influences the airfoil's lift generation and flow characteristics. As the angle of attack increases:

* The pressure difference between the upper and lower surfaces of the airfoil grows
* The risk of flow separation on the suction side increases
* The noise generation mechanisms may shift, potentially leading to increased overall noise levels

### **Chord Length (c)**

The chord length affects the Reynolds number of the flow and the size of the airfoil's wake. Larger chord lengths generally:

* Increase the Reynolds number, potentially leading to earlier transition to turbulent flow
* Provide a larger surface area for noise generation
* Influence the frequency content of the generated noise

### **Free-stream Velocity (U∞)**

The free-stream velocity is a critical parameter in airfoil noise generation. Higher velocities typically result in:

* Increased overall noise levels due to higher dynamic pressures
* Shifted noise spectra towards higher frequencies
* Potential changes in the dominant noise generation mechanisms

### **Boundary Layer Thickness (Δ)**

The boundary layer thickness is an important factor in airfoil self-noise, particularly for turbulent boundary layer trailing edge noise. A thicker boundary layer generally:

* Increases the low-frequency content of the noise spectrum
* May lead to earlier flow separation at higher angles of attack
* Affects the efficiency of noise reduction techniques such as serrated trailing edges

A graph with blue rectangular bars

Description automatically generated with medium confidence

Figure 5: Feature Importances

## **MACHINE LEARNING MODEL ANALYSIS**

The machine learning model developed for this dataset employs a Bagging Regressor with Random Forest as the base estimator. This ensemble approach combines multiple decision trees to create a robust and accurate prediction model for the Scaled Sound Pressure Level (SSPL).

**Model Performance Metrics**

The model's performance is evaluated using several metrics:

1. Mean Squared Error (MSE): 3.9273
2. Root Mean Squared Error (RMSE): 1.9817
3. R-squared Score: 0.9216
4. Cross-validation (CV) RMSE scores: [1.9255, 1.9523, 2.3630, 2.1320, 2.4035]
5. Average CV RMSE: 2.1553

These metrics indicate that the model performs well in predicting the SSPL. The R-squared score of 0.9216 suggests that the model explains approximately 92.16% of the variance in the target variable. The RMSE of 1.9817 dB indicates that, on average, the model's predictions deviate from the actual SSPL values by about 2 dB, which is relatively small considering the typical range of SSPL values in airfoil noise studies.

### **Residual Analysis**

Examining the residual plots (residual\_distributions.png and residuals\_vs\_predicted.png) provides further insights into the model's performance:

1. Distribution of Residuals (residual\_distributions.png):
   * The histogram shows a roughly normal distribution of residuals centred around zero.
   * This indicates that the model's errors are generally unbiased and symmetrically distributed.
   * The slight right skew suggests that the model might occasionally underpredict SSPL values.
2. Residuals vs. Predicted SSPL (residuals\_vs\_predicted.png):
   * The scatter plot shows no clear pattern or trend in the residuals across the range of predicted SSPL values.
   * This suggests that the model's performance is consistent across different noise levels.
   * The relatively uniform spread of residuals indicates homoscedasticity, which is a desirable property for regression models.

A purple graph with numbers

Description automatically generated

Figure 6: Distribution of Residuals

A green dots on a white background

Description automatically generated

Figure 7: Residuals vs Predicted SSPL

### **Actual vs. Predicted SSPL Analysis**

The scatter plot of Actual vs. Predicted SSPL (actual\_vs\_predicted.png) provides valuable insights:

* The points cluster closely around the perfect prediction line (red dashed line), indicating good overall agreement between predicted and actual values.
* The model performs well across the entire range of SSPL values, from low to high noise levels.
* There is a slight tendency for underprediction at very high SSPL values and overprediction at very low SSPL values, which is common in regression models and may be due to the limited number of extreme cases in the training data.

A graph with blue dots

Description automatically generated

Figure 8: Actual vs Predicted SSPL

### **New Predictions Analysis**

The scatter plot of Actual Data vs. New Predictions (actual\_vs\_new\_predictions.png) shows:

* The new predictions (red points) generally follow the distribution of the actual data (blue points).
* The model captures the overall trend and variability in the relationship between SSPL and frequency.
* There are some areas where the new predictions appear to be more clustered or sparse compared to the actual data, which may indicate regions where the model's performance could be improved with additional training data or feature engineering.

A screen shot of a graph

Description automatically generated

Figure 9: Actual vs New Predictions

## **ENGINEERING AND PHYSICS IMPLICATIONS/APPLICATIONS**

The machine learning model's ability to accurately predict SSPL values based on airfoil and flow parameters has several important implications for aeroacoustic (and thus, aerospace) engineering:

1. Design Optimization: The model can be used to rapidly evaluate different airfoil designs and operating conditions, allowing engineers to optimize for reduced noise while maintaining aerodynamic performance.
2. Noise Source Identification: By analysing the model's feature importances and predictions, engineers can gain insights into the dominant noise sources under different conditions, helping to focus noise reduction efforts.
3. Performance Envelope Exploration: The model enables quick exploration of the airfoil's acoustic performance across a wide range of operating conditions, helping to define safe and quiet operating envelopes for aircraft or wind turbines.
4. Scaling Effects: The inclusion of chord length and boundary layer thickness in the model allows for the study of scaling effects on airfoil noise, which is crucial for translating wind tunnel results to full-scale applications.
5. Interdependency Analysis: The model captures complex interactions between parameters like angle of attack, velocity, and frequency, providing a tool for studying how these factors jointly influence noise generation.
6. Rapid Prototyping: The speed of machine learning predictions compared to computational fluid dynamics (CFD) simulations allows for rapid prototyping and iteration in the early stages of airfoil design.

## **CONCLUSION**

The machine learning model developed for the NASA Airfoil Self-Noise Dataset demonstrates strong predictive capabilities, with an R-squared score of 0.9216 and an RMSE of 1.9817 dB. These results indicate that the model captures the complex relationships between airfoil parameters and the resulting Scaled Sound Pressure Level (SSPL) with high accuracy.

The model's performance is consistent across different noise levels and operating conditions, as evidenced by the residual analysis and actual vs. predicted SSPL plots. This consistency suggests that the model has successfully learned the underlying physics governing airfoil self-noise generation.

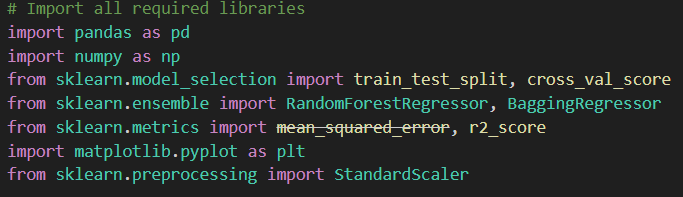
While the model shows excellent overall performance, there are opportunities for further improvement, particularly in predicting extreme SSPL values. This could potentially be addressed by incorporating additional relevant features, gathering more data for underrepresented operating conditions, or exploring more advanced machine learning techniques such as deep neural networks or gradient boosting algorithms.

The developed model serves as a valuable tool for aeroacoustic engineers and researchers, enabling rapid evaluation of airfoil designs, exploration of noise reduction strategies, and deeper understanding of the complex interplay between flow parameters and noise generation mechanisms. As the demand for quieter and more efficient aircraft and wind turbines continues to grow, such data-driven approaches will play an increasingly important role in advancing the field of aeroacoustics and driving innovation in low-noise airfoil design.

## **PYTHON CODE**

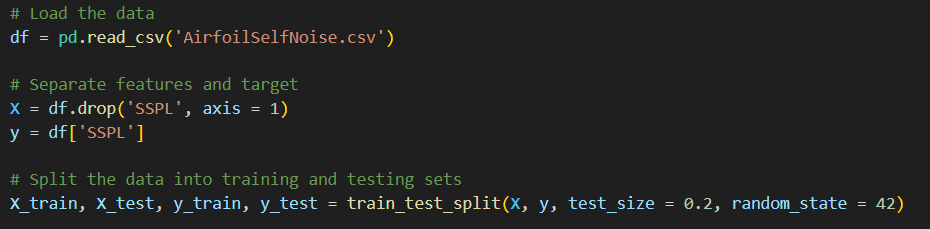
1. # import required libraries
2. import pandas as pd
3. import numpy as np
4. from sklearn.model\_selection import train\_test\_split, cross\_val\_score
5. from sklearn.ensemble import RandomForestRegressor, BaggingRegressor
6. from sklearn.metrics import mean\_squared\_error, r2\_score
7. import matplotlib.pyplot as plt
8. from sklearn.preprocessing import StandardScaler
9. # Load the data
10. df = pd.read\_csv('AirfoilSelfNoise.csv')
11. # Separate features and target
12. X = df.drop('SSPL', axis = 1)
13. y = df['SSPL']
14. # Split the data into training and testing sets
15. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)
16. # Scale the features
17. scaler = StandardScaler()
18. X\_train\_scaled = scaler.fit\_transform(X\_train)
19. X\_test\_scaled = scaler.transform(X\_test)
20. # Create a bagging regressor with Random Forest as the estimator
21. rf = RandomForestRegressor(n\_estimators = 100, random\_state = 42)
22. bagging\_rf = BaggingRegressor(estimator = rf, n\_estimators = 10, random\_state = 42)
23. # Fit the model
24. bagging\_rf.fit(X\_train\_scaled, y\_train)
25. # Make predictions
26. y\_pred = bagging\_rf.predict(X\_test\_scaled)
27. # Calculate metrics
28. mse = mean\_squared\_error(y\_test, y\_pred)
29. rmse = np.sqrt(mse)
30. r2 = r2\_score(y\_test, y\_pred)
31. print(f"\nMean Squared Error: {mse}")
32. print(f"Root Mean Squared Error: {rmse}")
33. print(f"R-squared Score: {r2}")
34. # Perform cross-validation
35. cv\_scores = cross\_val\_score(bagging\_rf, X\_train\_scaled, y\_train, cv = 5, scoring = 'neg\_mean\_squared\_error')
36. cv\_rmse = np.sqrt(-cv\_scores)
37. print(f"\nCross-validation RMSE scores: {cv\_rmse}")
38. print(f"Average CV RMSE: {np.mean(cv\_rmse)}")
39. # Plot actual vs predicted values
40. plt.figure(figsize=(12, 8))
41. plt.scatter(y\_test, y\_pred, alpha = 0.5, color = 'blue', label = 'Test Data')
42. plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw = 2, label = 'Perfect Prediction')
43. plt.xlabel('Actual SSPL')
44. plt.ylabel('Predicted SSPL')
45. plt.title('Actual vs Predicted SSPL')
46. plt.legend()
47. plt.tight\_layout()
48. plt.savefig('actual\_vs\_predicted.png')
49. plt.close()
50. # Plot feature importances
51. feature\_importance = bagging\_rf.estimators\_[0].feature\_importances\_
52. feature\_names = X.columns
53. plt.figure(figsize = (12, 8))
54. bars = plt.bar(feature\_names, feature\_importance)
55. plt.xlabel('Features')
56. plt.ylabel('Importance')
57. plt.title('Feature Importances')
58. plt.xticks(rotation = 45)
59. # Add value labels on top of each bar
60. for bar in bars:
61. height = bar.get\_height()
62. plt.text(bar.get\_x() + bar.get\_width()/2., height, f'{height:.4f}', ha = 'center', va = 'bottom')
63. plt.tight\_layout()
64. plt.savefig('feature\_importances.png')
65. plt.close()
66. # Plot residuals
67. residuals = y\_test - y\_pred
68. plt.figure(figsize = (12, 8))
69. plt.scatter(y\_pred, residuals, alpha = 0.5, color = 'green', label = 'Residuals')
70. plt.xlabel('Predicted SSPL')
71. plt.ylabel('Residuals')
72. plt.title('Residuals vs Predicted SSPL')
73. plt.axhline(y = 0, color = 'r', linestyle = '--', label = 'Zero Residual Line')
74. plt.legend()
75. plt.tight\_layout()
76. plt.savefig('residuals\_vs\_predicted.png')
77. plt.close()
78. # Plot error distribution
79. plt.figure(figsize = (12, 8))
80. plt.hist(residuals, bins = 30, color = 'purple', alpha = 0.7, label = 'Residuals')
81. plt.xlabel('Residuals')
82. plt.ylabel('Frequency')
83. plt.title('Distribution of Residuals')
84. plt.legend()
85. plt.tight\_layout()
86. plt.savefig('residuals\_distribution.png')
87. plt.close()
88. # Generate new data for predictions
89. new\_data = X.sample(n = 100, random\_state = 42)
90. new\_data\_scaled = scaler.transform(new\_data)
91. new\_predictions = bagging\_rf.predict(new\_data\_scaled)
92. # Plot actual data vs new predictions
93. plt.figure(figsize = (12, 8))
94. plt.scatter(y, X['f'], alpha = 0.5, color = 'blue', label = 'Actual Data')
95. plt.scatter(new\_predictions, new\_data['f'], alpha = 0.5, color = 'red', label = 'New Predictions')
96. plt.xlabel('SSPL')
97. plt.ylabel('Frequency (Hz)')
98. plt.title('Actual Data vs New Predictions')
99. plt.legend()
100. plt.tight\_layout()
101. plt.savefig('actual\_vs\_new\_predictions.png')
102. plt.close()
103. print("\nAll plots have been saved as PNG files.")

### **Code Explanation**

****

**Libraries Import:**

* **import pandas as pd:** pandas is imported to handle the dataset. pandas simplifies data manipulation and handling, allowing for easy loading, exploration, and transformation of tabular data. Here, it's crucial for reading the data and structuring it into features and target variables. It provides data structures like DataFrames, which are essential for handling tabular data (like CSV files).
* **import numpy as np:** numpy is used for numerical computations, such as calculating the root mean squared error (RMSE) and other statistical metrics. numpy's array structures also support efficient computation with pandas and scikit-learn.
* **from sklearn.model\_selection import train\_test\_split, cross\_val\_scoreB:** The train\_test\_split function divides the dataset into training and testing subsets, while cross\_val\_score performs cross-validation, a method for assessing model performance by training on different data folds.
* **from sklearn.ensemble import RandomForestRegressor, BaggingRegressor:** This module contains ensemble methods, which combine multiple models to improve prediction accuracy. The RandomForestRegressor is a machine learning model used here as the base estimator, that uses multiple decision trees to make predictions, within a BaggingRegressor, which is a bagging ensemble method that fits base learners (like decision trees) on random subsets of the data; creates multiple instances of RandomForestRegressor and averages their results, providing robustness to the model.
* **from sklearn.metrics import mean\_squared\_error, r2\_score:** These functions calculate evaluation metrics for the model. mean\_squared\_error computes the mean of squared errors between predicted and actual values, and r2\_score calculates the R-squared score, which measures the proportion of variance explained by the model.
* **import matplotlib.pyplot as plt:** matplotlib is used for creating static, animated, and interactive visualizations. **pyplot** is a module within Matplotlib that provides a MATLAB-like interface for plotting. Plots in this code display relationships between predicted and actual values, feature importance, and residuals, helping to interpret model performance visually.
* **from sklearn.preprocessing import StandardScaler:** This module provides functions to preprocess data before training a model. StandardScaler normalizes features, which scales the data, so each feature has a mean of 0 and a standard deviation of 1. This scaling improves the model’s convergence speed and accuracy.



**File Reading and Standardization**

* **pd.read\_csv:** Loads the dataset from a CSV file (‘AirfoilSelfNoise.csv’) into a pandas DataFrame denoted by df, which organizes the data in rows and columns, for further processing.
* **df.drop('SSPL', axis = 1):** Separates the features (X) by dropping the target column ('SSPL'), while **df['SSPL']** isolates this target variable (y), which represents the sound pressure level to be predicted – simply, X contains all columns except SSPL, while **y** contains only the SSPL column.
* **train\_test\_split:** Splits X and y into training and testing sets. Here, 80% of the data is used for training (X\_train, y\_train) and 20% for testing (X\_test, y\_test). The setting **random\_state = 42** ensures reproducibility of the split.

A screen shot of a computer code

Description automatically generated

**Feature Scaling and Model Initialization**

* The features are standardized using **StandardScaler()**, which scales them to have a mean of 0 and a standard deviation of 1 thus normalizing the features for more efficient training.
* **fit\_transform:** Calculates and applies scaling to X\_train, while **transform** applies this same scaling to X\_test to maintain consistency.
* **RandomForestRegressor:** Creates a model consisting of 100 decision trees. Random Forest is robust to noise and useful for reducing variance, providing stable predictions.
* **BaggingRegressor** uses **RandomForestRegressor** as its base estimator, averaging the predictions of 10 separate Random Forest models i.e. 10 different bootstrap samples or 10 bagging iterations. Bagging helps mitigate overfitting and increases model stability by training multiple models on different data subsets.
* The setting **random\_state = 42** ensures reproducibility.

A screenshot of a computer program

Description automatically generated

**Fitting and Predicting**

* **fit:** Trains the Bagging model on the scaled training data, learning the relationship between features and the target variable y\_train.
* **predict:** Generates predictions (y\_pred) for the test set based on the learned model.

A screen shot of a computer code

Description automatically generated

**Metrics Calculation**

* **mean\_squared\_error:** Calculates the average squared differences between actual (y\_test) and predicted (y\_pred) values.
* **np.sqrt(mse):** Computes RMSE, providing an error metric in the same units as the target.
* **r2\_score:** Calculates the R² metric, indicating how well the model explains the variance (by indicating its proportion) in y\_test.

A computer screen with text

Description automatically generated

**Cross-Validation**

* **cross\_val\_score:** Performs 5-fold cross-validation on the training data, providing a more reliable assessment of model performance. Using scoring='neg\_mean\_squared\_error', cross\_val\_score outputs negative MSE (mean-squared-error) for each fold (i.e. each cross-validation iteration), which is then converted to RMSE (root-mean-squared-error) through **np.sqrt(-cv\_scores).**

A computer code on a black background

Description automatically generated

**Actual vs Predicted Graph**

* **plt.scatter:** Plots  **y\_test vs y\_pred** to visualize the accuracy of predictions of actual vs. predicted SSPL values.
* **plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw = 2):** Adds a red dashed line representing perfect predictions.
* **plt.xlabel, plt.ylabel, plt.title:** Labels the axes and sets the plot title.
* **plt.legend:** Adds a legend to the plot.
* **plt.tight\_layout:** Adjusts the layout to fit all elements.
* **plt.savefig:** Saves this plot as an image (‘actual\_vs\_predicted.png’)

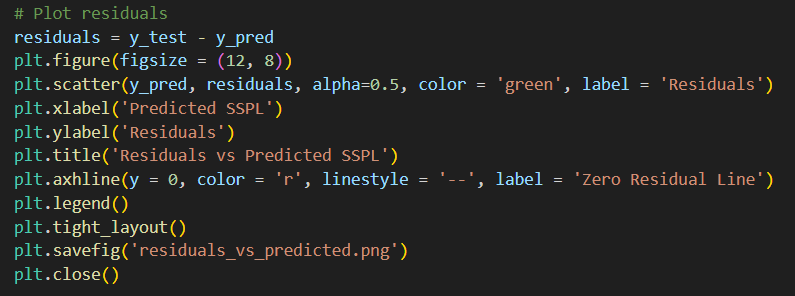
*[The above functions will also be used later for other graphs]*

A screen shot of a computer program

Description automatically generated

**Feature Importances Graph**

* **bagging\_rf.estimators\_[0].feature\_importances\_:** Extracts feature importances from one of the Random Forest models in the bagging ensemble, which indicate the relative contribution of each feature to the model's predictions.
* **plt.bar:** Visualizes these importances for each feature, aiding in identifying which features most influence SSPL.
* **plt.xticks(rotation = 45):** Rotates the feature names for better readability.
* **for bar in bars:** Annotates each bar with its value, making feature importance easier to interpret.

****

**Residuals vs Predicted Graph**

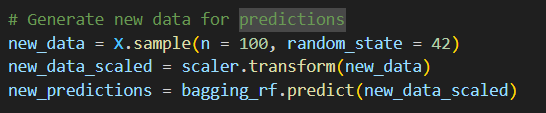
* **plt.scatter:** Plots the residuals (differences between predicted and actual values) against predicted values of SSPL, where a horizontal line at 0, given by **plt.axhline(y = 0)** indicates no error or zero residuals, revealing any model bias or inconsistencies.

A computer screen with colorful text

Description automatically generated

**Residuals Distribution Graph**

* **plt.hist:** Creates a histogram of residuals to visualise the distribution of errors, useful for detecting skewness or outliers.
* **bins = 30:** Sets the number of bins in the histogram.

****

**New Data Generation**

* **X.sample(n = 100):** Selects 100 random samples from X i.e. the original feature space to simulate new, random data.
* **scaler.transform(new\_data):** Scales the new data using the previously fitted scaler.
* **bagging\_rf.predict:** Generates predictions for this new scaled data.

A screen shot of a computer program

Description automatically generated

**Actual vs New Predictions Graph**

* **plt.scatter:** Plots (and compares) the actual and new predicted SSPL values, with actual data in blue and new predictions in red, and frequency on the y-axis, helping to gauge model generalizability on unseen data.
* In the end, a message is printed to indicate that all plots have been saved as PNG files.

## **REFERENCES**

1. Brooks, T.F., Pope, D.S. and Marcolini, A.M. Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989
2. Brooks, T., Pope, D., & Marcolini, M. (1989). Airfoil Self-Noise [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5VW2C>. (Dataset from Dr. Lopez, R. NASA Airfoil Self-Noise Dataset. [NASA Airfoil Self-Noise Dataset](https://www.kaggle.com/datasets/fedesoriano/airfoil-selfnoise-dataset/data))
3. Smith, M. J. T. Aircraft Noise. Cambridge Aerospace Series, Cambridge University Press. 2004
4. Anselmet, F., Mattei, Pierre-Olivier. Acoustics, Aeroacoustics and Vibrations. Wiley. 2016
5. Glegg, S., Devenport, W. Aeroacoustics of Low Mach Number Flows (2nd edition). Academic Press, Elsevier. 2024
6. Lopez, R. Predict airfoil self-noise using machine learning. neuraldesigner. 2023 ([Predict airfoil self-noise using machine learning](https://www.neuraldesigner.com/learning/examples/airfoil-self-noise-prediction/#DataSet))