MARMARA UNIVERSITY
FACULTY OF ENGINEERING

MACHINE LEARNING BASED INVERSE AERODYNAMIC DESIGN OF RAMJET ENGINE

Yunus Gözel 150420071, Süleyman Altundağ 150421002

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Prof. Dr. Emre Alpman

ISTANBUL, 2025

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

Machine learning based inverse aerodynamic design of Ramjet engine

by

Yunus Gözel, Süleyman Altundağ

June 23, 2025, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s) .....

Department of Mechanical Engineering

Certified By .....

Project Supervisor, Department of Mechanical Engineering

Accepted By.....

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, we would like to thank our supervisor Prof Dr. Emre Alpman for the valuable guidance and advice on preparing this thesis and giving me moral and material support.

June, 2025 Yunus Gözel, Süleyman Altundağ

Contents

ABSTRACT

Bu çalışma, ramjet motoru giriş tasarımının ters aerodinamik tasarımı için makine öğrenmesi tabanlı bir yaklaşım sunmaktadır. Halihazırdaki hava alığı tasarımı yöntemleri, zaman alıcı manuel hesaplamalara ve hesaplama açısından yoğun Hesaplamalı Akışkanlar Dinamiği analizlere dayanmaktadır. Bu projede, bir referans ramjet alık modeli değişkenlerle bağlı şekilde tasarlanmış ve bu sayede binlerce farklı tasarım oluşturulmuştur. Her bir tasarım ilgili koşullar altında analiz edilmeye çalışılmıştır fakat bazı tasarımlardaki fiziksel uygunsuzluklar analizlerin hata vermesine sebep olmuştur. Hata gerçekleşmeyen analizlerin sonuçları tek bir dosya içerisinde kaydedilmiştir. Elde edilen veri seti filtrelenerek bir makine öğrenmesi modeli eğitilmiş ve bu model, giriş ve çıkış Mach sayıları ile kütle akış oranı gibi kullanıcı tarafından belirtilen parametrelere dayalı olarak hızlı ön tasarım önerileri sunmaktadır. Proje için geliştirilen yazılım, makine öğrenmesi modelinin önerdiği tasarımların Hesaplamalı Akışkanlar Dinamiği analizini gerçekleştirme ve kullanıcı gereksinimleriyle analiz sonuçlarını karşılaştırma imkânı sunmaktadır. Ayrıca, kullanıcılar kendi veri tabanlarını oluşturarak yeni Hesaplamalı Akışkanlar Dinamiği analizleriyle modellerini eğitebilir, böylece daha özelleştirilmiş ve verimli tasarımlar elde edebilir. Makine öğrenmesi modeli, tutarlı ve güvenilir ön tasarımlar üreterek ramjet giriş tasarım sürecini önemli ölçüde hızlandırmış ve daha verimli, çevre dostu bir yaklaşım sunmuştur.

This paper presents a machine learning based approach for the inverse aerodynamic design of the ramjet engine inlet design. Current design methods are based on time-consuming manual calculations and computationally intensive Computational Fluid Dynamics (CFD) analysis. In this project, a reference ramjet inlet template was designed in a variable dependent manner, resulting in thousands of different designs. Each design was tried to be solved under the relevant conditions, but physical incompatibility in some designs caused the analysis to diverge. The results of the acceptable analyses were saved in a single file. The resulting data set was filtered, and a machine learning model was trained, which provides fast preliminary design recommendations based on user-specified parameters such as inlet and outlet Mach numbers and mass flow rate. The developed software offers the possibility to perform CFD analysis of the designs suggested by the model and compare them with user requirements. Furthermore, users can create their own databases and train their models with

new CFD analysis, thus achieving more customized and efficient designs. By producing consistent and reliable preliminary designs, the machine learning model has significantly accelerated the ramjet inlet design process and provided a more efficient and environmentally friendly approach.

SYMBOLS

$\boldsymbol{\rho}$	:Density
$\boldsymbol{p}$	:Static pressure
$\boldsymbol{T}$	:Temperature
$\boldsymbol{M}$	:Mach number
$\boldsymbol{\gamma}$	:Specific heat ratio
$\boldsymbol{\theta}$	:Ramp (cone) angle
$\boldsymbol{\beta}$	:Shock wave angle
$\boldsymbol{p_0}$	:Total (stagnation) pressure
$\boldsymbol{T_0}$	:Total (stagnation) temperature
$\boldsymbol{V_0}$	:Freestream velocity
$\boldsymbol{V_e}$	:Exit velocity
$\boldsymbol{P_0}$	:Ambient pressure
$\boldsymbol{P_e}$	:Exit pressure
$\boldsymbol{A_e}$	:Exit area
$\boldsymbol{\dot{m}}$	:Mass flow rate of air
$\boldsymbol{f}$	:Fuel-to-air ratio
$\boldsymbol{LHV}$	:Lower Heating Value
$\boldsymbol{c_p}$	:Specific heat at constant pressure
$\boldsymbol{\eta_{thermal}}$	:Thermal efficiency
$\boldsymbol{\eta_{prop}}$	:Propulsive efficiency
$\boldsymbol{\eta_{overall}}$	:Overall efficiency

$\boldsymbol{PRC}$	:Pressure Recovery Coefficient
$\boldsymbol{Re}$	:Reynolds number
$\boldsymbol{k}$	:Turbulent kinetic energy
$\boldsymbol{\varepsilon}$	:Turbulent dissipation rate
$\boldsymbol{\omega}$	:Specific dissipation rate
$\boldsymbol{R^2}$	:Coefficient of determination (ML metric)
$\boldsymbol{MSE}$	:Mean Squared Error (ML metric)

ABBREVIATION

CFD :Computational Fluid Dynamics

SU2 :Stanford University Unstructured

RANS :Reynolds-Averaged Navier-Stokes

GUI :Graphical User Interface

ML :Machine Learning

XGBoost :Extreme Gradient Boosting

IQR :Interquartile Range

MSE :Mean Squared Error

CSV :Comma-Separated Values

FVM :Finite Volume Method

ISA :International Standard Atmosphere

PDE :Partial Differential Equation

SA :Spalart–Allmaras

SST :Shear Stress Transport

ROE :Roe Approximate Riemann Solver

JST :Jameson-Schmidt-Turkel

HLLC :Harten-Lax-van Leer-Contact

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Supersonic speed, defined as velocities exceeding the speed of sound is a critical regime in aerospace engineering, enabling rapid transit and advanced military applications. At these speeds, air behaves compressible, forming shock waves that significantly influence aerodynamic performance. Ramjet engines, a type of air-breathing propulsion system, are uniquely suited for supersonic speed. Unlike turbojets, ramjets lack rotating components, relying on the vehicle's high-speed forward motion to compress incoming air through shock waves at the inlet. This simplicity makes ramjets efficient for high-speed applications but requires precise inlet design to optimize compression and ensure stable combustion. The need for ramjet engines arises in applications such as missiles, high-speed aircraft, and space access vehicles, where sustained supersonic performance is essential for mission success.

Designing ramjet engine inlets is a complex challenge, as the inlet must efficiently compress air while minimizing pressure losses and maintain compatibility with desired Mach numbers and mass flow rates. Traditional design methods depend on manual calculations and computationally expensive Computational Fluid Dynamics (CFD) analysis, which are time-intensive and limit rapid exploration of design configurations. These iterative processes often hinder optimization for specific performance metrics, such as pressure recovery or flow stability.

In recent years, machine learning (ML) has emerged as a transformative tool to address such engineering challenges by leveraging data-driven insights to accelerate design processes. By training models on CFD-derived datasets, ML can predict optimal design parameters, reducing reliance on extensive simulations. This project presents a machine learning-based approach for the inverse aerodynamic design of ramjet engine inlets, aiming to enhance efficiency and streamline the design process.

The primary objective is to develop an Random Forest-XGBoost algorithms based multi-output regression model to predict ramjet inlet design parameters, including ramp count, ramp length, and cone angle, based on user inputs such as inlet and outlet Mach numbers and mass flow rate. A parameterized ramjet inlet template was used to generate thousands of design variations, analyzed via CFD tools (Gmsh, SU2, and Paraview) to create a robust dataset. After filtering physically incompatible designs, the dataset trained a model

integrated into a user-friendly graphical user interface (GUI). This GUI enables engineers to input parameters, validate designs with CFD, and retrain the model with new data.

By combining machine learning with CFD, this approach significantly reduces design time, computational costs, and environmental impact by minimizing simulation requirements. This project offers a faster, more reliable, and sustainable method for ramjet inlet design, contributing to advancements in high-speed propulsion systems for aerospace applications.

2. THEORETICAL BACKGROUND

2.1.Machine Learning

Machine learning is an area of computer science that focuses on teaching computers to improve their performance on a task by learning from data and experience. Instead of giving a computer exact instruction, we provide it with information, and it learns to find patterns and make decisions on its own. As Tom M. Mitchell (1997) defines it, “A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance at task T, as measured by P, improves with experience E” (p. 2). In the long run, this field helps solve everyday problems more efficiently, with less computer processing and reduced human involvement.

Machine learning can learn in different ways. The most common is supervised learning, where the computer learns from examples that have clear answers, like predicting house prices or sorting emails into spam and not spam. Unsupervised learning doesn’t use labeled data—instead, it finds patterns or groups in the data, such as grouping customers with similar buying habits. Lastly, reinforcement learning teaches a computer by giving it rewards or penalties based on its actions, helping it learn the best decisions over time. This type is often used in games and robots. Together, these methods help computers learn from different kinds of information and improve their performance.

Machine learning tasks can be divided into four main categories: Regression, Classification, Clustering, and Embedding. Each category has different goals and is used for solving different types of problems.

2.1.1. Regression

Regression is a type of supervised learning used to predict continuous numerical values rather than separate categories. The model learns by finding the relationship between input data and matching output numbers from many examples. After learning from enough data, the model can predict the output for new inputs, even if it has never seen those exact examples before. For example, a regression model can predict the price of a house based on its size, location, and other features. To make good predictions, the model tries different ways to connect inputs and outputs. It must avoid two common problems: overfitting, which means the model works well only on the training data but fails with new data, and underfitting, which means the model is too simple and makes many errors even on known data. (Hastie et al., 2009) Algorithms like Linear Regression, Decision Trees, Support Vector Regression, and Neural Networks use various techniques to find the best relation between output and input.(Bishop, 2006)

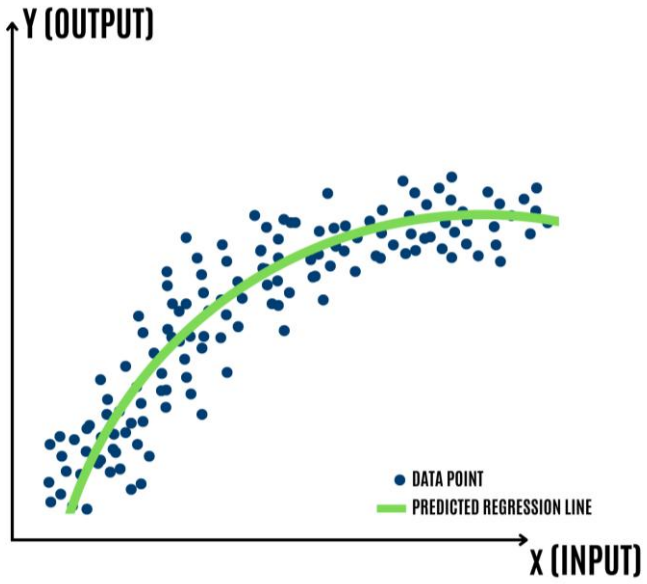

Figure 2.1:A Sample Regression for Finding Relation Between Output Y in Terms of Input X

2.1.2. Classification

Classification is a type of supervised learning where the goal is to predict a category or label instead of a number. The model learns from a dataset where each input has a known class or category. After training, the model can assign new, unseen data to one

of these categories. The main aim is to get as many correct predictions as possible and reduce the number of mistakes. For example, a classification model can be used to identify whether an email is “spam” or “not spam.” Common algorithms used for classification include Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM), and Neural Networks.(Hastie et al., 2009; Vapnik & Izmailov, 2020)

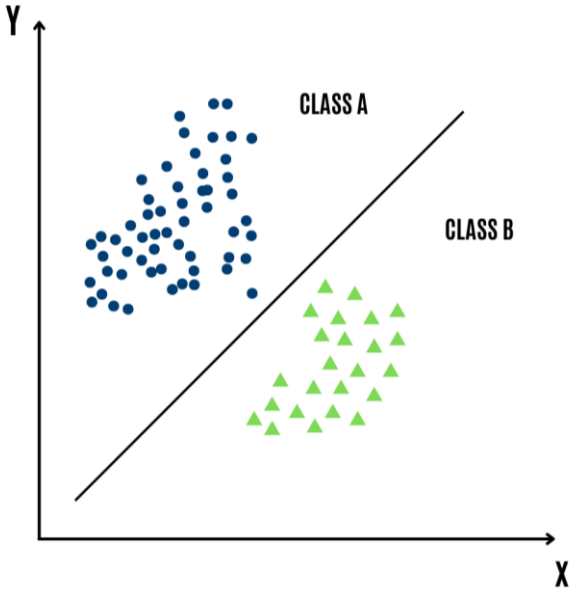

Figure 2.2: A Sample Graph That Shows 2 Different Data Divided Into 2 Classes via Line

2.1.3. Clustering

Clustering is a type of unsupervised learning used to find patterns or groupings in data that does not have labels. This means the data has no correct answers or categories given in advance. The algorithm looks at the similarities between data points and groups them into clusters, so that items in the same cluster are similar to each other and different from those in other clusters. For example, a company might use clustering to group customers by shopping behavior, even if there is no label like “high spender” or “low spender” in the data. The goal is to make the groups meaningful and useful for further analysis. Clustering is often used in market segmentation, social network analysis, document grouping, and detecting unusual data points. Common clustering methods include K-Means, Hierarchical Clustering, DBSCAN, and Gaussian Mixture Models.(Bishop, 2006; Jain & Dubes, 1988)

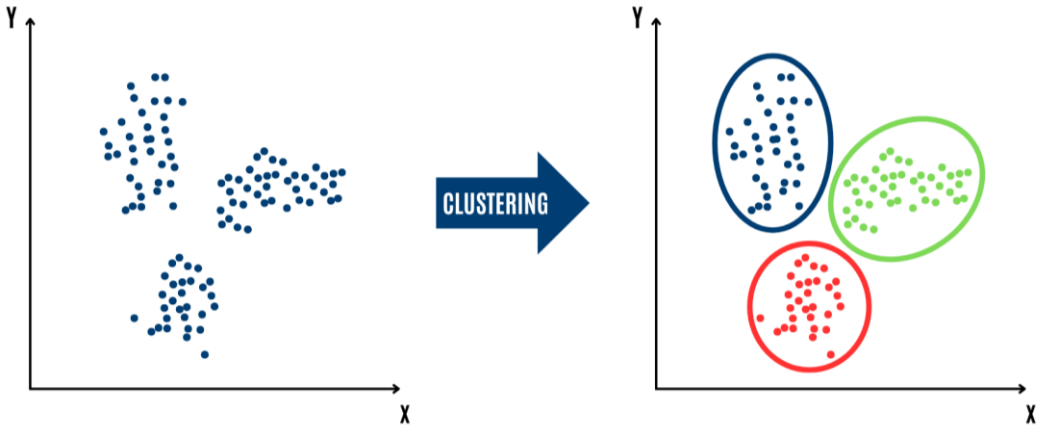

Figure 2.3: Diagram of Clustering Unlabeled Data Into 3 Different Clusters

2.1.4. Embedding

Embedding is a method used in machine learning to turn complex data into a simpler form that is easier for computers to understand and work with. It takes high-dimensional data—like words, pictures, or user behavior—and changes it into a lower-dimensional format, while keeping the most important relationships between the items.(Goodfellow et al., 2016) For example, in natural language processing, words can be turned into numbers using techniques like Word2Vec or GloVe. (Pennington et al., 2014) These numbers (called vectors) are designed so that words with similar meanings are placed close together. This helps the computer understand ideas like similarity or context. Embeddings are used in many modern AI systems, such as chatbots, recommendation engines, search tools, and deep learning models.

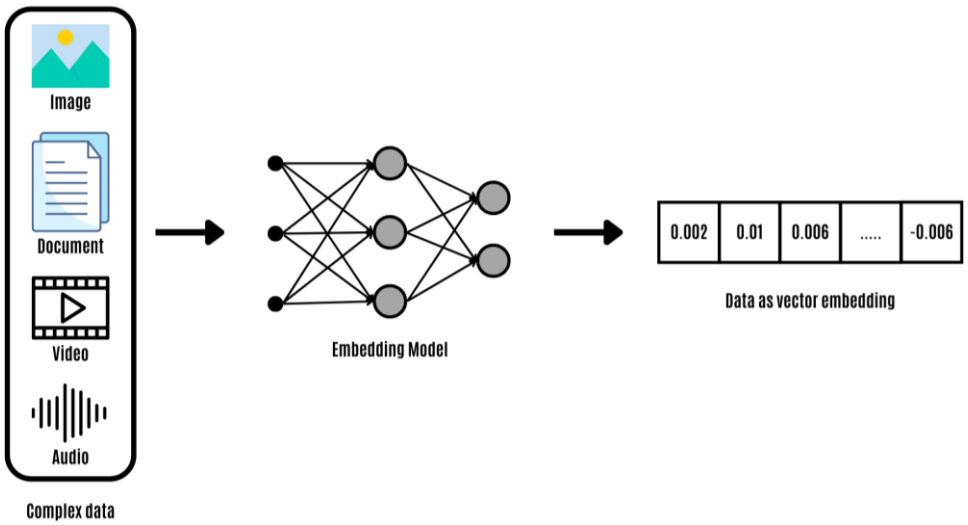

Figure 2.4: Diagram of Embedding

Table 2.1: Comparison of Main Machine Learning Tasks

Task	Learning Type	Output Type	Common Algorithms	Example Use Case
Regression	Supervised	Continuous numbers	Linear Regression, Decision Trees, SVR	Predicting house prices
Classification	Supervised	Categories/Labels	Logistic Regression, SVM, Random Forest	Email spam detection
Clustering	Unsupervised	Group(cluster) IDs	K-Means, DBSCAN, Hierarchical Clustering	Grouping customers by shopping behavior
Embedding	Representation learning (Unsupervised or semi-supervised)	Dense Numerical vectors	Word2Vec, GloVe, Autoencoders	Turning words into vectors for NLP tasks

2.2.Random Forest -XGBoosting

Random Forest Regressor is an ensemble learning algorithm used for regression tasks. It builds on Decision Trees, combining the predictions of multiple trees to improve accuracy, reduce overfitting, and increase robustness.(Breiman, 2001) Before understanding Random Forest, it is important to first explain how a single Decision Tree Regressor works.

2.2.1. Decision Tree

A Decision Tree is a model that looks like a tree. It helps us make predictions by dividing the dataset into smaller and smaller parts based on the values of different features. Each internal node asks a question about a feature (for example, "Is the temperature greater than 20°C?"). Each branch shows the result of that question (yes or no). Each leaf node gives a final prediction (in regression, this is usually a number). At every node, the algorithm

chooses the best feature and value to split the data. It does this by trying to make the prediction error as small as possible. (Quinlan, 1986)The error is usually measured using Mean Squared Error (MSE). The MSE at a node is calculated as:

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{n}(y_i - \bar{y})^2 \qquad (\mathbf{2.1})$$

Where:

y_i is actual value of the target variable,

$\bar{y}$ is the mean of the values in the current node,

n is the number of sampels in that nodes

(Hastie et al., 2009)

The algorithm keeps splitting the data until a certain rule tells it to stop. For example, it may stop when there are only a few data points left in a group or when the tree reaches a maximum depth. When we want to predict the output for a new input, we follow the path down the tree based on its feature values. In the end, we reach a leaf node, and the prediction is the average of the training outputs in that leaf. Structure of decision tree can be seen in Figure 2.5

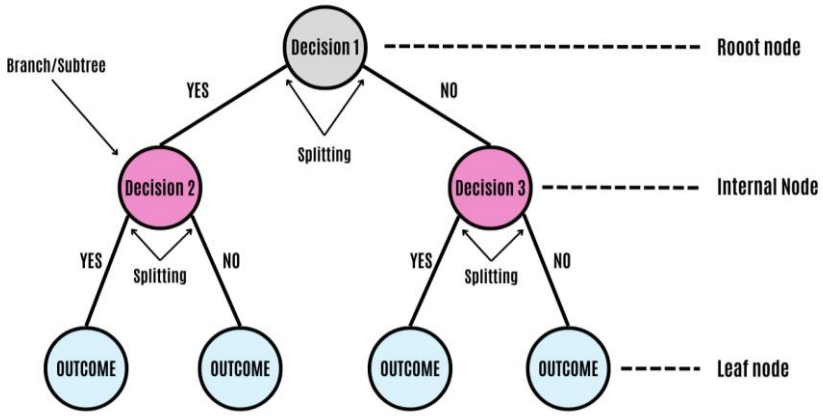

Figure 2.5: Structure of a Decision Tree

2.2.2. Random Forrest

A single Decision Tree is simple and easy to understand. However, it can often overfit the training data, especially if the tree is very deep. Overfitting means that the model

performs well on the training set but poorly on new, unseen data. To solve this problem, the Random Forest Regressor was developed. It improves accuracy and reduces overfitting by combining many decision trees into one model. This method is called bagging, short for bootstrap aggregating.

The idea behind Random Forest is to train multiple trees using different random parts of the dataset. For each tree, a new training set is created by randomly selecting data points with replacement from the original dataset. This process is known as bootstrap sampling. In addition, during the tree-building process, each split is made by considering only a random subset of the features rather than all available features. This extra layer of randomness helps create diverse trees, which improves the overall performance of the model.

Each tree in the forest is built either to full depth or until a stopping condition is met. Once all the trees are trained, the final prediction is made by averaging the outputs of all the trees. In regression tasks, this averaging process gives a more stable and accurate result than using a single tree. The prediction is calculated as:

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i \qquad (\mathbf{2.2})$$

where N is the number of trees in the forest, and $\hat{y}_i$ is the prediction from the i-th tree.

Random Forest models offer several advantages. They are generally more accurate than individual decision trees and are able to handle large datasets and many input features effectively. They are also robust to noise, missing values, and outliers, and they can model both linear and non-linear patterns in the data. However, one drawback is that Random Forests can be slower to train and make predictions when the number of trees is large.(Breiman, 2001; Hastie et al., 2009) Additionally, they are not as easy to interpret as a single decision tree because the final result comes from averaging many different trees.

A diagram can be helpful to visualize how Random Forest works. It usually shows several decision trees trained on different random samples of the data, and how their predictions are combined to make the final result. Figure 2.6 shows structure of Random Forest algorithm.

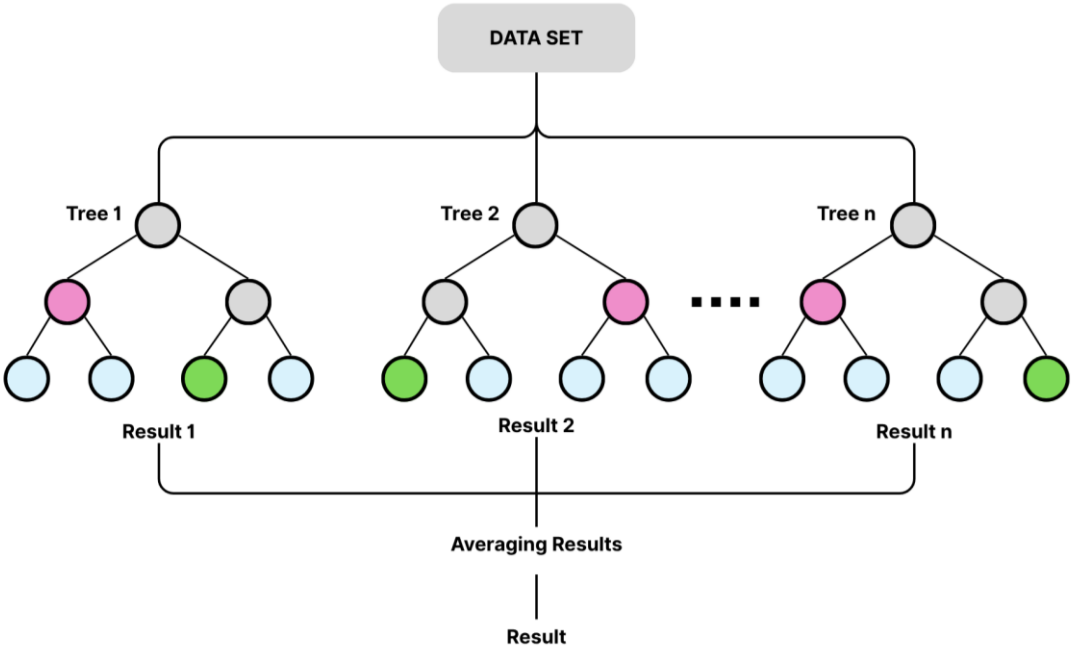

Figure 2.6: Overview of Random Forest Architecture

2.2.3. Boosting and XGBoosting

While Random Forest uses a method called bagging, which builds many decision trees at the same time and averages their results, boosting works in a different way. In boosting, the decision trees are built one after another, not in parallel. Each new tree focuses on the mistakes made by the previous trees. This way, the model keeps improving with each new step, and the final result becomes more accurate.

Boosting is a technique that combines many weak learners to create a strong model. A weak learner is usually a small decision tree that performs only slightly better than guessing. After each step, the algorithm gives more attention to the data points that were predicted incorrectly. As a result, the model gradually becomes better at making accurate predictions. In the end, the final prediction is a weighted combination of all the small trees. The prediction can be written as:

$$\hat{y}_i = \sum_{m=1}^{M} \alpha_m h_m(x_i) \qquad (2.3)$$

where $\hat{y}_i$ is the final prediction, $h_m(x_i)$ is the prediction made by the m-th small tree, α_m is the weight (or importance) of that tree, and M is the total number of trees used. (Friedman, 2001)

A visual comparison can help show the difference between bagging and boosting. In bagging (like Random Forest), all trees are trained at the same time. In boosting, trees are trained one after the other, with each tree learning from the mistakes of the one before it. Comparison of bagging and boosting algorithm structure can be seen in Figure 2.7

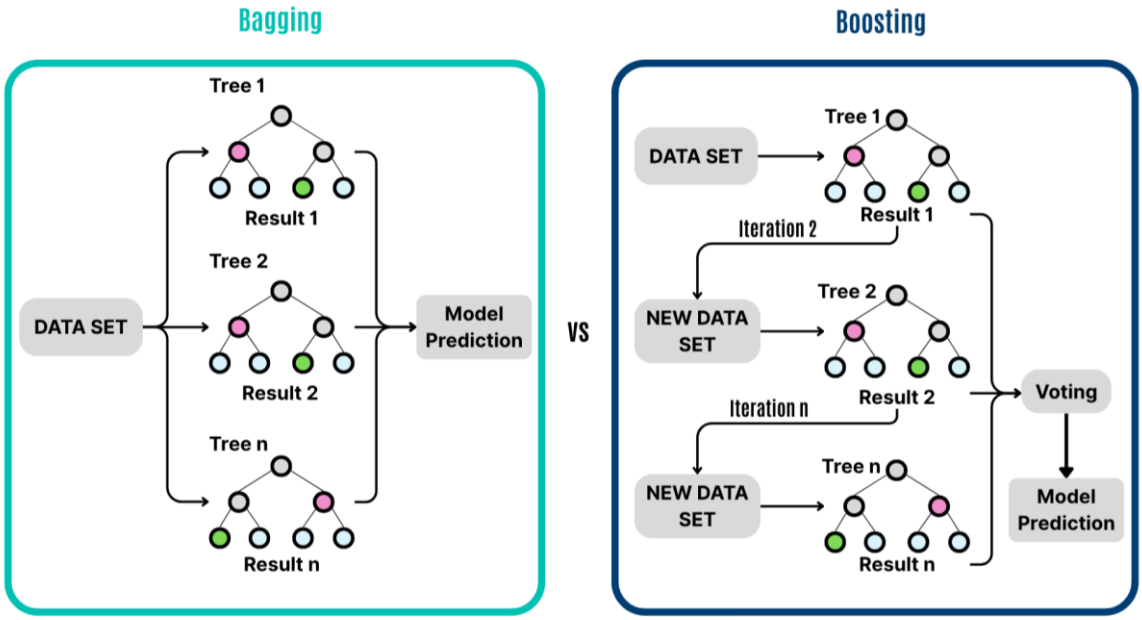

Figure 2.7: Comparison of Bagging and Boosting Algorithms

One of the most powerful and widely used boosting algorithms today is called XGBoost, which stands for Extreme Gradient Boosting.(Chen & Guestrin, 2016) XGBoost is an advanced version of boosting that improves performance in several ways. It is very fast because it uses computer memory and processing efficiently. It also includes regularization, which means it tries to avoid overfitting by controlling how complex the trees become. XGBoost can also handle missing data automatically and includes features like early stopping, which ends training if the model stops improving.

In XGBoost, each new tree is added by looking at the gradients, which measure how much the current prediction needs to change in order to reduce the total error. A simplified version of XGBoost's objective function looks like this(Chen & Guestrin, 2016):

$$\text{Objective} = \text{Loss} + \text{Penalty}) \tag{2.4}$$

Here, the Loss measures how far the predictions are from the actual values (for example, using Mean Squared Error), and the Penalty adds a cost for making the model too complex. This helps make sure that the model stays general and does not overfit the training data.

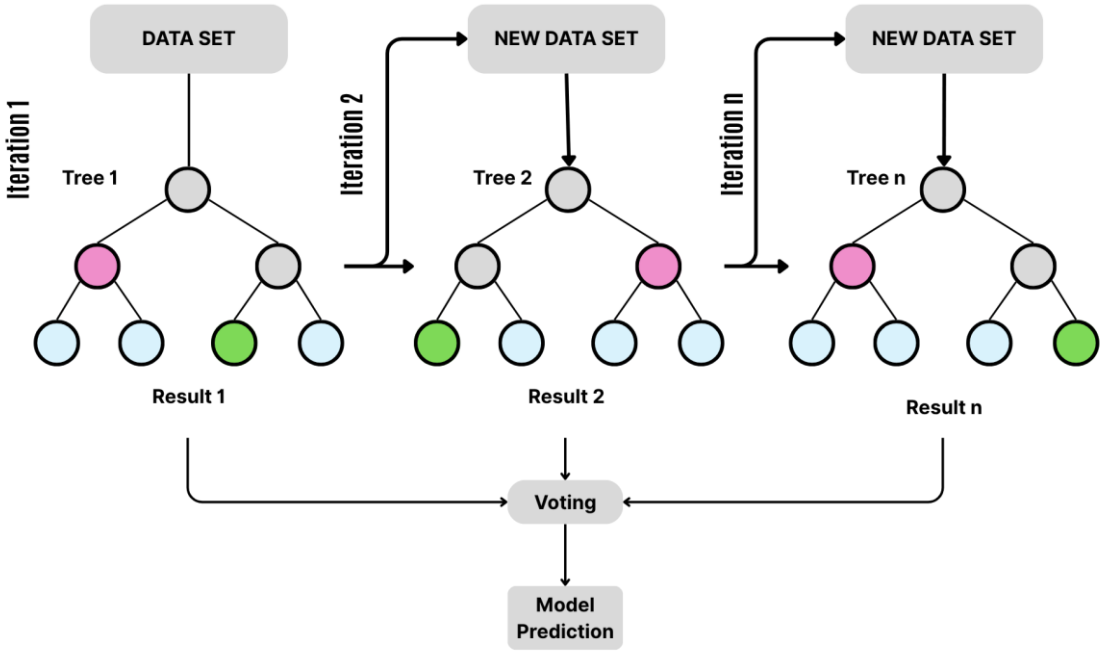

Figure 2.8: Illustration of the Boosting Process

Table 2.2: Comparison of Boosting Algorithms

Algorithm	Main Idea	Fast	Handles Missing Data	Notes
AdaBoost	Focuses on errors by giving them more weight	*	No	Good for simple problems
Gradient Boosting (GBM)	Adds trees to reduce mistakes gradually	*	No	Can be slow and overfit
XGBoost	Improved GBM with regularization and speed	***	Yes	Very popular and effective

LightGBM	Faster tree growth, good for big datasets	****	Yes	Best for large-scale problems
CatBoost	Works well with categorical (text) data	***	Yes	No need to convert categories

2.3.Hyperparameter Tuning in Gradient Boosting Mode

In supervised machine learning, the training process is governed by two types of parameters: model parameters and hyperparameters. Model parameters are internal values that are learned from the training data itself such as weights in a regression model or split points in a decision tree. In contrast, hyperparameters are external settings that must be defined before the training begins. These values guide the learning process and influence the model's capacity, speed, and generalization ability. In this project, the algorithm chosen for prediction is XGBoost (Extreme Gradient Boosting), a powerful ensemble method based on decision trees. XGBoost builds trees se- quentially, where each new tree aims to correct the errors made by the previous ones. This step-by-step correction forms a strong predictive model from many weak learners. The effec- tiveness and behavior of XGBoost are strongly determined by several key hyperparameters. The most important hyperparameters used in this study are as follows:

Number of Estimators (n_estimators): This determines the number of trees (boosting rounds) in the ensemble. A higher number allows the model to learn more complex patterns, but may increase the risk of overfitting if not properly regularized.

Learning Rate (learning_rate): This shrinkage factor controls how much each individual tree contributes to the final prediction. A smaller learning rate generally leads to better generalization but requires more trees to achieve convergence.

Maximum Tree Depth (max_depth): This sets the maximum depth for each decision tree. Deeper trees allow the model to capture more complex relationships, but may also cause overfitting, especially on small datasets.

Cross-Validation Folds (K in K-Fold Cross-Validation): Cross-validation is used to assess how well the model generalizes to unseen data.(Kohavi, 1995) In K-Fold cross-validation, the dataset is split into K subsets; the model is trained on K-1 folds and validated on the

remaining one. This process is repeated K times to obtain a reliable performance estimate.(Chen & Guestrin, 2016)

Hyperparameter tuning is a critical part of model development. Poorly chosen values can lead to underfitting (insufficient learning) or overfitting (memorizing the training data instead of learning patterns). In this work, the hyperparameters were manually set based on initial experiments. However, further improvements could be achieved using automated search methods such as grid search, random search, or Bayesian optimization.(Bergstra & Bengio, 2012; Snoek et al., 2012)

In conclusion, hyperparameters significantly influence the performance of machine learning models. Proper tuning ensures that the trained models can make accurate predictions for new ramjet inlet designs while maintaining good generalization on unseen data.

2.4.Ramjet engine

A ramjet engine is a special type of air-breathing propulsion system designed to operate exclusively at supersonic speeds. Unlike turbojets or turbofans, a ramjet contains no rotating components such as compressors or turbines.(Anderson, 2010) Instead, it relies entirely on the high-speed forward motion of the vehicle to compress the incoming air through aerodynamic means, primarily via shock waves formed at the inlet. This process—known as ram compression—is only effective when the inlet Mach number exceeds one ($M > 1$), meaning ramjets cannot produce static thrust and require an initial boost (e.g., from a rocket or another engine) to reach operational conditions. Once at sufficient speed, the engine performs compression, combustion, and expansion through a series of internal flow processes that follow a modified Brayton cycle.

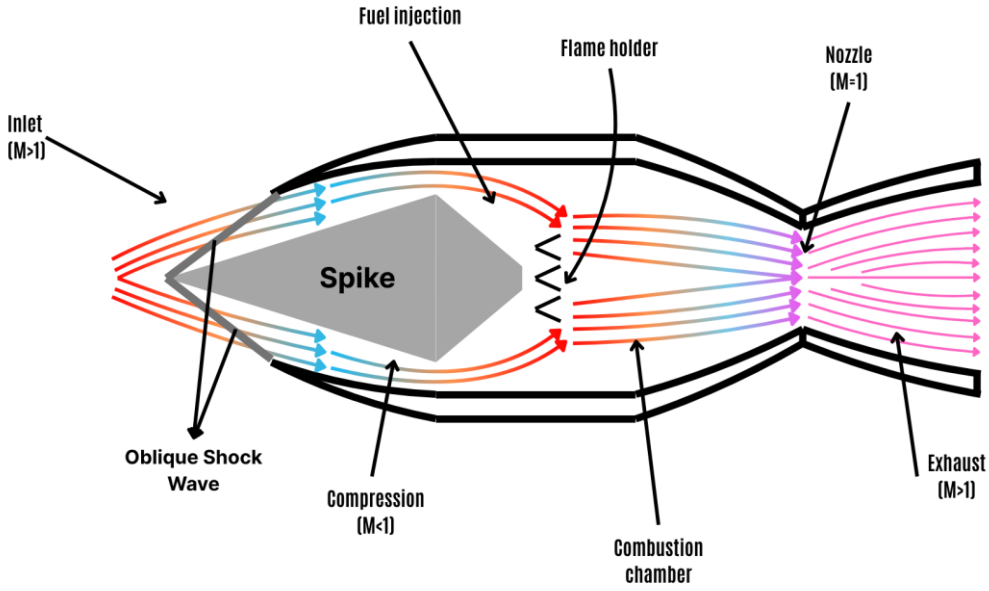

Figure 2.9: Basic Structure of Ramjet Engine

A shock wave is around 200 nm in thickness, thin region within a flow field where the flow properties such as pressure, temperature, and density change abruptly and irreversibly. Shock waves form when the flow speed exceeds the local speed of sound (M > 1). Across a shock wave, entropy increases and the process is no longer isentropic, resulting in total pressure loss. There are two types of shock waves relevant to ramjet inlets:

Oblique shocks, inclined to the flow direction, are formed when the flow is turned into itself by surfaces like ramps or cones. The flow remains supersonic after passing through an oblique shock.

Normal shocks are perpendicular to the flow direction. They decelerate the flow to subsonic speeds and cause a significant loss in stagnation pressure.(Anderson, 2010; Mattingly, 2006)

2.4.1. Operation of Ramjet

Ramjet engines can be divided into 3 phases which are Inlet Compression, Heat addition and Thrust generation.

The operation of a ramjet engine can be conceptually divided into three main phases: *inlet compression*, *heat addition*, and *thrust generation*. Each of these phases plays

a critical role in the overall performance of the engine. In the inlet compression phase, the supersonic incoming air is decelerated and compressed using a series of shock waves. This increases the pressure and temperature of the air before combustion. In the heat addition phase, fuel is injected and burned at nearly constant pressure in the combustion chamber, significantly raising the stagnation temperature of the flow. Finally, in the thrust generation phase, the high-energy gases expand through a converging-diverging nozzle, accelerating the flow and producing thrust. The following sections discuss each of these phases in detail, highlighting the governing equations and physical principles.

2.4.1.1. Inlet Compression: Oblique and Normal Shocks

Ramjet inlets typically employ a conical centerbody to generate oblique shock waves that gradually compress and decelerate the incoming air. The relationship between the cone half-angle θ, shock wave angle β, and the freestream Mach number M_1 is governed by conical flow theory and the Taylor–Maccoll equation. An engineering approximation for attached shocks is:

$$\theta = \tan^{-1}\left[\frac{2\cot\beta\,(M_1^2\sin^2\beta - 1)}{M_1^2(\gamma + \cos 2\beta) + 2}\right] \tag{2.5}$$

Here, γ is the specific heat ratio of air (approximately 1.4). For a given M_1, a higher cone angle θ increases β, strengthening the shock and the compression but also increasing total pressure loss. If θ exceeds a critical value, the shock detaches and becomes a bow shock, reducing pressure recovery. The total pressure ratio across a oblique shock is:

$$\frac{P_{0_2}}{P_{0_1}} = \left[\frac{(\gamma+1)M_1^2\sin^2\beta}{(2+(\gamma-1)M_1^2\sin^2\beta)}\right]^{\frac{\gamma}{\gamma-1}} \cdot \left[\frac{\gamma+1}{2\gamma M_1^2\sin^2\beta - (\gamma-1)}\right]^{\frac{1}{\gamma-1}} \tag{2.6}$$

If necessary, a normal shock may appear downstream to fully decelerate the flow to subsonic conditions. The total pressure ratio across a normal shock is(Anderson, 2021; Liepmann & Roshko, 1957):

$$\frac{P_{0_2}}{P_{0_1}} = \left[\frac{(\gamma+1)M_1^2}{(2+(\gamma-1)M_1^2)}\right]^{\frac{\gamma}{\gamma-1}} \cdot \left[\frac{\gamma+1}{2\gamma M_1^2-(\gamma-1)}\right]^{\frac{1}{\gamma-1}} \quad \textbf{(2.7)}$$

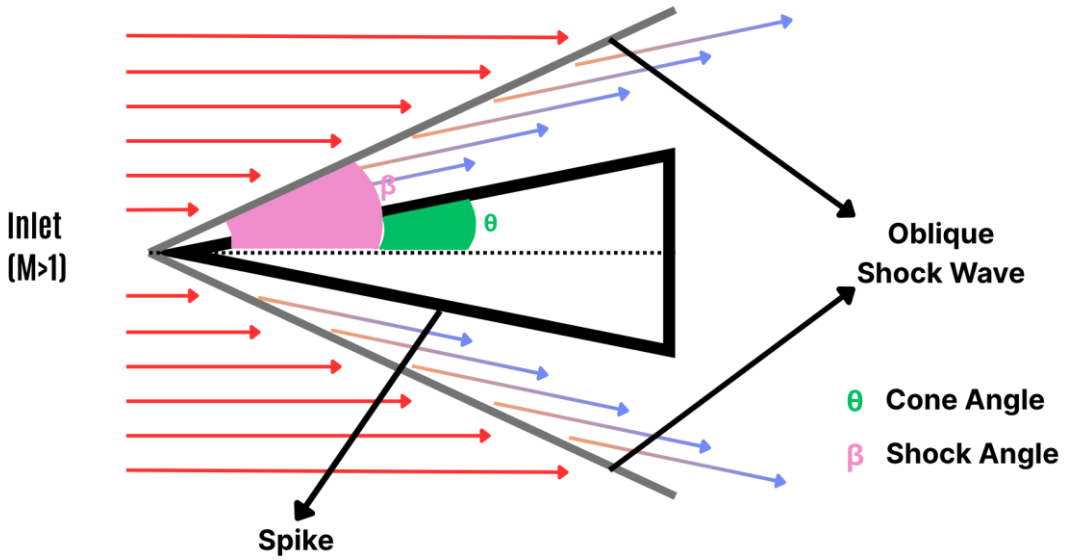

Figure 2.10: Illustration of Ramjet Spike

2.4.1.2. Heat Addition in the Combustion Chamber

Following compression, fuel is injected into the combustion chamber where it burns at approximately constant pressure. The resulting heat addition increases the stagnation temperature of the flow. This process is modeled as:

$$T_{0_3} = T_{0_2} + \frac{\eta_c \cdot f \cdot \mathrm{LHV}}{c_p} \quad \textbf{(2.8)}$$

where f is the fuel-to-air ratio, LHV is the lower heating value of the fuel, c_p is the specific heat at constant pressure, and η_c is the combustion efficiency. In practice, a small stagnation pressure loss is also considered (typically 3–8%)(Hill & Peterson, 1992; Mattingly, 2006).

2.4.1.3. Expansion and Thrust Generation

The high-temperature gas is then expanded through a converging-diverging nozzle, converting thermal energy into kinetic energy. The nozzle exit Mach number and area are related by the area–Mach number equation:

$$\frac{A}{A^*} = \frac{1}{M}\left[\frac{2}{\gamma+1}\left(1+\frac{\gamma-1}{2}M^2\right)\right]^{\frac{\gamma+1}{2(\gamma-1)}} \quad \textbf{(2.9)}$$

The exit velocity assuming isentropic expansion is(Anderson, 2021):

$$V_e = \sqrt{2c_p T_{0_3}\left(1-\left(\frac{P_e}{P_{0_3}}\right)^{\frac{\gamma-1}{\gamma}}\right)} \quad \textbf{(2.10)}$$

The net thrust is determined by(Sutton & Biblarz, 2010):

$$F = \dot{m}_a V_e + (P_e - P_0)A_e - \dot{m}_a V_0 \quad \textbf{(2.11)}$$

2.4.2. Performance Metrics

Several key parameters are used to evaluate the performance of a ramjet engine. These metrics provide insight into how efficiently the engine converts fuel energy into thrust and how effectively it utilizes the high-speed incoming air.(Mattingly, 2006)

2.4.2.1. Specific Thrust:

$$\frac{F}{\dot{m}_a} = V_e - V_0 + \frac{(P_e - P_0)A_e}{\dot{m}_a} \quad \textbf{(2.12)}$$

Specific thrust is defined as the thrust produced per unit mass flow rate of air entering the engine. It consists of two components: the change in momentum between the exit and freestream velocities $(V_e - V_0)$, and a pressure thrust term accounting for the difference between exit and ambient pressure.

2.4.2.2. Thermal Efficiency:

Thermal efficiency quantifies how much of the chemical energy from the fuel (based on fuel-to-air ratio f and lower heating value, LHV) is converted into useful kinetic energy of the exhaust flow. It depends on the velocity gain across the engine.

$$\eta_{thermal} = \frac{V_e^2 - V_0^2}{2f \cdot \mathrm{LHV}} \tag{2.13}$$

2.4.2.3. Propulsive Efficiency:

Propulsive efficiency measures how effectively the engine converts the kinetic energy of the exhaust jet into thrust. Maximum efficiency occurs when the exhaust velocity V_e is close to the freestream velocity V_0, minimizing wasted energy.

$$\eta_{\mathrm{prop}} = \frac{2V_0}{V_e + V_0} \tag{2.14}$$

2.4.2.4. Overall Efficiency:

Overall efficiency is the product of thermal and propulsive efficiencies.(Hill & Peterson, 1992) It represents the total efficiency of converting fuel energy into thrust-producing work. The design of a high-performance ramjet requires careful management of shock wave strength, total pressure losses across shocks and the combustor, and structural limitations. Balancing these factors ensures that the engine operates efficiently over a desired supersonic flight regime.

$$\eta_{\mathrm{overall}} = \eta_{\mathrm{thermal}} \cdot \eta_{\mathrm{prop}} \tag{2.15}$$

3. PROCESS

To achieve the goal of this study, the aerodynamic design of the ramjet inlet had to be completed first, allowing sufficient CFD data to be generated for proper training of the machine learning models. During the design process, several software tools were used to

improve and adjust the initial inlet design. These tools helped shape the geometry, run the analysis, and analyze the performance of the inlet section more effectively.

The design stage of the ramjet inlet must be performed using Gmsh, SU2, Paraview programs in the specified order.

After using the specified programs in order, the machine learning stage was conducted with the extracted data.

3.1.Gmsh

Gmsh is simply a mesh generator which has a CAD engine in it. With the user-friendly 3D finite element meshing, users can make some processes and visualize the results of their designs using post processors. Due to "light program" aim, developers made it usable with its own language, C++ or Python. This open-source tool requires parametric inputs to create geometry and continue for meshing process.(Geuzaine & Remacle, 2009; *Gmsh: A Three-Dimensional Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities*, n.d.)

These are the main components contained in this program:

- Geometry
- Mesh
- Solver
- Post-Processing

To run the CFD analysis of the Ramjet inlet, there must be a design and a meshing result in .su2 format. By using Gmsh, this geometry is drawn according to the theoretical calculations made earlier. The example calculations of oblique and normal shocks made for the ramjet inlet with these parameters:

- $12°$ theta angle for initial ramp
- $0.2\ m$ for each ramp
- 3 ramps inlet

Theoretical calculations help to determine the point coordinates of those ramps. It gives the coordinates of two end points for each line which are named as ramps. For this example design, three lines (ramps) connected each other to form the ramps of ramjet. The result of

theoretical calculations gives the point coordinates of the four points of these three connected lines and a focal point according to these ramps. 12° cone angle(θ) used for the calculation of first ramp's slope; then defined 12 degree cone angle used for calculating shock angle(β) for determining focal point of oblique shocks. After determining the coordinates of five critical points for design stage other required points need to be determined to draw a complete inlet geometry and create a closed surface. The closed geometry given is drawn, using these line identities: inlet, wall, outlet, far field and axis. These IDs are given in the .geo file as:

```
Physical Curve("inlet") = {21};

Physical Curve("axis") = {1};

Physical Curve("wall") = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18};

Physical Curve("outlet") = {9};

Physical Curve("far_field") = {19, 20};
```

Generating 3D shape the Ramjet inlet will be done by revolving the 2D shape about the axis "axis".

The first point of this design started from the bottom left of the Figure 11 and continued to draw on the Gmsh in a counterclockwise direction. There is an inclination between the points 1-5 because it is aimed to create three oblique shocks and one normal shock between points 1-5 using these inclined three ramps. To run the Ramjet engine properly, air needs to be compressed by these shocks to lower the velocity below Mach number of 1 and sent to the combustion chamber to ignite the fuel which comes from injectors. Increasing the pressure of the air which passed through shocks will help to increase the temperature of air and ignite the fuel better. After air which passed normal shock, air speed become subsonic. For subsonic speeds increasing cross-sectional area will reduce speed and increase pressure. To achieve this, the decline of the lines between points 5-9 is designed while maintaining the "outlet" line set to be twice as throat length ($y_{14} - y_5$) in 2D design. Using straight lines preferred by considering manufacturing process. From a manufacturing aspect of view, avoiding the complex shapes, using lines rather than concave or convex shapes make manufacturing a lot easier and cheaper. Production tolerances must be strict to have a better-functioning nose cone, using simple shapes will allow for the manufacturers to comply with requirements.

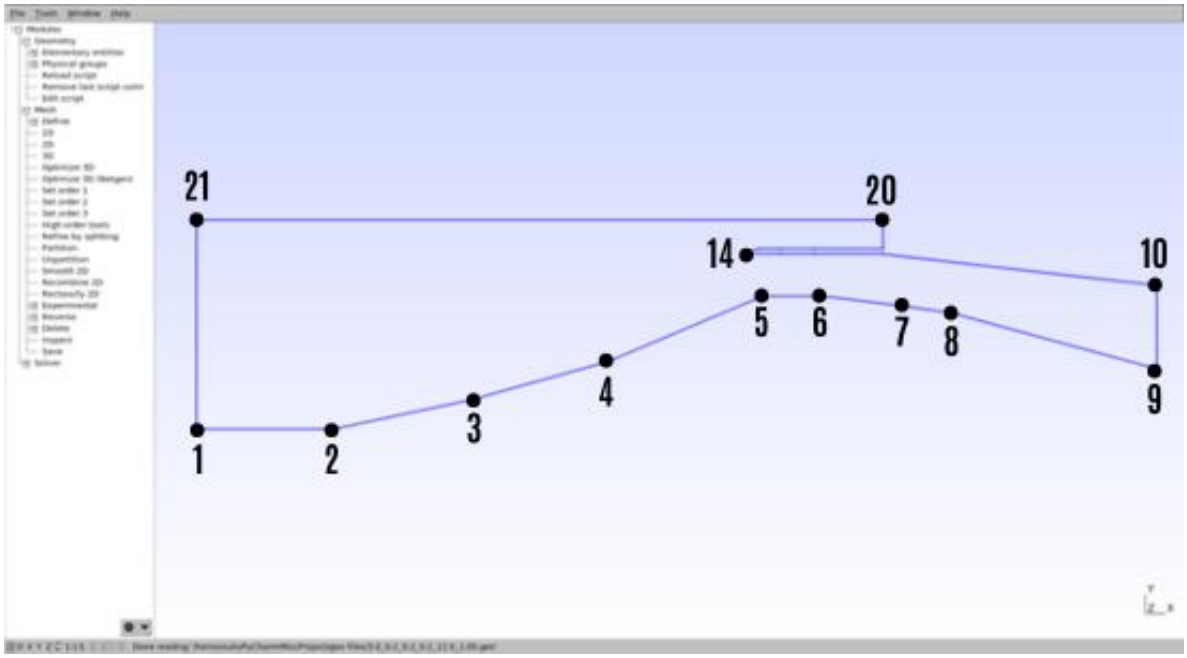

Figure 3.1: Closed Surface Drawn on Gmsh

There are 21 points and 21 lines drawn between these points in this design. This closed shape forms a surface which will be meshed, and it is like a one-half cross-section of the Ramjet inlet. The surface of meshing and the closed shape is defined as:

```
Curve Loop(1) = {20, 21, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19};

Plane Surface(1) = {1};
```

As shown in Figure 12 the meshed surface has different mesh density around the surface. The lines identified as “wall” and “outlet” designed to have more partition to capture shock effects and outlet values better. Also, the mesh size is adjusted according to where the mesh needed more.

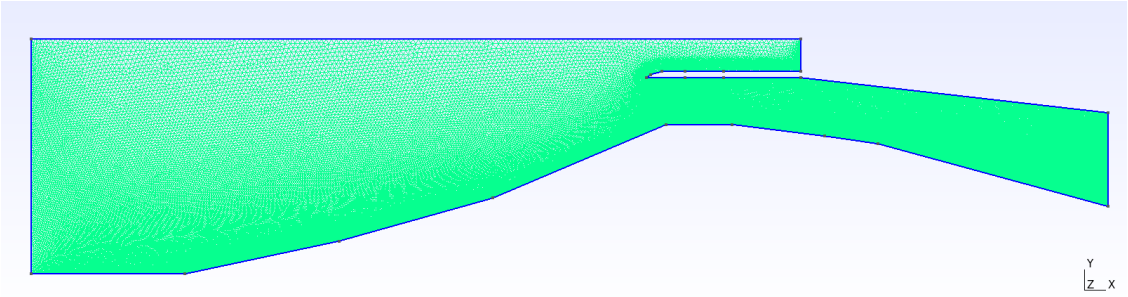

Figure 3.2: 2D Meshed Result of The Inlet Geometry

These are the parts of those codes which determine the number of splits on a line:

```
Transfinite Curve {10} = 252 Using Progression 1;

Transfinite Curve {16} = 30 Using Progression 1;
```

It defines that the line with ID 10 is divided into 252 equal parts, and line 16 divided into 30 parts. By increasing these values, the mesh quality can be increased but the CFD running time will be much higher. The meshing settings in .geo files are significant as much as the shape of the design of ramjet inlet because misusage of settings the CFD can diverge (goes to infinite values) the analysis.

```
Mesh.Smoothing = 10;  // Apply 10 smoothing iterations

Mesh.Optimize = 1;     // Enable basic mesh optimization
```

By using these settings, Gmsh will apply 10 iterations of a Laplacian smoothing on the mesh, and this will reposition the nodes closer considering their neighbor's centroids. This setting will lower the number of skewed shapes which are produced by meshing algorithms. By minimizing the distortions, it removes the poorly shaped triangles in the mesh. Optimize the mesh by removing the negative Jacobians and stretched geometries.(Geuzaine & Remacle, 2009)

Boundary layers help the CFD solver to capture the near wall properties like temperature, velocity, friction, pressure etc. better with creating structured mesh near required surfaces. Also the turbulence model needs to function better near to the walls to simulate the CFD analysis as close as physical environment.

In the .geo file boundary layer defined as:

```
Field[1] = BoundaryLayer;

Field[1].EdgesList = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18};

Field[1].hwall_n = 0.0001;

Field[1].thickness = 0.001;

Field[1].hfar = 0.008;

Field[1].ratio = 1.2;

Field[1].IntersectMetrics = 0;

Field[1].Quads = 1;

BoundaryLayer Field = 1;
```

The boundary layer creates the first layer 0.0001 m above the surface, growth factor between layers is 1.2 and the total boundary layer will be 0.001 m. It was sufficient to capture the properties of the example design.

Also, above the boundary layer there needs to be another setting which determines additional meshing parameters for the regions with critical flow. Figure 13 below shows the different mesh intensity around different regions.

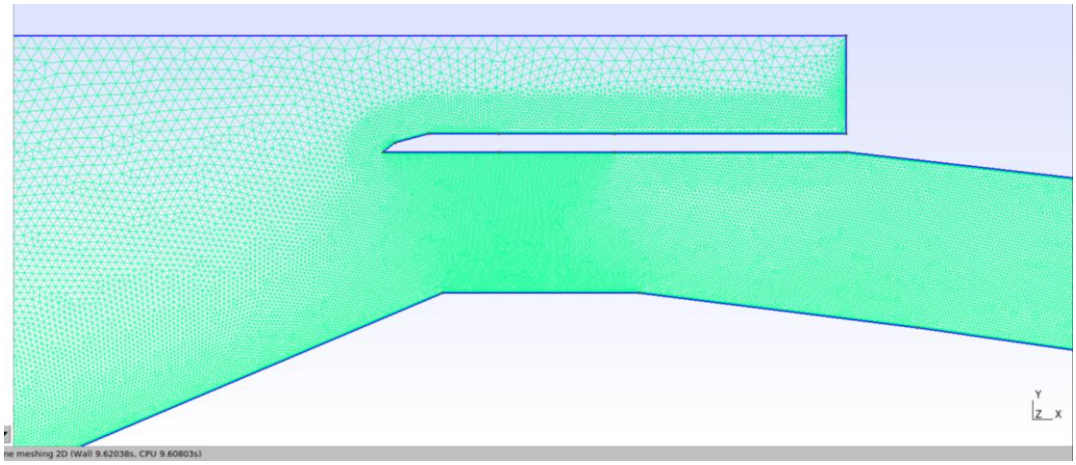

Figure 3.3: Regional Mesh Quality Differences

```
Define the Background Mesh Size Field using a Distance field for key features

Field[2] = Distance;

Field[2].EdgesList = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18}; // specify the curves near critical features


// Use a Threshold field to vary the mesh size based on the distance from Field[2]

Field[3] = Threshold;

Field[3].IField = 2;      // Use the distance from Field[2]

Field[3].LcMin = 0.0015;   // Fine mesh size near key features

Field[3].LcMax = 0.008;    // Coarse mesh size far from key features

Field[3].DistMin = 0.015193667097058203;   // Below this distance, use LcMin

Field[3].DistMax = 0.030387334194116405;   // Above this distance, use LcMax
```

Fine and coarse mesh size near key feature is determined using the feedback gathered from CFD analysis made earlier. For the designing stage of the template design, it is seen as enough due to moderate CFD running time, acceptable and converged CFD outputs.

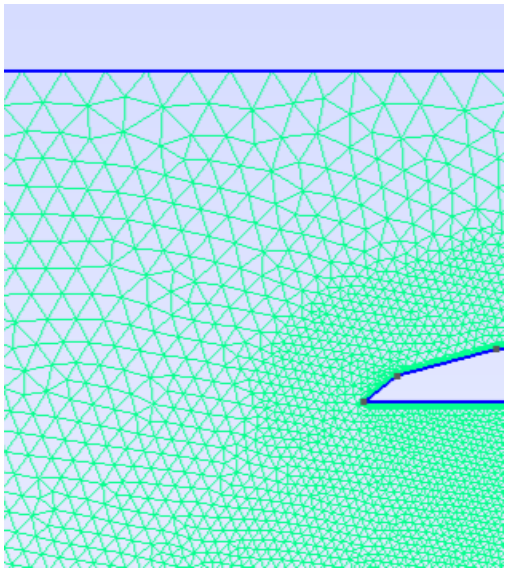

Figure 3.4: Triangular Mesh Across the Surface

Field[3].DistMin, Field[3].DistMax values are calculated by the program that developed. It calculates these values with the throat length calculated by the program itself. For each different design, these values will be recalculated and will be unique.

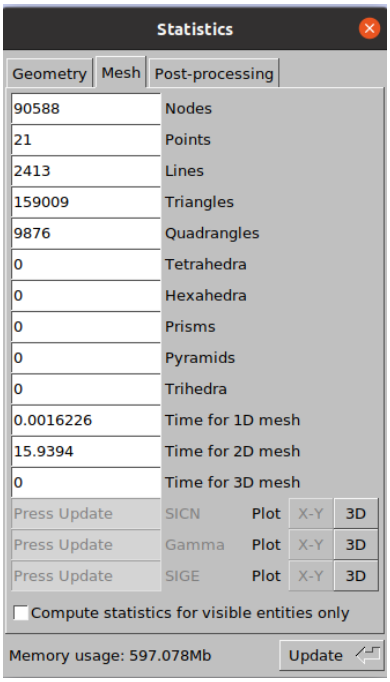

Figure 3.5: Mesh Statistics

"**BoundaryLayer**" and the "**Distance**" both defines the mesh sizes for the regions required to increase the mesh quality. Using both codes can be unnecessary because defining two different mesh quality for same region can create confusion for the Gmsh and program can only apply the last code that defines the mesh quality. Avoid this miscalculation the "**Min**" combines both sizing functions and select the minimum mesh size as background mesh.

```
// 7. Combine All Fields Using the Min Operator

Field[4] = Min;

Field[4].FieldsList = {1, 3}; // take the smallest mesh size computed by these fields

Background Field = 4;  // Set Field[7] as the background mesh size field
```

Mesh Quality Metric	Minimum	Maximum
Orthogonality Angle (deg.)	11.0343	90
CV Face Area Aspect Ratio	1.00005	40.1123
CV Sub-Volume Ratio	1	33.0732

Figure 3.6: Mesh Quality Metrics Calculated on SU2

After finishing the meshing process, the file of the meshed geometry must be saved with extension of ".su2". This file is needed for the SU2 CFD solving stage of ramjet inlet design.

3.2.SU2

SU2 is an open source CFD program which is contributed by developers around the globe. With the built in C++ modules, it can provide Partial Differential Equation (PDE) analysis for several areas of use: aerodynamic shape optimization, potential flow, elasticity etc. Users can work on the project based on Partial Differential Equations and developers continuously adding new updates and modules.(Economon et al., 2016a)

The second and most important part of the design stage of Ramjet inlet is to simulate the designed geometry with targeted environmental properties. Without creating a working premise ramjet inlet in real life, the CFD simulations can be done with much lower expenditure.

SU2 version 8.0.1 Harrier was used for this project. (SU2 Development Team, 2024)

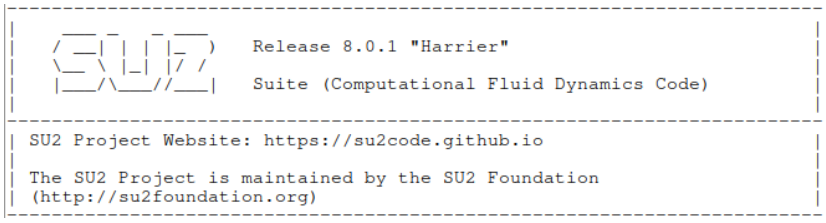

Figure 3.7: Version of SU2 Used In This Project

The saved mesh in ".su2" extension needs to be simulated using the SU2 CFD solver. To be able to gather data from the CFD results it is important to adjust the correct settings of the program in the configuration file saved with .cfg extension. The code below points out the .su2 mesh file for the calculations:

```
%-------------------- SU2 CFD Configuration --------------------

MESH_FILENAME= 3.0_0.2_0.2_0.2_12.0_1.05.su2
```

After defining the required mesh, the solver setting must be done. The solver option required to be carefully chosen to simulate the design of Ramjet inlet in acceptable physical environment.

3.2.1. Solvers

There are several other types of solvers that can be used in SU2, these can be used for different types of situations and these solvers are:

EULER: It is the simplified version of compressible Navier-Stokes and with the lack of ability to perform viscous and thermal conducting environment it was not accepted to be the solver. This solver would be enough to see the shock properties of early designs without considering the boundary play a significant role.(Anderson, 20; Economon et al., 2016a)

NAVIER_STOKES: Excluding the Reynolds stresses in the flow, it treats it as laminar compressible flow. It includes molecular viscosity and thermal conduction but disregard the turbulence mixing in the flow so the solution will be laminar and give manipulated skin friction values.(Blazek, 2015; Hirsch, 2007)

RANS: It is the selected solver option for the Ramjet inlet CFD analysis because of the high Reynold number and need for the capturing turbulence effects in the system. Also, the boundary layer properties of the flow need to be same as the real physical environment so

the surface drag, flow regime on the film and turbulence effects could be calculated with lower errors. Without the viscous flows and the turbulence in the system it is not reliable to initiate any design process with solved CFD analysis.(Economon et al., 2016a; Spalart, 2000)

SOLVER= RANS

This is the usage of the code in the .cfg file.

```
----------------- Physical Case Definition ( Zone 0 ) -------------------
Compressible RANS equations.
Turbulence model: Menter's k-omega SST-2003m with no production modification.
Hybrid RANS/LES: No Hybrid RANS/LES
Mach number: 3.
Angle of attack (AoA): 0 deg, and angle of sideslip (AoS): 0 deg.
Reynolds number: 2.54452e+07. Reference length 1.
```

Figure 3.8: CFD Solver Settings

3.2.2. Turbulence Models

For the different flow regimes and simulations goals there should or shouldn't be turbulence models in the simulations. The SU2 offers different types of models that can be acceptable for Ramjet analysis. A suitable model is decided according to project expectations. In this project it is required to select the model which has the ability to capture shock effects and near wall properties close to real life results. Some of these models are:

NONE: It computes the flow as laminar and do not include any turbulence model, it can be acceptable for the preliminary designs with low Reynolds numbers. Treating the turbulence viscosity as zero makes this model useless for the flow separations and complex vortex flows.(Hirsch, 2007)

SA: Is a one equation model which means it solves using one variable to compute the turbulence in the system. It provides robust computational performance and quick results for the CFD, but it can give deficient results in a system which has flow separations and re-attachment. For the flows with low pressure gradients this model can be acceptable at the stage of initial shape optimization.(SPALART & ALLMARAS, n.d.)

SST: Is a two-equation model developed by Menter which synthesizes the k-ω turbulence model near walls to capture better wall properties and the k-ε model away from walls to include complex turbulence effects. As is mentioned, two equations, these are turbulent kinetic energy k and specific dissipation rate ω. By including these in the computation, it

increases the ability of processing high pressure gradients and separated turbulent flows better than other models. This model is useful for systems with high Reynolds which require precise boundary layer separation calculations. It can capture multiple shock waves and the highly compressed flows near walls with high accuracy but while having these advantages it also has limitations. It requires designs with high mesh quality to have better near wall calculations, because of this reason the computational time to finalize the CFD increases. It conducts more expensive simulations. (Economon et al., 2016b; Hellsten, 2004)

The example usage of the code in .cfg file is:

```
KIND_TURB_MODEL= SST
```

After defining these parameters, the settings are adjusted based on the example CFD simulation. The analysis requires specifying a fluid type, and since the ramjet operates at Mach 3 at an altitude of 10 km, the fluid properties, freestream conditions, and Reynolds number are calculated using the ISA model. Additionally, all computations are performed using SI units.

```
%------------ DIRECT, ADJOINT, AND LINEARIZED PROBLEM DEFINITION ------------%

FLUID_MODEL= STANDARD_AIR

SYSTEM_MEASUREMENTS= SI

MACH_NUMBER= 3.0

REYNOLDS_NUMBER= 25445174    %% Approx. for 10 km altitude & L = 1 m

%% Standard Atmosphere at 10000 m

FREESTREAM_PRESSURE= 26435    %% Pa

FREESTREAM_TEMPERATURE= 223.1  %% K
```

```
-- Initial and free-stream conditions:
+-------------------------------------------------------------------------------+
|                   Name|     Dim. value|     Ref. value|       Unit|Non-dim. value|
+-------------------------------------------------------------------------------+
|        Static Pressure|        26426.9|              1|         Pa|        26426.9|
|                Density|       0.412645|              1|     kg/m^3|       0.412645|
|            Temperature|          223.1|              1|          K|          223.1|
|           Total Energy|         566601|              1|   m^2/s^2|         566601|
|             Velocity-X|        898.297|              1|        m/s|        898.297|
|             Velocity-Y|              0|              1|        m/s|              0|
|    Velocity Magnitude|        898.297|              1|        m/s|        898.297|
+-------------------------------------------------------------------------------+
|              Viscosity|    1.45677e-05|              1|   N.s/m^2|    1.45677e-05|
|         Conductivity|              -|              1|   W/m^2.K|              -|
|     Turb. Kin. Energy|        3026.01|              1|   m^2/s^2|        3026.01|
|     Spec. Dissipation|     8.5715e+06|              1|        1/s|     8.5715e+06|
+-------------------------------------------------------------------------------+
|           Mach Number|              -|              -|          -|              3|
|       Reynolds Number|              -|              -|          -|    2.54452e+07|
```

Figure 3.9: Initial and Free-Stream Conditions for CFD On SU2

Also, the boundaries of the design and the inlet, outlet, far field sections of the meshed 2D surface must be defied including their pressure values to create a proper flow criterion. These are defined in .cfg for the example simulation as:

```
%% ------------------- BOUNDARY CONDITION DEFINITION -------------------------%%

MARKER_HEATFLUX= ( wall,0 )

MARKER_FAR= ( far_field,26435)

MARKER_SUPERSONIC_INLET= ( inlet, 223.1, 26435, 898.4, 0.0, 0.0 )

MARKER_OUTLET= ( outlet, 369926.113 )

MARKER_SUPERSONIC_OUTLET= ( far_field )

AXISYMMETRIC= YES

MARKER_SYM= (axis)
```

```
------------------- Geometry Preprocessing ( Zone 0 ) -------------------
Two dimensional problem.
96640 grid points before partitioning.
180989 volume elements before partitioning.
5 surface markers.
100 boundary elements in index 0 (Marker = inlet).
150 boundary elements in index 1 (Marker = axis).
1646 boundary elements in index 2 (Marker = wall).
298 boundary elements in index 3 (Marker = outlet).
219 boundary elements in index 4 (Marker = far_field).
```

Figure 3.10: Defined Boundaries In .cfg File

These values in the code are calculated by the developed program for Ramjet design. As is given above, the inlet defined as MARKER_SUPERSONIC_INLET, the outlet as MARKER_OUTLET and the far field as MARKER_SUPERSONIC_OUTLET. The inlet requires the temperature, pressure, velocity parameters; the outlet and the far field only needs pressure to run the code. The reaction of fluid and strength of oblique and normal shocks differs in conical shape, to obtain realistic result with less computer power CFD analysis solved axisymmetric. The 2D shape of surface revolved about the line defined as “axis”. The Figure 21 below expresses the how those boundaries defined in SU2 configuration file.

```
------------- Config File Boundary Information ( Zone 0 ) ---------------
+-------------------------------------------------------------------------+
|                         Marker Type|                        Marker Name|
+-------------------------------------------------------------------------+
|                          Far-field|                          far_field|
|                                   |                              26435|
+-------------------------------------------------------------------------+
|                      Symmetry plane|                               axis|
+-------------------------------------------------------------------------+
|          Supersonic inlet boundary|                              inlet|
+-------------------------------------------------------------------------+
|         Supersonic outlet boundary|                          far_field|
+-------------------------------------------------------------------------+
|                     Outlet boundary|                             outlet|
+-------------------------------------------------------------------------+
|                      Heat flux wall|                               wall|
+-------------------------------------------------------------------------+
```

Figure 3.11: Marker Type and Marker Names on SU2

3.2.3. Viscosity Model

Viscosity is a significant parameter to consider while running analysis with boundary-layer development, shock-boundary-layer interaction and heat transfer play critical role. Viscosity model determines how the SU2 must handle the viscosity calculations, with different type of viscosity models can be selected for Ramjet inlet design. These options are:

CONSTANT: This means that SU2 will use a fixed value for the dynamic viscosity whether temperature changes the properties of fluid or not, so this gives overpredicted or underpredicted results. That is why this is suitable for isothermal flows.(Anderson, 20)

SUTHERLAND: This law is a relationship extensively applied to precisely estimate the temperature dependence of dynamic viscosity. The shock waves create a lot of temperature difference between the nose cone and the throat of the Ramjet so this variation in temperature must influence the viscosity of the fluid. Behavior of the particles in air cannot be considered as constant because it leads to incorrect modeling of wall shear stresses, boundary layer thickness, and heat transfer rates. That is the reason SUTHERLAND selected. (Blazek, 2015; Economon et al., 2016b)

```
%% ---------------------------- VISCOSITY MODEL ---------------------------------%%

VISCOSITY_MODEL= SUTHERLAND

MU_REF= 1.716E-5

MU_T_REF= 273.15

SUTHERLAND_CONSTANT= 110.4
```

-- Models:

Viscosity Model	Conductivity Model	Fluid Model
SUTHERLAND	CONSTANT_PRANDTL	STANDARD_AIR

-- Fluid properties:

Name	Dim. value	Ref. value	Unit	Non-dim. value
Ref. Viscosity	1.716e-05	1	N.s/m^2	1.716e-05
Sutherland Temp.	273.15	1	K	273.15
Sutherland Const.	110.4	1	K	110.4
Prandtl (Lam.)	-	-	-	0.72
Prandtl (Turb.)	-	-	-	0.9
Gas Constant	287.058	1	N.m/kg.K	287.058
Spec. Heat Ratio	-	-	-	1.4

Figure 3.12: Fluid Properties and Model Settings

3.2.4. Convective Schemes

Convective schemes are needed to calculate how fluid properties, for example mass, momentum continue across the boundaries of grid cells in simulations for the Finite Volume Method (FVM). It has several options in SU2 to define suitable methods, these are:

JST (Jameson-Schmidt-Turkel Scheme): This scheme is a one of the central schemes, it has added artificial dissipation to have steady results, but this is regulated to control oscillations near shocks. Because of this reason it can lower reliability.(Blazek, 2015; Jameson et al., 1981)

HLLC (Harten-Lax-van Leer-Contact): Is a type of upwind method which precisely handles contact discontinuities between different materials, and shear waves where the flow layers move in different velocities. (Economon et al., 2016b)

ROE (Roe's Approximate Riemann Solver): The Roe scheme is an upwind method which is good at capturing shocks and contact discontinuities. It linearize the flux Jacobian, as it is mentioned in the name of the scheme, it approximates to the Riemann problem. It has precise shock capturing ability with lower computational work. That is the reason ROE selected for Ramjet CFD. (Roe, 1981)

```
%% -------------------- FLOW NUMERICAL METHOD DEFINITION ----------------------%%

CONV_NUM_METHOD_FLOW= ROE

MUSCL_FLOW= NO

SLOPE_LIMITER_FLOW= VENKATAKRISHNAN

LAX_SENSOR_COEFF= 0.15
```

JST_SENSOR_COEFF= (0.5, 0.02)

TIME_DISCRE_FLOW= EULER_IMPLICIT

Convergence criteria are important for managing time required for overall CFD process, if the criteria is so strict then the need for better devices to solve those mathematical equations increases. For the example CFD analysis several simulations conducted using different converge criteria, the setting below considered acceptable.

%% ---------- CONVERGENCE PARAMETERS ----------%%

CONV_FIELD= RMS_DENSITY

CONV_RESIDUAL_MINVAL= -5

CONV_STARTITER= 100

CONV_CAUCHY_ELEMS= 100

CONV_CAUCHY_EPS= 1E-7

At the end of the .cfg file there must be code for what type of data needs to be saved in history files. For the Ramjet inlet CFD analysis, it is important to have temperature, pressure, velocity parameters to be saved in .csv files for future calculations. The code below express that these data will be saved in .csv format.

TABULAR_FORMAT= CSV

OUTPUT_FILES= (CSV,SURFACE_CSV, PARAVIEW, SURFACE_PARAVIEW)

HISTORY_OUTPUT= (ITER, RMS_RES,
SURFACE_MASSFLOW,SURFACE_TOTAL_PRESSURE,SURFACE_TOTAL_TEMPERATURE,SURFACE_MACH)

CONV_FILENAME= 3.0_0.2_0.2_0.2_12.0_1.05_history.csv

	A	B	C	D	E	F	G	H	I	J	K	L
1	Time_Iter	Outer_Iter	Inner_Iter	rms[Rho]	rms[RhoU]	rms[RhoV]	rms[RhoE]	rms[k]	rms[w]	Avg_Massflow	Avg_Mach	Avg_TotalTemp
2	0	0	0	-1,9018631	-1,737682021	-14,40345858	3,897921354	3,688348897	7,10430794	-41,85990642	3,000195436	624,6649485
3	0	0	5	-2,13955808	0,6853484418	0,5950011472	3,689428643	1,743019447	5,1019301	-42,257999	2,944836966	631,2505212
4	0	0	10	-2,25407893	0,6186799214	0,4857605635	3,63048483	1,447061201	4,25199195	-41,50739169	2,870383485	662,1282171
5	0	0	15	-2,38138718	0,5284534197	0,3265121872	3,602852558	1,458503835	3,99292902	-39,37464369	2,765935978	725,778261
6	0	0	20	-2,39126253	0,5449761992	0,3386419492	3,679755288	1,519681303	3,88972883	-37,34372616	2,679255214	763,9341187
7	0	0	25	-2,36452625	0,5263249992	0,3897705399	3,682921785	1,519448678	3,77166193	-34,5769106	2,580633821	820,0667647
8	0	0	30	-2,32862305	0,5526316045	0,4269938347	3,735037692	1,933078201	12,3927267	-31,40320584	2,475651093	882,0908186
9	0	0	35	-2,28665923	0,5541880802	0,4826605475	3,764155252	1,630378126	17,0548716	-28,29516395	2,379723338	944,2412058
10	0	0	40	-2,22033241	0,589839681	0,5688306207	3,84844817	1,709716145	13,9728022	-25,28319521	2,299980414	1012,194481
11	0	0	45	-2,2026382	0,5928127928	0,6069017347	3,846006874	1,74294171	10,9491	-22,33986107	2,218761597	1084,160672

Figure 3.13: Computed Values From CFC and Changes According to Each Iterations

3.3.Paraview

Paraview is an open source, interactive analyzing program which is used for post processing applications. With its visualizations engine it is possible to work on large datasets using automatic batch processing.(*About ParaView*, n.d.)

After running CFD simulations those results must be evaluated and if needed there must be some changes applied to design or to solver settings. The figures below are the visualization of .vtu file which is a output file of SU2 CFD solver.

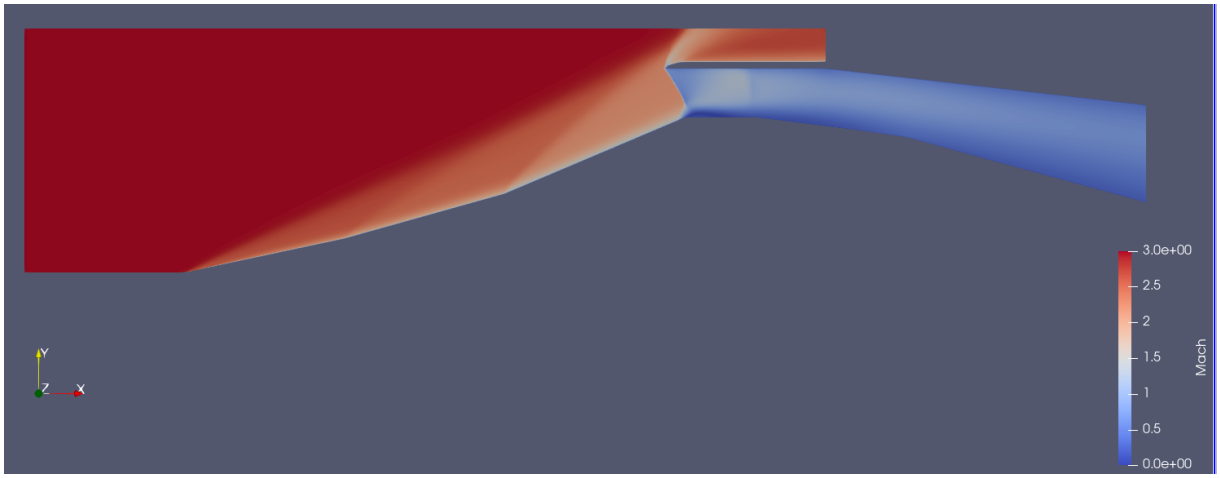

Figure 3.14: Contour Plot of the Ramjet Inlet Cross-Section for Varying Mach Numbers

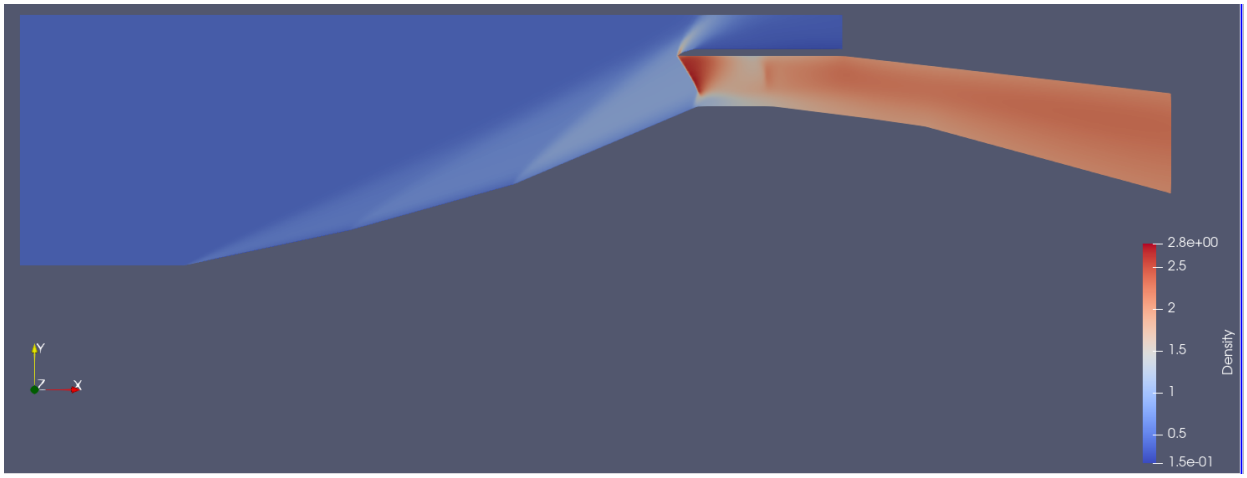

Figure 3.15: Contour Plot of the Ramjet Inlet Cross-Section for Varying Density Values

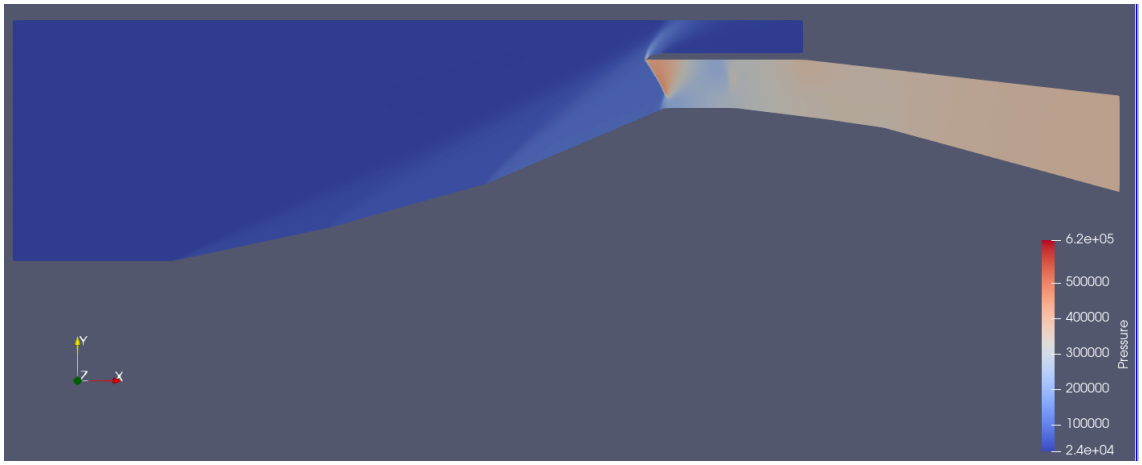

Figure 3.16: Contour Plot of the Ramjet Inlet Cross-Section for Varying Pressure Values

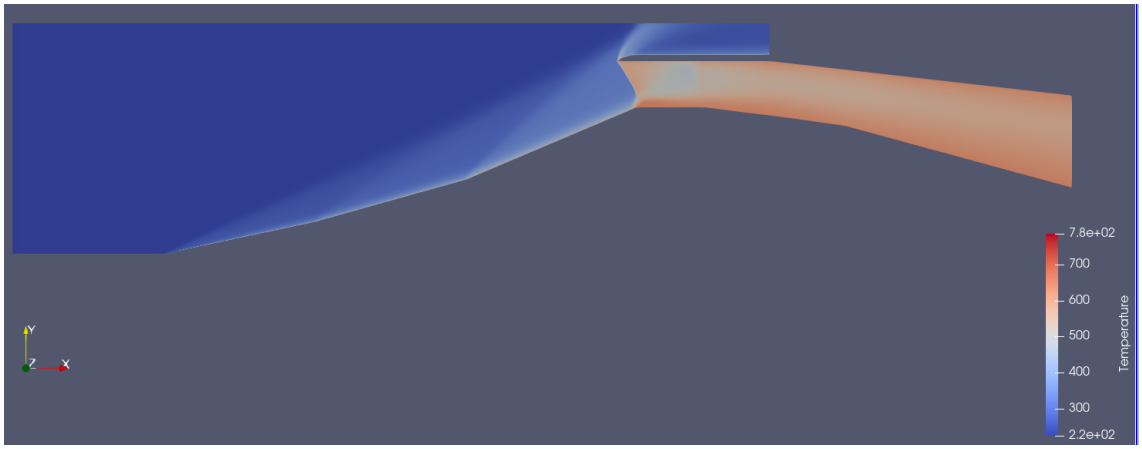

Figure 3.17: Contour Plot of the Ramjet Inlet Cross-Section for Varying Temperatures

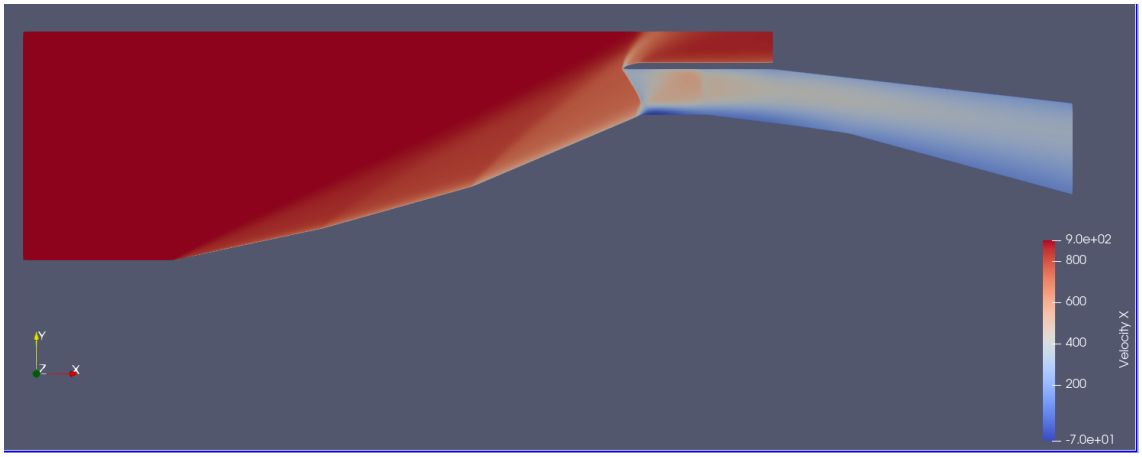

Figure 3.18: Contour Plot of the Ramjet Inlet Cross-Section for Velocity in X Axis

Due to sudden compression of air at the start of the Ramjet throat, pressure increases rapidly but after that there is an expansion wave. That wave decrease the pressure as much as it can.

Because supersonic waves on the inclined ramps helps to compress the air but reverse shape creates expansion of air. At the end of the channel, flow becomes subsonic and the expansion of throat increases the pressure of air. Figure 29 gives the line chart of pressure change across the channel.

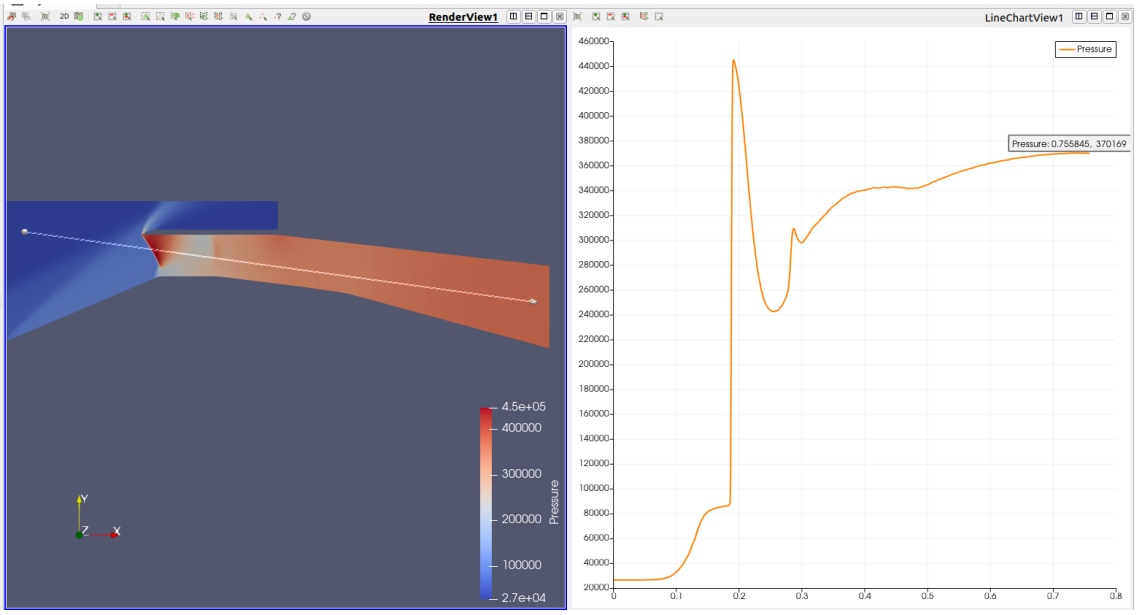

Figure 3.19: Contour Plot of the Ramjet Inlet and Line Chart of Pressure

As it was expressed in the pressure graph in Figure 30, the expansion wave of air increses the velocity of fluid passing the throat. That is the reason of sudden increase of mach after normal shock.

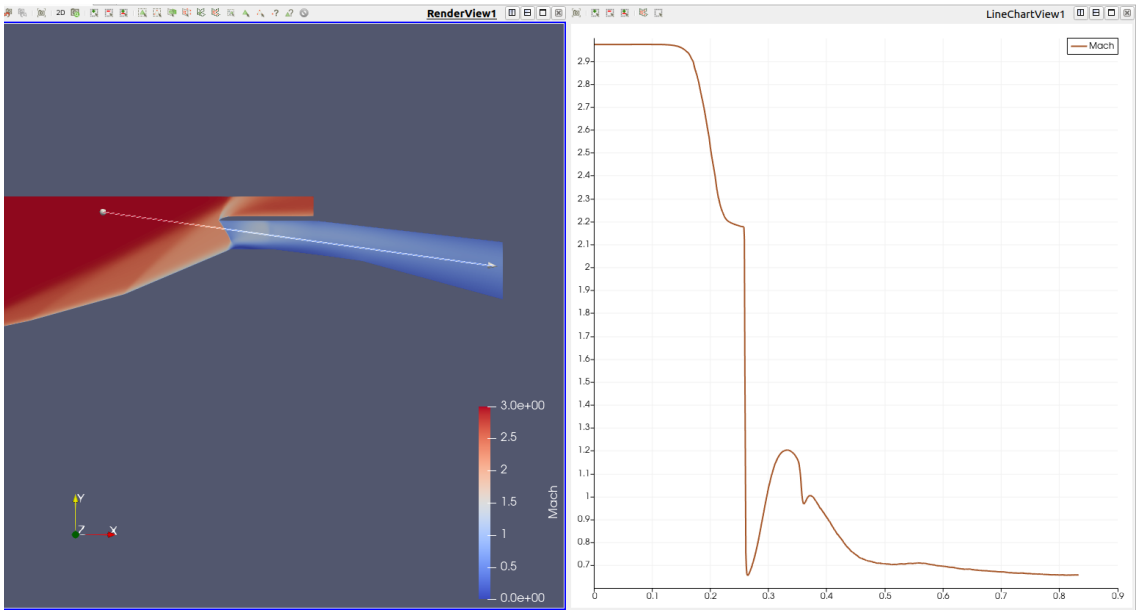

Figure 3.20: Contour Plot of the Ramjet Inlet and Line Chart of Mach Inside the Throat

These three ramps create three oblique shock waves and because of these waves air compressed three times. As shown on the graphs in Figure 31, shocks don't have the same strength. Stair-like shape of graph doesn't have any sudden increase in the first wave but at the third wave it became more powerful with rapid increase. As expected, the normal shock is the strongest among these shocks and decreases the flow around 2.12 to 0.64 mach.

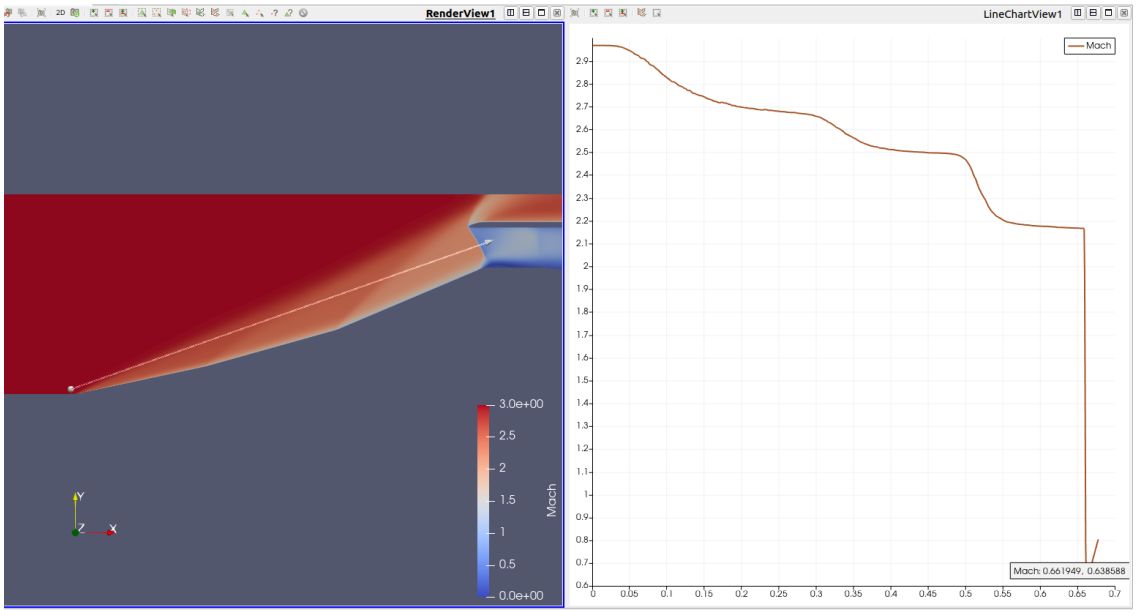

Figure 3.21: Contour Plot of the Ramjet Inlet and Line Chart of Mach on the Ramps

3.4.Machine Learning

In this project, machine learning (ML) is leveraged to enhance the design and optimization of a ramjet engine inlet. The ML workflow enables accurate predictions of critical parameters, such as the Pressure Recovery Coefficient (PRC) and key design variables like ramp count, ramp length, and back pressure. These parameters are essential for optimal engine performance, and the goal is to provide a more efficient design process through data-driven insights.

The dataset used in this project contains various physical and operational parameters related to the ramjet engine. These include parameters such as Mach number, freestream pressure, freestream temperature, mass flow rate, and ramp lengths, among others. These features are used as input for the ML models, which are trained to predict the PRC and design parameters. This workflow employs several advanced techniques in machine learning, including multi-output regression, where multiple target variables are predicted simultaneously, allowing for a more integrated and efficient design prediction process.

The libraries used to implement the machine learning pipeline include:

pandas for data manipulation and analysis, numpy for numerical operations, scikit-learn for machine learning algorithms, model evaluation, and hyperparameter optimization, xgboost for implementing the XGBoost regression models, known for their efficiency and ability to handle complex, non-linear relationships in large datasets, joblib for saving and loading trained models.

The core model used for this task is multi-output regression. Specifically, the XGBRegressor from the xgboost library is employed in a multi-output configuration using the MultiOutputRegressor from scikit-learn. This model is well-suited for the task, as it allows for simultaneous prediction of multiple target variables (such as ramp count, ramp length, and back pressure) based on the input features.

This section provides an overview of the key stages in the machine learning workflow, including:

Data Preparation, Loading and preprocessing the dataset, including outlier detection and removal. Model Training, Training individual regression models for PRC prediction and design parameter prediction using XGBoost in a multi-output configuration. Hyperparameter Optimization, Fine-tuning the model's hyperparameters to maximize performance using techniques like grid search or randomized search. Prediction, Using the trained models to make predictions for new engine designs based on input parameters.

3.4.1. Data Preparation

The first step in the ML process involves preparing the dataset for model training. The dataset consists of the following columns:

- **BaseName**: Identifier for each record in the dataset.
- **Cfg_MACH_NUMBER**: Inlet Mach number for the configuration.
- **Cfg_FREESTREAM_PRESSURE**: Freestream pressure for the configuration.
- **Cfg_FREESTREAM_TEMPERATURE**: Freestream temperature for the configuration.
- **Cfg_MARKER_OUTLET**: Outlet marker of design.
- **Ramp_Lengths**: Ramp length values used in the engine design.
- **Theta**: The cone angle parameter in the first ramp.
- **Backpressure_constant**: The back pressure value used in the simulations.
- **Pressure_Recovery_Coefficient**: The measured PRC for the design.
- **Avg_Massflow_outlet**: The average mass flow rate at the outlet.
- **Avg_Mach_outlet**: The average Mach number at the outlet.
- **Avg_TotalTemp_outlet**: The average total temperature at the outlet.
- **Avg_TotalPress(outlet)**: The average total pressure at the outlet.

Each of these columns represents important parameters that contribute to the performance and design of the ramjet engine. However, before training the models, the dataset needs to be cleaned and prepared.

Since datasets often contain outliers, which can affect the model's ability to generalize, outlier detection is applied using the Interquartile Range (IQR) method. This method

removes data points that fall outside a defined range (1.5 * IQR above Q3 and below Q1). This ensures that the model trains on consistent and reliable data.

The function “load_data(csv_path)“ reads the data, identifies numerical columns, calculates the IQR, and removes rows containing outliers. The cleaned dataset is then used for model training.

```
df = load_data(csv_path)
num = df.select_dtypes(include=[np.number])
Q1 = num.quantile(0.25)
Q3 = num.quantile(0.75)
IQR = Q3 - Q1
df = df[~((num < (Q1 - 1.5 * IQR)) | (num > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Figure 3.22: Data cleaning and outlier removal code

3.4.2. Model Training

With the cleaned data, we proceed to train the machine learning models. The goal is to train two models.

First model is PRC Prediction Model. This model predicts the Pressure Recovery Coefficient (PRC) based on input features such as Inlet and Outlet Mach number, mass flow rate.This part is crucial since for design model to work it need pressure recovery coefficient that is not given by user.

Second model is Design Parameter Prediction Model. This model predicts key design parameters, such as ramp count, ramp length, and cone angle(θ) of first ramp, based on the predicted PRC and user inputs of Inlet and Outlet Mach number, mass flow rate.

To train the models, XGBRegressor from the xgboost library was used. XGBoost is an efficient and powerful machine learning algorithm that handles large datasets and complex relationships well. For each model, a pipeline was build that first scales the input features using StandardScaler and then fits the data to the XGBRegressor model.

K-fold cross-validation was used to evaluate the models' performance. K-fold cross-validation splits the dataset into K subsets, training and testing the model K times. This provides a more reliable estimate of model performance and helps prevent overfitting.

```python
# PRC model
X1 = df[['Avg_Mach_outlet', 'Avg_Massflow_outlet', 'Cfg_MACH_NUMBER']]
y1 = df['Pressure_Recovery_Coefficient']
p1 = Pipeline([('s', StandardScaler()),
               ('x', XGBRegressor(n_estimators=n, learning_rate=lr,
                                  max_depth=md, objective='reg:squarederror',
                                  random_state=42, n_jobs=-1))])
s1 = cross_val_score(p1, X1, y1, cv=kf, scoring='r2', n_jobs=-1)
self.train_log.insert("end",
                      f"PRC R² per fold: {np.round(s1, 4).tolist()}, mean {s1.mean():.4f}\n\n"
                      )

# Design model
X2 = df[['Pressure_Recovery_Coefficient', 'Avg_Mach_outlet',
         'Avg_Massflow_outlet', 'Cfg_MACH_NUMBER']]
y2 = df[['ramp_count', 'ramp_length', 'Theta', 'Backpressure_constant']]
p2 = Pipeline([('s', StandardScaler()), ('m',
                                         MultiOutputRegressor(
                                             XGBRegressor(n_estimators=n, learning_rate=lr,
                                                          max_depth=md, subsample=0.9,colsample_bytree=0.8, objective='reg:squarederror',
                                                          random_state=42, n_jobs=-1)))])
s2 = cross_val_score(p2, X2, y2, cv=kf, scoring='r2', n_jobs=-1)
self.train_log.insert("end",
                      f"Design R² per fold: {np.round(s2, 4).tolist()}, mean {s2.mean():.4f}\n"
                      )
```

Figure 3.23: Training code of the PRC prediction model and Design model

Here, a pipeline is built for the PRC model and Design model, and cross-validation is performed using 7 folds. The R^2 score for each fold is calculated and averaged, providing an estimate of model performance.

3.4.3. Hyperparameter Optimization

Hyperparameter optimization is an essential step in fine-tuning the model for the best possible performance. The model was optimized several key hyperparameters of the XGBRegressor, including n_estimators, learning_rate, max_depth, subsample, colsample_bytree.

For optimization grid search approach applied, where multiple combinations of these hyperparameters tested to find the best configuration. The model's performance is evaluated based on the R^2 score for the design parameter prediction model. The configuration with the highest R^2 score is selected.

Table 3.1: 5 Best Hyperparameter Configuration for Developed Model

Configuration	n_estimators	learning_rate	max_depth	colsample_bytree	R^2
1	**300**	**0.026**	**6**	**0.90**	**0.4901**
2	320	0.026	6	0.90	0.4895

3	340	0.026	6	0.90	0.4895
4	380	0.026	6	0.90	0.4891
5	400	0.026	6	0.90	0.4891

Prediction and Model Evaluation

Once the models are trained and optimized, they are used for predictions. The process follows these steps:

First user inputs Inlet Mach number, Outlet Mach number and mass flow rate then program uses PRC prediction model, to predict the PRC

Secondly, program use both user inputs and predicted PRC as an input then by using Design model it predicts ramp count, ramp length, and cone angle(θ) of first ramp.

```
def run_prediction(self):
    self.pred_log.delete("1.0", "end")
    prc_path = os.path.join(self.base_dir, "ml results", "model_prc_xgb.pkl")
    design_path = os.path.join(self.base_dir, "ml results", "model_design_xgb.pkl")
    if not os.path.exists(prc_path) or not os.path.exists(design_path):
        return messagebox.showerror("Missing", "Train models first")
    prc_m = joblib.load(prc_path)
    des_m = joblib.load(design_path)
    try:
        mout = float(self.mo_entry.get())
        mf = float(self.mf_entry.get())
        Min = float(self.im_entry.get())
    except ValueError:
        return messagebox.showerror("Invalid", "Enter numeric values")
    X1 = pd.DataFrame([[mout, mf, Min]],
                      columns=["Avg_Mach_outlet", "Avg_Massflow_outlet", "Cfg_MACH_NUMBER"])
    prc_pred = prc_m.predict(X1)[0]
    X2 = pd.DataFrame([[prc_pred, mout, mf, Min]],
                      columns=["Pressure_Recovery_Coefficient",
                               "Avg_Mach_outlet", "Avg_Massflow_outlet", "Cfg_MACH_NUMBER"])
    cnt_f, L, θ, bc = des_m.predict(X2)[0]
    cnt = int(round(cnt_f))
    txt = (f"PRC: {prc_pred:.4f}\n"
           f"ramp_count: {cnt}\n"
           f"ramp_length: {L:.3f}\n"
           f"Theta: {θ:.3f}\n"
           f"Backpressure const: {bc:.4f}\n")
    self.pred_log.insert("end", txt)
    self._last_pred = {'Min': Min, 'L': L, 'θ': θ, 'rc': cnt, 'bc': bc,
                       'mout': mout, 'mf': mf, 'prc_pred': prc_pred}
```

Figure 3.24: Design Prediction Model Code

3.5.Gui (Graphical User Interface)

For every new design of Ramjets, engineers are required to calculate the point coordinates of the points in .geo files. After calculating they must update the geo files according to ne parameters. To eliminate this time-consuming process, a GUI developed. Users can create new designs by changing the parameters considering their needs and run the CFD analysis immediately. This ability of GUI reduces the time it takes to calculate shocks, drawing 2D surface on Gmsh and running analysis on SU2 into 15 seconds from 20 minutes. This amount of difference will make the design stage of Ramjet much quicker and cheaper.

The machine learning module of GUI will reduce the trial-and-error method, and it will create new designs without needing to run new analysis for each different parameter. With enough CFD results, the ML models can give sufficient predictions for future designs. By providing mass flow rate, input and output Mach values for the Ramjet, models can predict the overall design and run CFD using those parameters to test the prediction quality. Within seconds models can give new designs, this reduces the amount of CFD needs to run for each try. Figure 35 shows the interface of CFD solver.

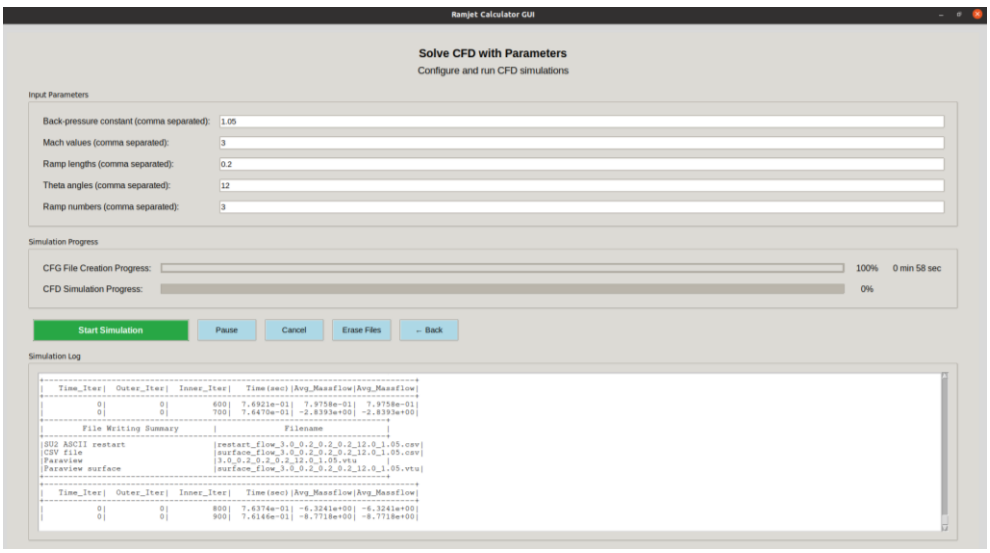

Figure 3.25: Screenshot of The Developed Program, CFD Solving Page

By using machine learning models, the need for CFD analysis will be less and the time it takes to finalize the design stage will be much quicker. Figure 36 shows the interface of ML prediction and test page.

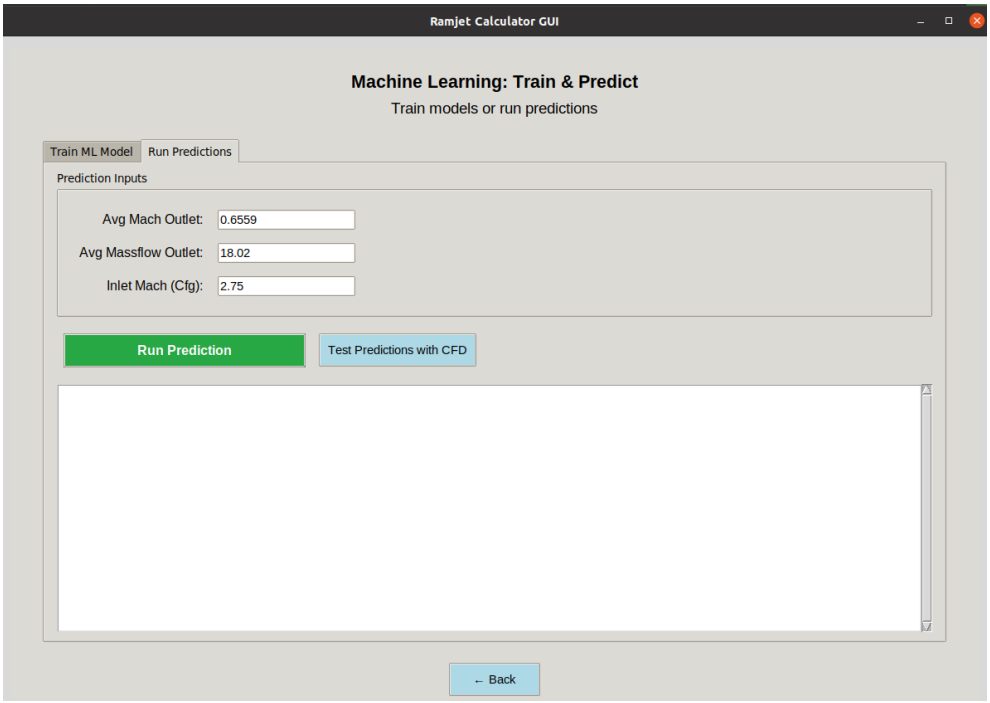

Figure 3.26: Screenshot of The Developed Program, Machine Learning Page

The program is an open-source software which developed on Python. Users can reach these codes via Github(https://github.com/suleymanaltundag/Ramjet-Inlet-Design-.git)

By using software user can work on their own CFD analysis to create better predicting machine learning models. The user manual can help them to learn how to use it(see Appendix 1).

4. RESULTS AND DISCUSSION

After conducting more than 8000 different CFD analysis, resulting data first filtered by software using outlet mach number and mass flow rate upper, lower limits which are defined by user. After computer filtering process finished results were reduced around 1200. Since every inlet design computed with several back pressure values to see the affect of pressure on design, the remained results was filtered manually to determine the optimum back pressure value for each. By manual filtering only 225 results remained to train model.

This results enables the development of a machine learning model using Random Forest and XGBoost. The model is trained with a dataset of 225 filtered design examples, which is obtained from the initial design simulations. The aim of the model is to predict the Ramjet

engine inlet design based on the inlet design parameters; this includes Mach numbers, mass flow rate and pressure recovery.

In terms of model accuracy, five new designs that have never been tested before were tested. The model predictions were compared with the CFD analysis results of these test designs. The average error obtained for each parameter was calculated. The details of these five designs and their average respective error rates are presented in Table 3.

Table 4.1: Test Results of Unknown Data

NO	User Requirements			Machine Learning Errors %		
	Outlet Mach number	Mass Flow Rate (kg/s)	Inlet Mach Number	Outlet Mach number	Mass Flow Rate	Pressure Recovery Coefficient
1	0,55	30,00	3,50	15,17	12,87	2,96
2	0,51	14,40	3,25	7,48	11,69	2,32
3	0,54	15,00	3,00	8,95	6,06	1,32
4	0,61	25,00	2,75	3,31	9,33	1,92
5	0,52	21,00	2,50	5,39	4,36	0,62
	Averages			**8,06**	**8,86**	**1,83**

While the results of the machine learning model show that the model provides valuable insights into the Ramjet engine design process, some limitations were also recognized, in the mass flow rate and outlet Mach number estimation. The relatively high error rate observed in these parameters can be attributed to several factors.

First, the limited computer hardware used for the project created limitations in terms of processing power and analysis time. These limitations affected the accuracy of the model, especially in the training phase, leading to some deviations in the predictions. In particular, the model's ability to generalize between different designs was challenged by the limited size and diversity of the data set. Especially lack of data above 3.5 Mach number caused higher error around this value.

Also, the reference geometry parametrization used in the model may not have been sufficiently optimized, which led to some inconsistencies in the predictions. The lack of parameterization led to a high proportion of unusable data in the CFD simulations, which negatively affected the quality of the training data. As a result, the machine learning model struggled to make accurate predictions for some parameters.

In future studies, it is understood that the dataset should be expanded and the parameterization should be optimized to improve the accuracy of the model. A more comprehensive data set will enable the model to better learn the complex relationships between input parameters and engine performance. Furthermore, improvements in the

reference geometry and parameterization will reduce the proportion of unusable data from CFD simulations and minimize the computational power required for model training.

5. CONCLUSION

A machine learning model with Random Forest and XGBoost algorithm was developed using ScikiLearn library of Python. This model is capable of generating a ramjet engine inlet design in accordance with the requirements specified by the users. In addition, the software developed within the scope of the project, which consist this machine learning model aswell, offers the opportunity to test and verify the designs using Gmsh, SU2 and Paraview. While the software allows the modeling of existing data with different parameters, it also allows the model to be retrained with new data to be created. The model developed according to these studies, produced not highly consistent designs due to lack of data. But it comply with the needs of the users in the preliminary design phase. In this way, it has been possible make the ramjet engine design process in a faster, more efficient and environmentally friendly way.

6. REFERENCES

About ParaView. (n.d.). Retrieved June 21, 2025, from https://www.paraview.org/about/

Anderson, J. D. (2010). *Fundamentals of Aerodynamics*. McGraw-Hill Education.

Anderson, J. D. (2021). *Modern compressible flow: With historical perspective* (4th ed., international student ed). McGraw-Hill Education. https://www.mheducation.com/highered/product/Modern-Compressible-Flow-With-Historical-Perspective-Anderson.html

Anderson, J. D. (20). *Computational fluid dynamics: The basics with applications* (International Editions 1995). McGraw-Hill.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, *13*(null), 281–305.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blazek, J. (2015). *Computational Fluid Dynamics: Principles and Applications*. Butterworth-Heinemann.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., & Alonso, J. J. (2016a). SU2: An Open-Source Suite for Multiphysics Simulation and Design. *AIAA Journal*, *54*(3), 828–846. https://doi.org/10.2514/1.J053813

Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., & Alonso, J. J. (2016b). SU2: An Open-Source Suite for Multiphysics Simulation and Design. *AIAA Journal*, *54*(3), 828–846. https://doi.org/10.2514/1.J053813

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. (n.d.). Retrieved June 20, 2025, from https://gmsh.info/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning (adaptive computation and machine learning)*. MIT Press Cambridge.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer. https://doi.org/10.1007/978-0-387-84858-7

Hellsten, A. (2004). *New two-equation turbulence model for aerodymamics applications*. Helsinki University of Technology.

Hill, P. G., & Peterson, C. R. (1992). *Mechanics and Thermodynamics of Propulsion*. Addison-Wesley.

Hirsch, C. (2007). *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*. Elsevier.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jameson, A., Schmidt, W., & Turkel, E. (1981). Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *14th Fluid and Plasma Dynamics Conference*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.1981-1259

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1137–1143.

Liepmann, H. W., & Roshko, A. (1957). *Elements of Gasdynamics*. Wiley.

Mattingly, J. D. (2006). *Elements of Propulsion: Gas Turbines and Rockets*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/4.861789

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, *43*(2), 357–372. https://doi.org/10.1016/0021-9991(81)90128-5

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, *25*. https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html

SPALART, P., & ALLMARAS, S. (n.d.). A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.1992-439

Spalart, P. R. (2000). Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow*, *21*(3), 252–263. https://doi.org/10.1016/S0142-727X(00)00007-2

SU2 Development Team. (2024, February 26). *SU2 version 8.0.1 "Harrier."* GitHub. https://github.com/su2code/SU2/releases

Sutton, G. P., & Biblarz, O. (2010). *Rocket Propulsion Elements*. John Wiley & Sons.

Vapnik, V., & Izmailov, R. (2020). Complete statistical theory of learning: Learning using statistical invariants. *Proceedings of the Ninth Symposium on Conformal and Probabilistic Prediction and Applications*, 4–40. https://proceedings.mlr.press/v128/vapnik20a.html

7. APPENDICIES

7.1.Appendix 1 – User Manual

Main Menu Overview

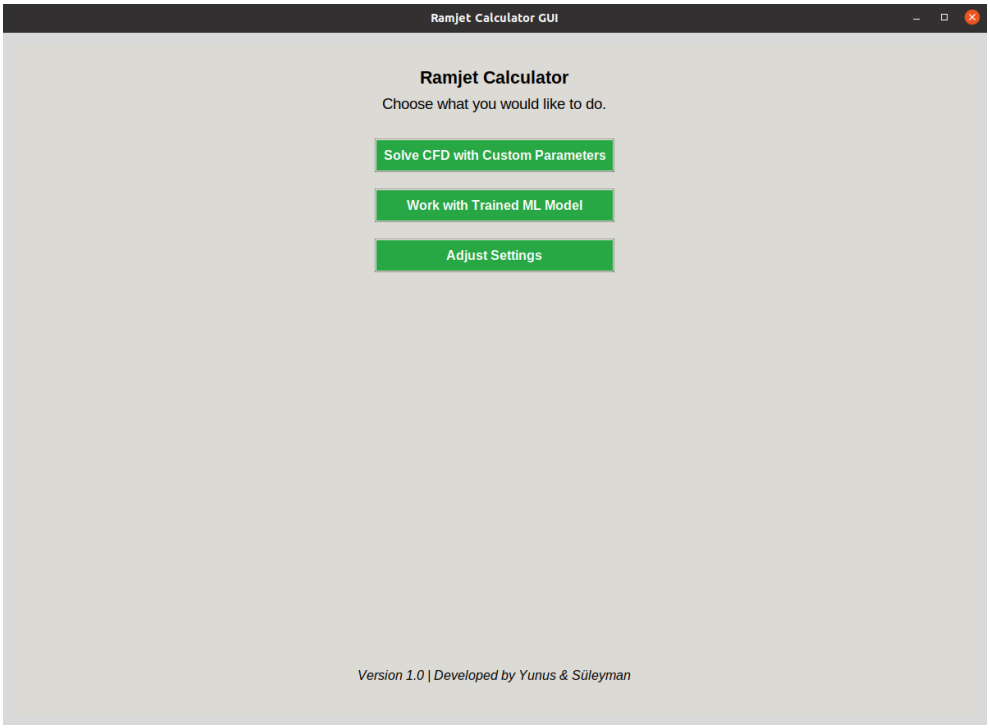

Figure 7.1: Screenshot of The Developed Program, Main Page

The home screen of the GUI provides three options to direct users according to their needs. These buttons let user access other pages to use the program. These buttons are:

1. Solve CFD with Custom Parameters
2. Work with Trained ML Model
3. Adjust Settings

Users can select whether they want to train their own machine learning model on their own devices or work with already trained machine learning model which is trained by developers. The last button helps user to make changes of the settings of the program.

CFD Workflow Options

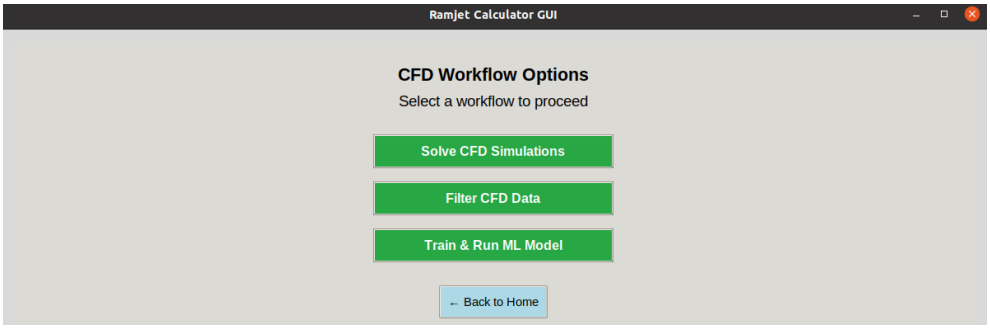

Figure 7.2: Screenshot of The Developed Program, CFD Workflow Page

When user clicks “Solve CFD with Custom Parameters” button, this means that user want to run CFD analysis on their own devices. By using the data gathered from the CFD results, the user can train their own machine learning model. On this page there are three buttons which are placed in queue:

1. Solve CFD Simulations
2. Filter CFD Data
3. Train & Run ML Model

Solve CFD with Parameters

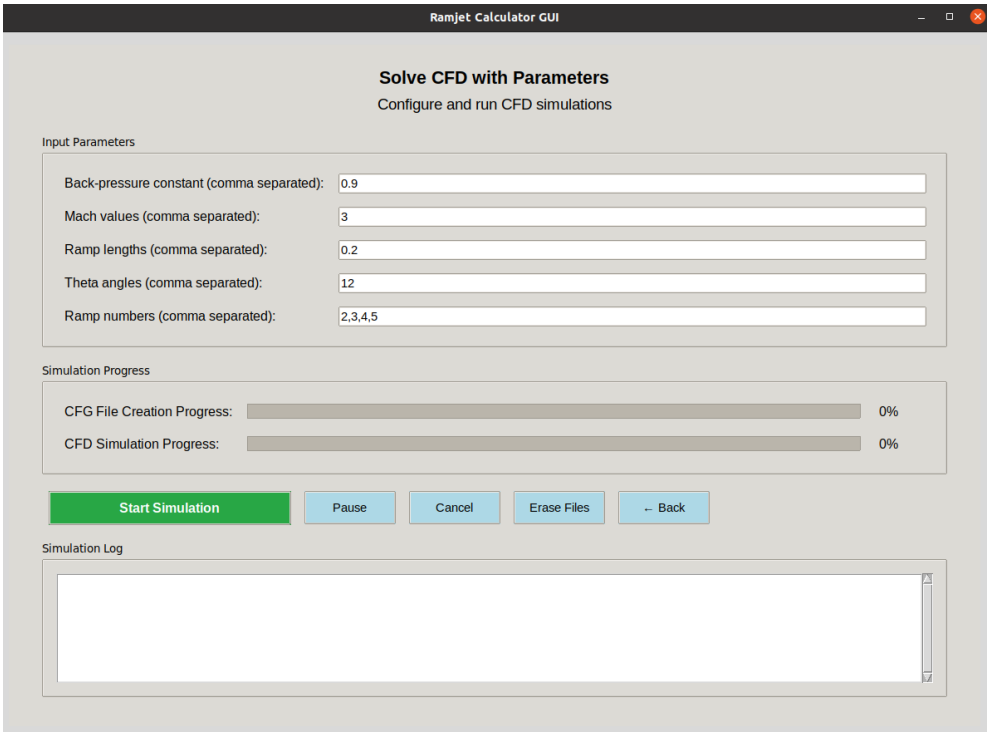

Figure 7.3: Screenshot of The Developed Program, CFD Solving Page

After clicking the "Solve CFD Simulations" button, this page will open to run the CFD simulations. In this page there are two main parts in the page: input field, control buttons & logs. In the input field there are five parameters to be filled by the user. As it is expressed in the page, these numerical input values must be separated by comma. Like in the ramp number input filed.

Input Fields:

Back-pressure constant: To see the effects of different back-pressure values on the same designs, the user can adjust the values in input field. These inputs will be multiplied by the back-pressure values calculated by the program during .cfg file creation process for SU2. Required input values are between 0 and 1.

Mach values: User can run CFD analysis with different inlet velocities expressed in mach.

Ramp lengths: To determine the lengths of each ramp's user can type in meter unit.

Theta angles: The first ramp's theta angle of Ramjets determines the angle of other ramps; user can define the angle limits for the first theta angle in degree unit.

Ramp numbers: GUI can run CFD for the Ramjet which are designed with 2,3 and 4 ramps.

Control Buttons & Logs

Start Simulation: Initiates file creation and run CFD analysis.

Pause / Cancel / Erase Files: Control simulation execution.

Log Box: Displays real-time system and solver outputs, users can follow the iterations completed by SU2 CFD solver whether analysis diverged or converged.

Progress Bars: Show progress of .cfg file generation and CFD execution.

Filter CFD Data

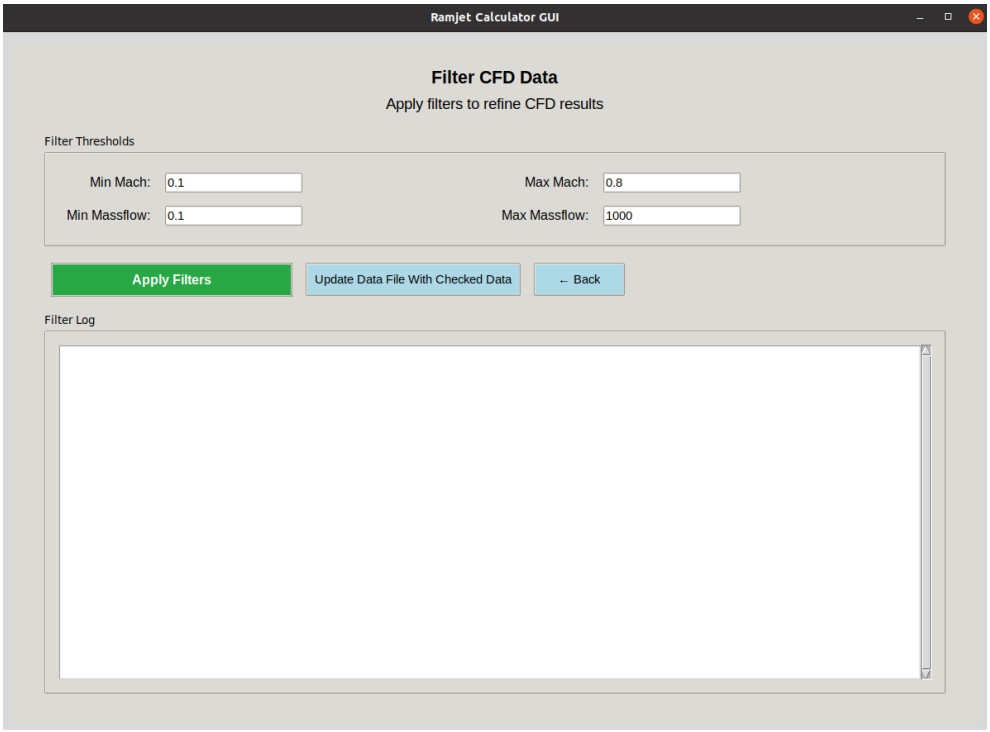

Figure 7.4: Screenshot of The Developed Program, Data Filtering Page

This page helps users filter outlier CFD results like diverged analysis, those analysis with supersonic outlet Mach or useless mass flow value. There are 2 buttons and 4 input fields on this page. Users can type the limits of the outlet Mach and outlet mass flow values to eliminate those analysis results with useless outputs. Mass flow input field filters the data according to inputs given in unit of kg/s.

Apply Filters: This button creates files which has these “single_type”, “many_type”) suffixes in their names. Files will be saved in “csv files” folder. This button activates the function that reads the last row of all the “_history.csv” files in “cfg files” folder and determine whether those analysis converged or diverged. If this function detects the converged analysis, then it reads “.cfg” and “_history.csv” files and stores those needed data in “data_many_type.csv” and stores the ID of those diverged analysis in name of “diverged_many_type.csv”. By using converged data file, it creates “.xlsx” files for statistics of converged results. Users can make comments by examining the graphs or numerical tables in those files to be able to understand the quality of collected CFD analysis results. After that, this function continues to run the same process again but it detects the same geometry designs with different back-pressure values so user can select the best CFD result with

optimum back pressure value. For this process those files named as “data_single_type.csv”, “diverged_single_type.csv”. Users have to eliminate the same designs with unacceptable back pressure results and select the best CFD result among those by visualizing the “.vtu” files in Paraview program. After determining the ID’s of accepted CFD results, users must write those ID’s in “checked_data.csv” file which is in “csv files” folder.

Update Data File with Checked Data: After the “checked_data.csv” file updated by the users, then this button must be clicked so program will read those ID in this file and update the “data.csv” file only using those accepted CFD results.

Train & Run ML Model

Train ML Model

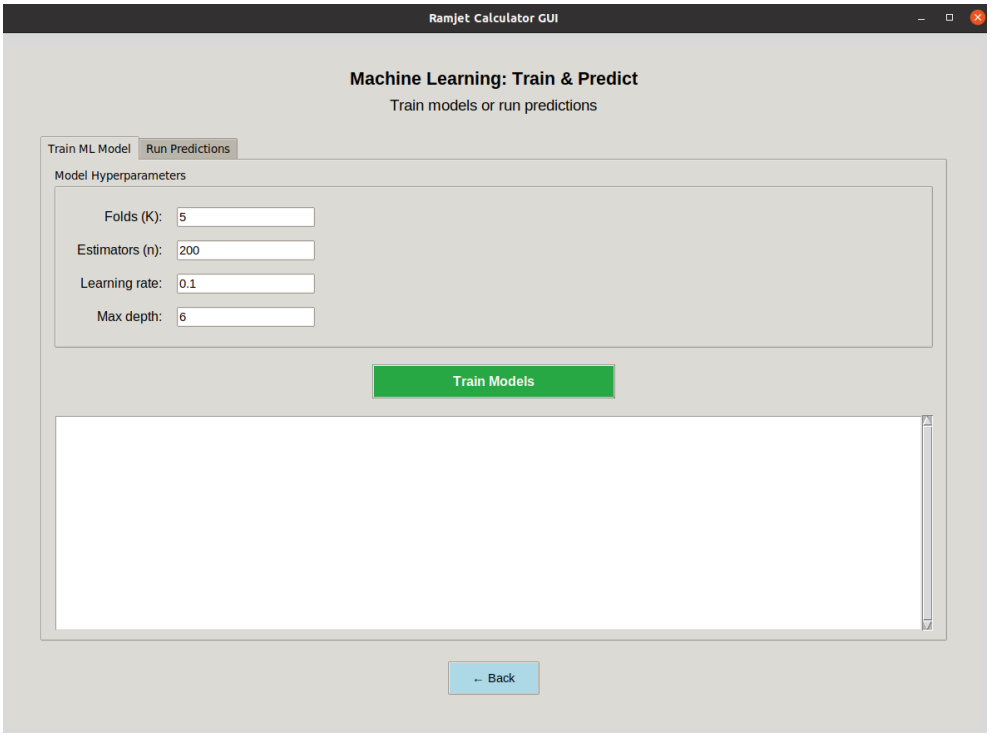

Figure 7.5: Screenshot of The Developed Program, Machine Learning Train Page

After finishing filtering process of data users can train their own machine learning model by using the “data.csv” file. There are four settings on this page(Breiman, 2001):

Table 7.1: Hyperparameters Values of ML Model

Hyper-parameter	Purpose	Typical Range	Default
Folds (K)	Number of K-Fold CV splits	3 – 10	7
Estimators (n)	Number of trees in XGBoost	50 – 500	300
Learning rate	Gradient shrinkage factor	0.01 – 0.3	0.026
Max depth	Maximum tree depth	3 – 10	6

Users can make changes to those settings for their own models. After clicking the button on the page, the machine learning model will be saved in “ml results” folder.

Run Predictions

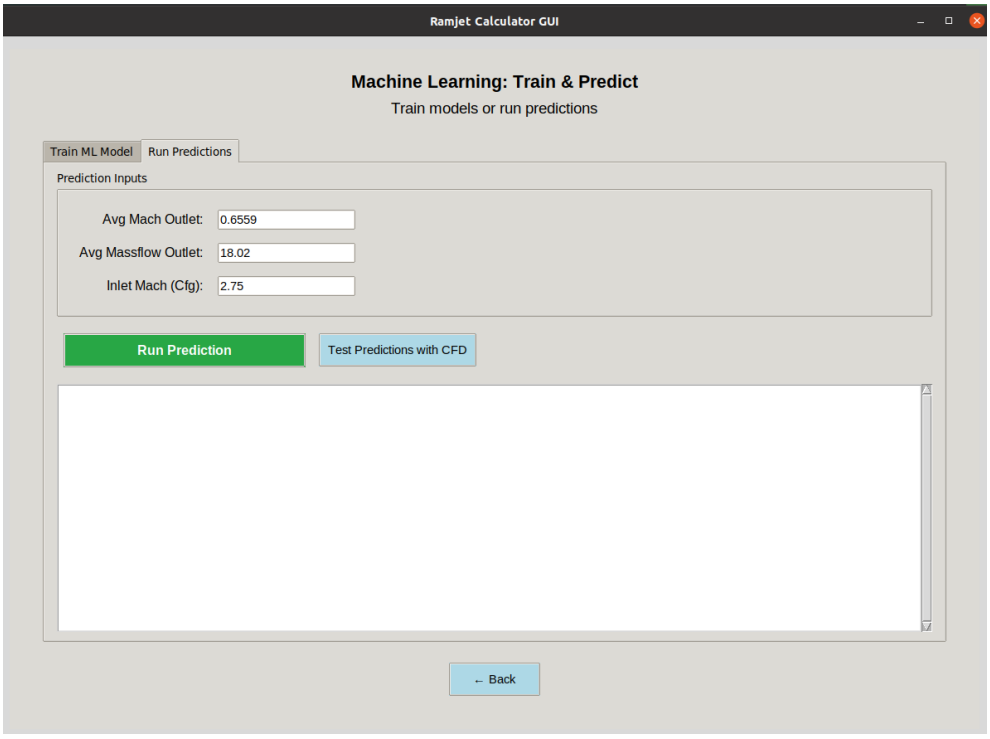

Figure 7.6: Screenshot of The Developed Program, Machine Learning Prediction Page

After the models are saved in expressed folder, users can test their machine learning model by these data in input fields:

Average Mach (Outlet): The Mach value of air which passed through the intake and will continue to combustion chamber.

Average Mass flow (Outlet): The mass flow value of air which passed through the intake and will continue to combustion chamber. The unit is kg/s for this input field.

Inlet Mach (Inlet): The Mach value of air which will pass through the intake.

“Run Prediction” button use these inputs to calculate the Pressure Recovery Coefficient value from “model_prc_xgb.pkl” model. After that by using three inputs and one PRC result, the main design model“model_design_xgb.pkl” will determine design parameters to draw geometry.

Users can see the predicted results for, theta angle, number of ramps, lengths of ramps, pressure recovery coefficient and back-pressure constant values. With the “Test predictions with CFD” button, users can run CFD analysis on SU2 using those predicted results. If the CFD analysis converges then program opens “.vtu” file of runned CFD on Paraview program, draws the 3D model (see Figure 7.7) of design and gives the error between CFD results and values predicted by machine learning models. Then a question appears on the screen which ask if user want to add this CFD result in “data.csv” to increase the train data and have better machine learning model. If user accepts and clicks “Yes” then this new data will be added in “data.csv” file and it will rerun the training model with updated data. If user don’t accept the result and wants to rerun the CFD analysis again with different back-pressure they can say “No” to that popup question and type new back-pressure value. With new back-pressure value, CFD analysis will run again, and it will ask the same popup question to users.

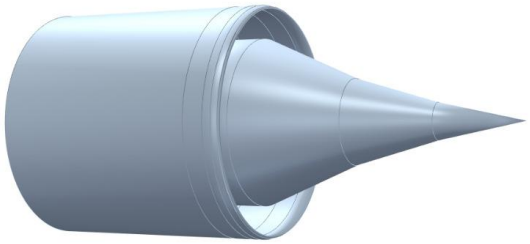

Figure 7.7: Screenshot of The Developed Program, 3D model

Work with Trained ML Model

If user clicks the “Work with Trained ML Model” button on the main page of GUI, user will see the same “Train ML Model” and “Run Predictions” pages but on this page the machine learning model’s data is created by the developers so users can run machine learning predictions without running CFD on their own devices. This will help them to gain time for their projects without struggling with CFD data and manual filtering processes. But machine learning model files created by this pages will be saved in “already trained model/ml results”, those models will use the “data.csv” file in “already trained model/csv files” folder.

Adjust Settings

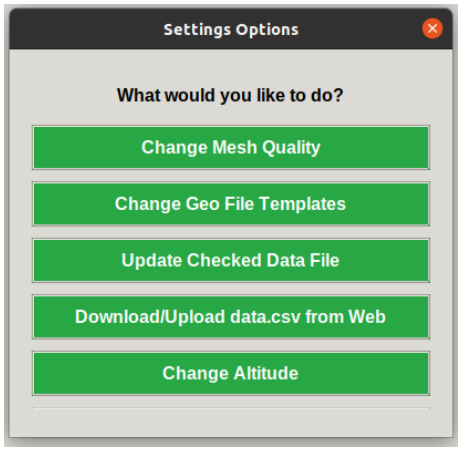

Figure 7.8: Screenshot of The Developed Program, Settings Page

After clicking the “Adjust Settings” button, user have five options to make change on the settings of the program.

1. Change Mesh Quality
2. Change Geo File Templates
3. Update Checked Data File
4. Download/Upload data.csv from Web
5. Change Altitude

Change Mesh Quality

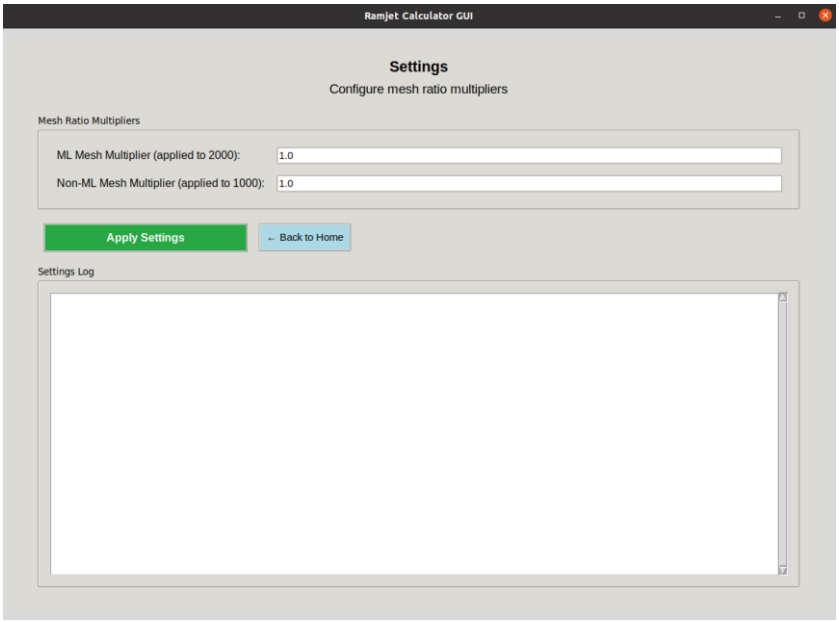

Figure 7.9: Screenshot of The Developed Program, Meshing Quality Page

While creating ".geo" files for CFD analysis mesh quality is calculated according to the lengths of each ramp so for every design the mesh quality is parametrized according to certain variables. In that mesh quality formula, there is a constant as 1000 for "Solve CFD with Parameters" page and it is 2000 for "Test predictions with CFD" page as default. But users can increase or decrease the quality of mesh by typing new numerical value to multiply these default numbers. If user want to decrease the quality of mesh to reduce the time takes to complete the CFD analysis for "Test predictions with CFD" page they can type 0.8 in first input filed to multiply with 2000, new mesh constant will be 1600 rather than 2000.

Change Geo File Templates

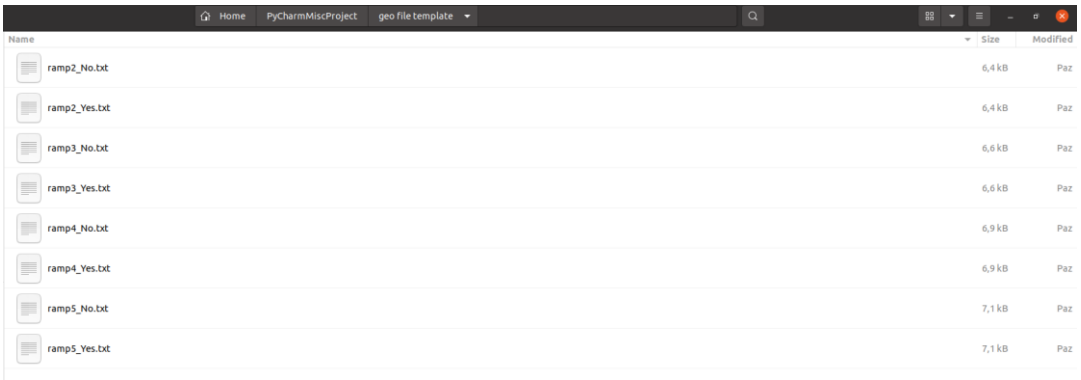

Figure 7.10: Screenshot of The Developed Program, .geo File Templates

By clicking “Change Geo File Templates” button user will be directed to “geo file template” folder so user can make changes on the shape drawing template for “.geo” files.

Update Checked Data File

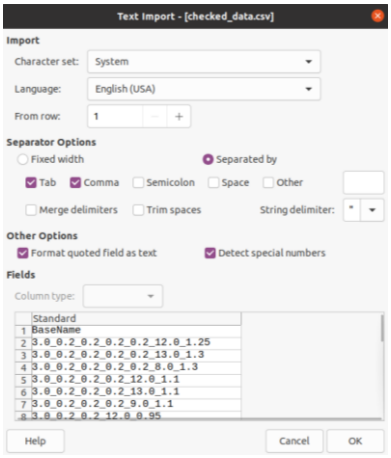

Figure 7.11: Screenshot of The Developed Program, Checked Data File

“Update Checked Data File” button will open the “checked_data.csv” file which is in “csv file” folder. Users can add new checked data or erase ID for the filtering process. “Update Data File With Checked Data” button on the “Filter CFD Data” page needs to be clicked by the user after saving the file with updated data. After filtering process, machine learning must be trained again using the updated “data.csv” file.

Download/Upload data.csv from Web

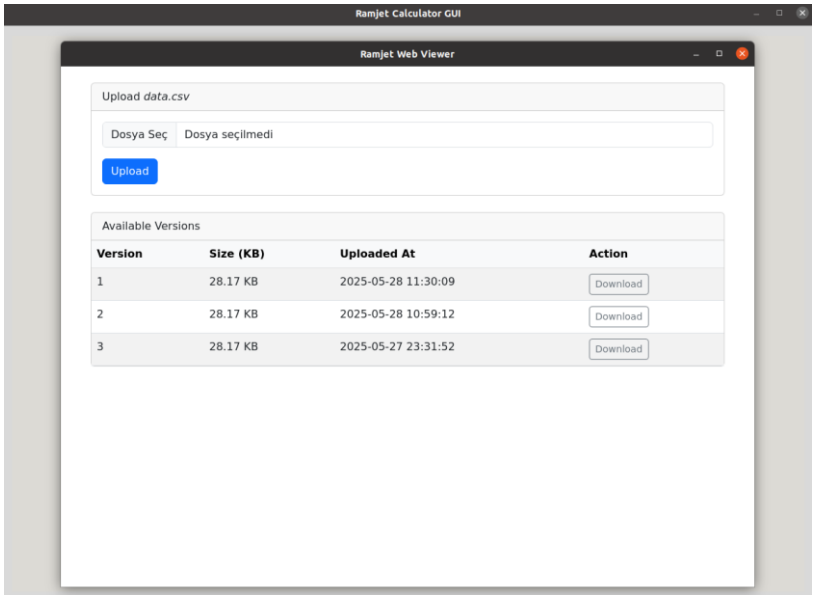

Figure 7.12: Screenshot of The Developed Program, Website Page

Each user will have their own “data.csv” file when they run their own CFD analysis. They will filter those data according to their needs and train their own machine learning models. At the end of filtering process their data will be ready for training on the program. If users want to share their own “data.csv” file with other users around the globe they can use this page to upload that file also if they don’t have enough time to run CFD and spend time with this process they can download already filtered data file from the Web. Devices need internet only for this page to operate.

Change Altitude

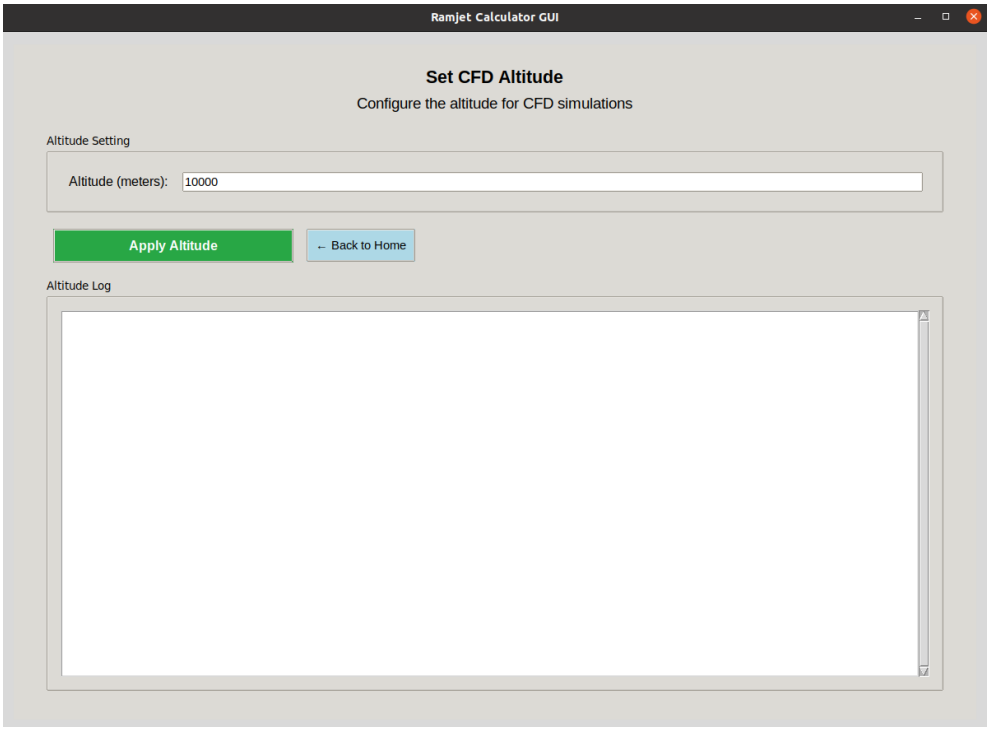

Figure 7.13: Screenshot of The Developed Program, Altitude Adjusting Page

CFD analysis require pressure values to run without any problem and environmental pressure values and back pressure values calculated according to the altitude of the flight of Ramjet. Users can adjust this altitude value and initiate CFD analysis. Also when user clicks “Test predictions with CFD” button on machine learning page, a CFD analysis starts so this analysis requires altitude value to make calculations. Changes of the altitude value affect this calculation so user need to be careful changing altitude and testing machine learning trained by the developers because developers trained at 10000 m altitude which is given as default setting.

7.2.Appendix 2 - .geo codes

Below ".geo" Code for template design used in project

```
// Gmsh project created on Wed Jun 04 23:25:22 2025

SetFactory("OpenCASCADE")

point1_x = 0.2000;

point1_y = 0.0000;

point2_x = 0.4000;

point2_y = 0.0425;

point3_x = 0.6000;

point3_y = 0.0984;

point4_x = 0.8257;

point4_y = 0.1939;

point5_x = 0.8000;

point5_y = 0.2547;

size_ratio=1.0000;

Point(1) = {0, 0, 0, 1.0};

Point(2) = {point1_x, point1_y, 0, 1.0};

Point(3) = {point2_x, point2_y, 0, 1.0};

Point(4) = {point3_x, point3_y, 0, 1.0};

Point(5) = {point4_x, point4_y, 0, 1.0};

Point(6) = {point4_x+0.085*size_ratio, point4_y, 0, 1.0};

Point(7) = {point4_x+0.20533*size_ratio, point4_y-0.015*size_ratio, 0, 1.0};

Point(8) = {point4_x+0.275*size_ratio, point4_y-0.025*size_ratio, 0, 1.0};

Point(9) = {point5_x+0.6*size_ratio, point5_y-(((point5_y-point4_y)*5.5)/2), 0, 1.0};

Point(10) = {point5_x+0.6*size_ratio, point5_y-(((point5_y-point4_y)*1.5)/2), 0, 1.0};

Point(11) = {point5_x+0.2*size_ratio, point5_y, 0, 1.0};

Point(12) = {point5_x+0.1*size_ratio, point5_y, 0, 1.0};

Point(13) = {point5_x+0.05*size_ratio, point5_y, 0, 1.0};

Point(14) = {point5_x, point5_y, 0, 1.0};
```

```
Point(15) = {point5_x+0.005*size_ratio, point5_y+0.004*size_ratio, 0, 1.0};
Point(16) = {point5_x+0.02*size_ratio, point5_y+0.008*size_ratio, 0, 1.0};
Point(17) = {point5_x+0.05*size_ratio, point5_y+0.008*size_ratio, 0, 1.0};
Point(18) = {point5_x+0.1*size_ratio, point5_y+0.008*size_ratio, 0, 1.0};
Point(19) = {point5_x+0.2*size_ratio, point5_y+0.008*size_ratio, 0, 1.0};
Point(20) = {point5_x+0.2*size_ratio, point5_y+0.05*size_ratio, 0, 1.0};
Point(21) = {0, point5_y+0.05*size_ratio, 0, 1.0};
Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 5};
Line(5) = {5, 6};
Line(6) = {6, 7};
Line(7) = {7, 8};
Line(8) = {8, 9};
Line(9) = {9, 10};
Line(10) = {10, 11};
Line(11) = {11, 12};
Line(12) = {12, 13};
Line(13) = {13, 14};
Line(14) = {14, 15};
Line(15) = {15, 16};
Line(16) = {16, 17};
Line(17) = {17, 18};
Line(18) = {18, 19};
Line(19) = {19, 20};
Line(20) = {20, 21};
Line(21) = {21, 1};
Physical Curve("inlet") = {21};
Physical Curve("axis") = {1};
```

```
Physical Curve("wall") = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18};

Physical Curve("outlet") = {9};

Physical Curve("far_field") = {19, 20};

Transfinite Curve {21} = 101 Using Progression 1;

Transfinite Curve {1} = 151 Using Progression 1;

Transfinite Curve {2} = 200 Using Progression 1;

Transfinite Curve {3} = 200 Using Progression 1;

Transfinite Curve {4} = 200 Using Progression 1;

Transfinite Curve {12} = 50 Using Progression 1;

Transfinite Curve {13} = 50 Using Progression 1;

Transfinite Curve {10} = 252 Using Progression 1;

Transfinite Curve {8} = 273 Using Progression 1;

Transfinite Curve {16} = 30 Using Progression 1;

Transfinite Curve {17} = 31 Using Progression 1;

Transfinite Curve {18} = 51 Using Progression 1;

Transfinite Curve {14} = 11 Using Progression 1;

Transfinite Curve {15} = 11 Using Progression 1;

Transfinite Curve {9} = 299 Using Progression 1;

Transfinite Curve {19} = 61 Using Progression 1;

Transfinite Curve {5} = 101 Using Progression 1;

Transfinite Curve {6} = 100 Using Progression 1;

Transfinite Curve {20} = 160 Using Progression 1;

Transfinite Curve {7} = 51 Using Progression 1;

Transfinite Curve {11} = 51 Using Progression 1;

Curve Loop(1) = {20, 21, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19};

Plane Surface(1) = {1};

// --- Global Mesh Options ---

Mesh.Smoothing = 10;  // Apply 10 smoothing iterations

Mesh.Optimize = 1;     // Enable basic mesh optimization

Field[1] = BoundaryLayer;
```

```
Field[1].EdgesList = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18};

Field[1].hwall_n = 0.0001;

Field[1].thickness = 0.001;

Field[1].hfar = 0.008;

Field[1].ratio = 1.2;

Field[1].IntersectMetrics = 0;

Field[1].Quads = 1;

BoundaryLayer Field = 1;

// Define the Background Mesh Size Field using a Distance field for key features

Field[2] = Distance;

Field[2].EdgesList = {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18}; // specify the curves near critical features

// Use a Threshold field to vary the mesh size based on the distance from Field[2]

Field[3] = Threshold;

Field[3].IField = 2;      // Use the distance from Field[2]

Field[3].LcMin = 0.0015;   // Fine mesh size near key features

Field[3].LcMax = 0.008;    // Coarse mesh size far from key features

Field[3].DistMin = 0.015193667097058203;   // Below this distance, use LcMin

Field[3].DistMax = 0.030387334194116405;   // Above this distance, use LcMax

// 7. Combine All Fields Using the Min Operator

Field[4] = Min;

Field[4].FieldsList = {1, 3}; // take the smallest mesh size computed by these fields

Background Field = 4;  // Set Field[7] as the background mesh size field

// --- Additional meshing directives ---

Physical Surface("domain") = {1};
```

7.3.Appendix 3 - .cfg code

Below ".cfg" code for template design used in project

```
%-------------------- SU2 CFD Configuration --------------------

MESH_FILENAME= 3.0_0.2_0.2_0.2_12.0_1.05.su2

%------------- DIRECT, ADJOINT, AND LINEARIZED PROBLEM DEFINITION ------------%
```

```
SOLVER= RANS

FLUID_MODEL= STANDARD_AIR

KIND_TURB_MODEL= SST

MATH_PROBLEM= DIRECT

RESTART_SOL= NO

SYSTEM_MEASUREMENTS= SI

%----------- COMPRESSIBLE FREE-STREAM DEFINITION (10 km, Mach=3) ----------%

MACH_NUMBER= 3.0

REYNOLDS_NUMBER= 25445174    %% Approx. for 10 km altitude & L = 1 m

REYNOLDS_LENGTH= 1

AOA= 0.0

SIDESLIP_ANGLE= 0.0

%% Standard Atmosphere at 10000 m

FREESTREAM_PRESSURE= 26435    %% Pa

FREESTREAM_TEMPERATURE= 223.1   %% K

INIT_OPTION= REYNOLDS

FREESTREAM_OPTION= TEMPERATURE_FS

%% ---------------------- REFERENCE VALUE DEFINITION ----------------------%%

REF_ORIGIN_MOMENT_X = 0.00

REF_ORIGIN_MOMENT_Y = 0.00

REF_ORIGIN_MOMENT_Z = 0.00

REF_LENGTH= 1

REF_AREA= 0

%% -------------------- BOUNDARY CONDITION DEFINITION ----------------------%%

MARKER_HEATFLUX= ( wall,0 )

MARKER_FAR= ( far_field,26435)

%% Updated supersonic inlet conditions:

%%  Velocity = 898.4 m/s

MARKER_SUPERSONIC_INLET= ( inlet, 223.1, 26435, 898.4, 0.0, 0.0 )

%% Example outlet boundary (static back-pressure):
```

%% Adjust as needed if you require a normal shock inside the inlet.

MARKER_OUTLET= (outlet, 369926.113)

MARKER_SUPERSONIC_OUTLET= (far_field)

AXISYMMETRIC= YES

MARKER_SYM= (axis)

%% ---------------------------- VISCOSITY MODEL ---------------------------------%%

VISCOSITY_MODEL= SUTHERLAND

MU_REF= 1.716E-5

MU_T_REF= 273.15

SUTHERLAND_CONSTANT= 110.4

%% ------------- COMMON PARAMETERS DEFINING THE NUMERICAL METHOD ----------%%

NUM_METHOD_GRAD= WEIGHTED_LEAST_SQUARES

CFL_NUMBER= 1.1979

CFL_ADAPT= YES

CFL_ADAPT_PARAM= (0.1, 2.0, 5.0, 10)

RK_ALPHA_COEFF= (0.66667, 0.66667, 1.000000)

ITER= 10000

LINEAR_SOLVER= FGMRES

LINEAR_SOLVER_PREC= ILU

LINEAR_SOLVER_ERROR= 1E-6

LINEAR_SOLVER_ITER= 20

%% --------------------------- MULTIGRID PARAMETERS -----------------------------%%

MGLEVEL= 0

MGCYCLE= W_CYCLE

MG_PRE_SMOOTH= (1, 2, 3, 3)

MG_POST_SMOOTH= (0, 0, 0, 0)

MG_CORRECTION_SMOOTH= (0, 0, 0, 0)

MG_DAMP_RESTRICTION= 0.8

MG_DAMP_PROLONGATION= 0.8

%% -------------------- FLOW NUMERICAL METHOD DEFINITION -----------------------%%

CONV_NUM_METHOD_FLOW= ROE

MUSCL_FLOW= NO

SLOPE_LIMITER_FLOW= VENKATAKRISHNAN

LAX_SENSOR_COEFF= 0.15

JST_SENSOR_COEFF= (0.5, 0.02)

TIME_DISCRE_FLOW= EULER_IMPLICIT

%% ---------------------------- CONVERGENCE PARAMETERS ----------------------------%%

CONV_FIELD= RMS_DENSITY

CONV_RESIDUAL_MINVAL= -5

CONV_STARTITER= 100

CONV_CAUCHY_ELEMS= 100

CONV_CAUCHY_EPS= 1E-7

%% --------------------- TURBULENT NUMERICAL METHOD DEFINITION -------------------%%

CONV_NUM_METHOD_TURB= SCALAR_UPWIND

TIME_DISCRE_TURB= EULER_IMPLICIT

CFL_REDUCTION_TURB= 1.0

%% ------------------------- INPUT/OUTPUT INFORMATION --------------------------%%

MARKER_PLOTTING= (outlet)

MARKER_MONITORING= (outlet)

MARKER_ANALYZE= (outlet)

SCREEN_OUTPUT=(ITER, WALL_TIME, SURFACE_MASSFLOW)

MESH_FORMAT= SU2

MESH_OUT_FILENAME= mesh_out_3.0_0.2_0.2_0.2_12.0_1.05_.su2

SOLUTION_FILENAME= solution_flow_3.0_0.2_0.2_0.2_12.0_1.05_.dat

SOLUTION_ADJ_FILENAME= solution_adj_3.0_0.2_0.2_0.2_12.0_1.05_.dat

TABULAR_FORMAT= CSV

OUTPUT_FILES= (CSV,SURFACE_CSV, PARAVIEW, SURFACE_PARAVIEW)

HISTORY_OUTPUT= (ITER, RMS_RES,
SURFACE_MASSFLOW,SURFACE_TOTAL_PRESSURE,SURFACE_TOTAL_TEMPERATURE,SURFACE_MACH)

CONV_FILENAME= 3.0_0.2_0.2_0.2_12.0_1.05_history.csv

RESTART_FILENAME= restart_flow_3.0_0.2_0.2_0.2_12.0_1.05.dat

RESTART_ADJ_FILENAME= restart_adj_3.0_0.2_0.2_0.2_12.0_1.05.dat

VOLUME_FILENAME= 3.0_0.2_0.2_0.2_12.0_1.05.vtu

VOLUME_ADJ_FILENAME= adjoint_3.0_0.2_0.2_0.2_12.0_1.05.vtu

GRAD_OBJFUNC_FILENAME= of_grad_3.0_0.2_0.2_0.2_12.0_1.05.dat

SURFACE_FILENAME= surface_flow_3.0_0.2_0.2_0.2_12.0_1.05.csv

SURFACE_ADJ_FILENAME= surface_adjoint_3.0_0.2_0.2_0.2_12.0_1.05.csv

SCREEN_WRT_FREQ_INNER= 100

SCREEN_WRT_FREQ_OUTER= 100

SCREEN_WRT_FREQ_TIME= 100

HISTORY_WRT_FREQ_INNER= 5

HISTORY_WRT_FREQ_OUTER= 5

HISTORY_WRT_FREQ_TIME= 5

%% ---------- DESIGN VARIABLE PARAMETERS ----------%%

DV_KIND= SCALE_GRID

DV_PARAM= (1.0)

DV_VALUE= 10.0