BCSE497J Project-I

TIME-SERIES PREDICTION OF SPRAY CHARACTERISTICS USING CLASSICAL MACHINE LEARNING APPROACHES

22BCE3942 YUVIN RAJA

Under the Supervision of

P. SURESH KUMAR

Assistant Professor Sr. Grade 2

Automotive Research Centre (ARC)

B.Tech.

in

Computer Science and Engineering

School of Computer Science and Engineering

September 2025

ABSTRACT

Accurate prediction of spray characteristics, such as spray angle and penetration length, is critical for enhancing control and efficiency in applications like fuel injection in combustion engines, agricultural spraying, and thermal systems. Traditional approaches rely on computationally expensive computational fluid dynamics (CFD) simulations or time-consuming experimental methods, creating a need for faster, data-driven alternatives suitable for real-time control applications. This study addresses the research gap by evaluating six classical machine learning models—linear regression, decision tree, random forest, gradient boosting, support vector regressor, and K-nearest neighbors—for time-series prediction of spray characteristics using experimental data.

The dataset comprises 726 experimental samples across six sequential runs derived from spray experiments measured via shadowgraph and Mie scattering techniques. Six input features were utilized: time step, chamber pressure, chamber temperature, injection pressure, fuel density, and kinematic viscosity. Four output variables were predicted: spray angle and penetration length measured through both shadowgraph and Mie scattering methods. A time-aware 80/20 holdout split was implemented to preserve temporal dependencies, avoiding data leakage common in traditional random splitting approaches. Model performance was evaluated using mean squared error (MSE), mean absolute error (MAE), and R^2 score metrics.

Results demonstrate that all models achieve high accuracy for penetration length prediction ($R^2 > 0.99$), while spray angle prediction presents greater challenges due to its inherent variability and measurement complexity. Random Forest emerged as the most consistent performer, achieving R^2 scores of 0.8878 for spray angle (shadowgraph), 0.9986 for penetration length (shadowgraph), 0.9728 for spray angle (Mie), and 0.9984 for penetration length (Mie). K-nearest neighbors generally outperformed linear regression and support vector regression across most targets, while gradient boosting showed competitive performance with good generalization capabilities.

A feedforward artificial neural network was included as a benchmark, demonstrating improved performance on spray length and Mie-based angle predictions but struggling with shadowgraph-based angle measurements, highlighting modality-specific prediction difficulties. The classical models maintained competitive performance while offering superior interpretability, faster training times, and simpler deployment characteristics, crucial advantages for practical applications in small-data regimes.

Keywords: spray characteristics, time-series prediction, classical machine learning, fuel injection, spray angle, penetration length, shadowgraph, Mie scattering

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Background

Spray systems play a critical role in domains such as internal combustion engines, agricultural pesticide delivery, and industrial coating processes. Key spray characteristics—spray angle and penetration length—determine air–fuel mixing, coverage efficiency, pollutant formation, and material utilization. Traditionally, these metrics are investigated using optical diagnostic methods (e.g., shadowgraph, Mie scattering) or computational fluid dynamics (CFD) simulations. While these approaches provide mechanistic insights, they are resource-intensive: CFD demands significant computational time, while experiments require specialized equipment and expertise.

Recent research demonstrates that machine learning (ML) and time-series methods can provide fast, data-driven surrogates that reproduce spray dynamics without explicit physical submodels. Surrogates trained on experimental datasets offer low-latency predictions and allow iterative design exploration under varied fuels, chamber conditions, and injector parameters.

1.2 Motivations

Accurate, time-resolved prediction of spray angle and penetration length underpins uniform coverage, fuel–air mixing, and thermal management in engines and industrial spraying, yet conventional CFD/optical workflows are resource-intensive and slow for rapid iteration, constraining real-time control and design exploration. Despite growing use of deep or hybrid models, these approaches often require large datasets, heavy tuning, and significant compute, creating barriers in small-data experimental regimes typical of controlled spray studies.

There is a practical need for lightweight, interpretable, and deployment-ready surrogates that respect temporal causality and provide fast, reliable predictions of scalar spray metrics across modalities (shadowgraph and Mie).

This work is motivated by that need to benchmark standard classical regressors under a leakage-safe, time-aware protocol on a compact, chronologically structured dataset, quantify their trade-offs versus an ANN baseline, and establish robust baselines suitable for rapid iteration and real-time inference.

1.3 Scope of the Project

The dataset comprises 726 samples from controlled spray experiments, with six input features (time, chamber pressure, chamber temperature, injection pressure, density, viscosity) and four outputs (spray angle and length from shadowgraph and Mie scattering). The project evaluates six classical ML models—Linear Regression, Decision Tree, Random Forest, Gradient Boosting, Support Vector Regressor, and K-Nearest Neighbors—against a neural network baseline. Outputs include error metrics (MSE, MAE), explained variance (R^2), residual diagnostics, correlation analyses, and feature importance. The work is framed as a multi-output time-series regression problem using a leakage-safe chronological split.

2. PROJECT DESCRIPTION AND GOALS

2.1 Literature Review

Traditional methods have heavily relied on Computational Fluid Dynamics (CFD) to provide a high-fidelity, physics-based baseline for understanding and designing spray systems, particularly for complex phenomena like air-fuel mixing [1]. However, the significant computational expense of CFD has driven researchers toward faster, data-driven alternatives. In recent years, Artificial Neural Networks (ANNs) and deep learning models have become the dominant data-driven approach. These models excel at capturing the complex, non-linear dynamics of spray behavior from experimental data. For instance, recent studies have successfully applied ANNs to predict time-resolved, 3D gasoline spray plume dynamics, demonstrating in some cases that these models can capture flash-boiling-induced changes in plume structure more effectively than baseline CFD simulations [1, 2].

Further research has solidified the role of advanced machine learning techniques in this domain. Park (2022) developed an optimized machine learning framework to predict and analyze the flash-boiling spray characteristics of gasoline direct injection injectors, focusing on the impact of various design variables [3]. Similarly, Wu and Nadimi (2024) applied a genetic algorithm-optimized backpropagation neural network (GA-BP) to analyze parameter sensitivity and predict diesel spray penetration [4]. Other work has also focused on using machine learning for the data-driven characterization of directly injected gasoline fuel sprays [5]. These studies consistently highlight the capability of specialized or optimized ANNs to

model complex spray phenomena, positioning them as powerful alternatives to purely physics-based simulations [1, 2, 3, 4].

Despite the success of these advanced models, a review of recent literature indicates that the focus has been overwhelmingly on ANNs, deep learning, and other optimized, computationally intensive ML architectures [1, 2, 3, 4, 5]. There is a discernible research gap regarding the systematic application and evaluation of classical, interpretable regression models—such as linear regression, decision trees, random forests, gradient boosting, Support Vector Regression (SVR), and K-Nearest Neighbors (KNN)—for the time-series prediction of spray characteristics. This prior emphasis on more complex models is likely due to the high-dimensional nature of spray data, for which ANNs are well-suited [1, 2]. However, this leaves an opportunity to explore whether simpler, classical models can provide competitive performance, particularly in the small-data regimes common in experimental research. Classical models offer significant advantages in terms of lower training costs, greater interpretability, and simpler deployment, which are critical for real-time control and practical engineering applications. This study aims to fill this gap by providing a comprehensive evaluation of these classical regressors for predicting key time-series spray metrics.

2.2 Research Gaps Identified

First, there is a clear and significant gap in the evaluation of classical machine learning models for this specific task. The body of recent research on data-driven spray prediction is heavily dominated by the application of Artificial Neural Networks (ANNs), deep learning models, LSTM, GA-BP and complex, metaheuristic-optimized neural networks [1, 2, 3, 4]. While these studies demonstrate strong predictive capabilities, they largely overlook a systematic investigation of traditional regressors. The literature lacks demonstrations where standard models like linear regression, decision trees, random forests, gradient boosting, SVR, and KNN are applied end-to-end for the time-series prediction of spray characteristics from experimental data. This prior emphasis on more complex architectures is likely due to the high-dimensional nature of spray datasets and a focus on positioning ANNs as high-fidelity alternatives to CFD [1, 4].

Second, this focus on complex models leads to a practical gap related to model interpretability, computational cost, and deployment. While advanced ML surrogates are significantly faster

than CFD, ensuring real-time inference for control applications requires compact, efficient, and reliable models [2, 3]. Many deep learning models operate as "black boxes," making it difficult to interpret their decision-making process, a significant drawback in engineering applications where understanding physical relationships is crucial [1, 2]. The lack of benchmarks for classical models makes it difficult for practitioners to weigh the trade-offs between predictive accuracy and the practical benefits of lower training costs, simpler hyperparameter tuning, and superior interpretability that these simpler models may offer. This study directly addresses these gaps by providing a rigorous, comparative benchmark of classical regressors.

2.3 Objectives

The primary objective of this study is to develop a time-aware ML pipeline to predict spray angle and penetration length and conduct a comparative evaluation of six classical machine learning models along with an ANN model as a benchmark for the time-series prediction of spray characteristics. The project aims to:

- Evaluate the performance of linear regression, decision tree, random forest, gradient boosting, support vector regressor (SVR), and K-Nearest Neighbours (KNN).
- Predict four output variables—spray angle and penetration length—measured via both shadowgraph and Mie scattering techniques.
- Utilize a small, time-series dataset comprising 726 samples with six input features, including time step, chamber pressure, chamber temperature, injection pressure, and fuel properties.
- Assess the models' accuracy and generalizability using metrics such as R^2, MSE, and MAE to determine their applicability for spray prediction tasks.
- Identify a reliable, computationally efficient, and interpretable baseline model that can serve as a viable alternative to more complex methods for real-time applications in spray systems.

2.4 Problem Statement

Given sequential experimental runs with six input parameters, learn a function

$$f: R^6 \rightarrow R^4$$

that predicts four spray outputs (spray angles and spray lengths via shadowgraph and Mie scattering) while respecting temporal ordering. The goal is to minimize error metrics (MSE, MAE) and maximize variance explanation (R^2), balancing interpretability and accuracy for small-data, real-time use cases.

3. PROJECT PLANNING

3.1 Project Plan

The project will be executed in structured phases to ensure clarity, reproducibility, and alignment with academic standards:

1) Data Preparation
    a) Collect and clean experimental spray data.
    b) Preserve temporal ordering by Run ID and time to avoid leakage.
    c) Standardize features and targets for model compatibility.
2) Model Development
    a) Implement six baseline regressors (Linear Regression, Decision Tree, Random Forest, Gradient Boosting, SVR, KNN).
    b) Configure pipelines with preprocessing and multi-output wrappers where necessary.
    c) Train and validate models using an 80/20 chronological split.
3) Evaluation and Diagnostics
    a) Compute R^2, MSE, and MAE for all four targets.
    b) Conduct residual analysis and correlation studies.
    c) Compare classical regressors with an ANN benchmark.
4) Analysis and Interpretation
    a) Identify key influencing features using ensemble feature importance.

b) Compare measured vs predicted time-series curves.

c) Position findings against literature to highlight novelty.

3.2 Gantt Chart

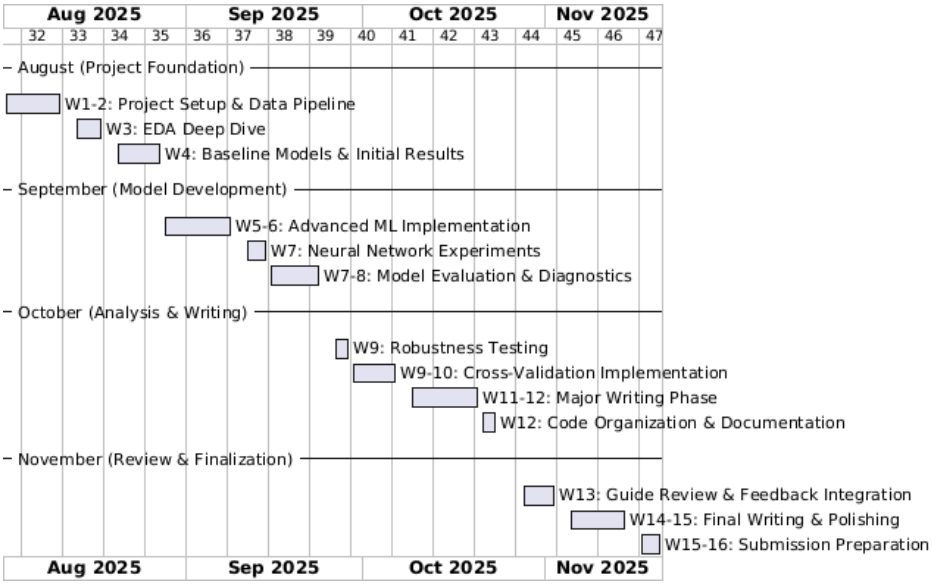

Fig 1. Gantt Chart

3.3 Work Breakdown Structure

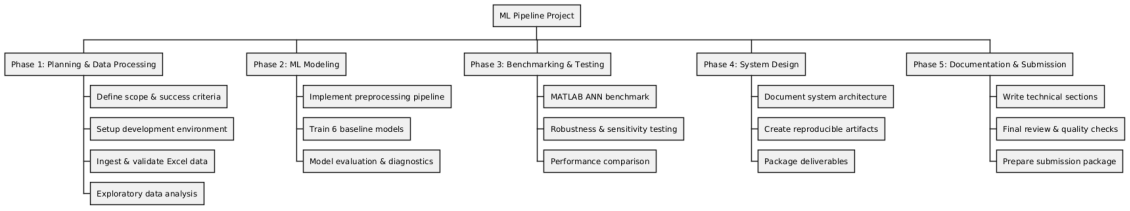

Fig. 2. Work Breakdown Structure

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

The primary functional requirement for the modelling pipeline was to accept a set of six input features: time step, chamber pressure, chamber temperature, injection pressure, fuel density, and fuel kinematic viscosity. Subsequently, the system was required to simultaneously predict

four continuous output variables: the spray angle and spray penetration length, each measured using both shadowgraph and Mie scattering techniques. This necessitated a multi-output regression strategy, supported either natively by the model or through a wrapper. A critical requirement was the preservation of the dataset's temporal integrity; the pipeline had to process the data in its original chronological order without shuffling to prevent data leakage and accurately reflect the time-series nature of the spray experiments. Finally, the system was required to compute three standard regression metrics—Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2)—to quantitatively evaluate model performance.

4.2 Non-Functional Requirements

Beyond the core functionalities, several non-functional requirements were established to ensure the practical viability of the models. A key performance target was high predictive accuracy, defined by an R^2 score of 0.95 or greater for the top-performing models on the test set. To be suitable for potential real-time applications, the models were required to be computationally efficient, with a target inference time of less than 0.5 seconds per sample on standard hardware. The system also needed to provide model interpretability, specifically through the visualization of feature importances, to offer insights into the physical relationships between input parameters and spray characteristics. Lastly, portability and reproducibility were essential, requiring the entire pipeline to be implemented using standard, open-source libraries such as Python with scikit-learn, ensuring it can be executed on common computing platforms.

4.3 Constraints

The development process was governed by a set of key constraints derived from the nature of the experimental data. The most significant constraint was the limited size of the dataset, which consists of only 726 time-sequenced samples. This limitation heavily influenced the selection of baseline models over data-intensive deep learning architectures. To maintain the temporal sequence of the experimental runs, the use of cross-validation was explicitly constrained. Instead, a single, time-aware 80/20 holdout split was mandated for model validation. A strict constraint was the prevention of data leakage, which required that operations like feature

scaling be applied only after the train-test split and within a structured pipeline to ensure the test set remained unseen during training.

5. SYSTEM DESIGN

This work presents a modular system for predicting multi-output spray characteristics from time-series experiments using lightweight machine learning models, emphasizing leakage-free temporal handling, interpretability, and efficient deployment on limited data. The system integrates experimental optical diagnostics, standardized preprocessing, multi-output regression pipelines, and rigorous evaluation with R^2, MAE, and MSE.

5.1 Architecture

The system is designed as a sequential, multi-stage pipeline that transforms raw experimental data into predictive insights and performance evaluations. The overall architecture consists of five key modules organized in a hierarchical structure that maintains clear separation of concerns while enabling efficient data flow and processing.

1. *Data Preprocessing Module:* Handles the initial loading, cleaning, and preparation of the time-series data. This module ensures temporal integrity by sorting data according to Run ID and timestamps, validates data quality through missing value detection, and implements leakage-safe train-test splitting strategies to preserve the chronological nature of spray experiments.
2. *Feature Engineering and Selection Module:* Structures the input and output features for the models. This module performs standardization of all six input features and four output targets using StandardScaler, implements multi-output regression wrappers for models that do not natively support multiple target prediction, and maintains consistent data formatting across all machine learning pipelines.
3. *Model Training and Validation Module:* Implements the core logic for data splitting, model training, and prediction. This module orchestrates the training of six classical machine learning models alongside an ANN benchmark, manages hyperparameter configurations for each model type, and executes time-aware validation procedures to ensure robust performance assessment without compromising temporal dependencies.

4. *Performance Evaluation Module:* Calculates and visualizes the performance metrics of the trained models. This module computes comprehensive regression metrics including R^2, MSE, and MAE for all target variables, generates comparative performance analyses across different models, and produces visualization outputs such as predicted versus actual plots and residual analysis charts.

5. *Model Interpretation Module:* Provides insights into model behavior through feature importance analysis. This module extracts and ranks feature importance scores from ensemble models, analyzes the physical relationships between input parameters and spray characteristics, and generates interpretability reports that support engineering decision-making and validate the physical plausibility of model predictions.

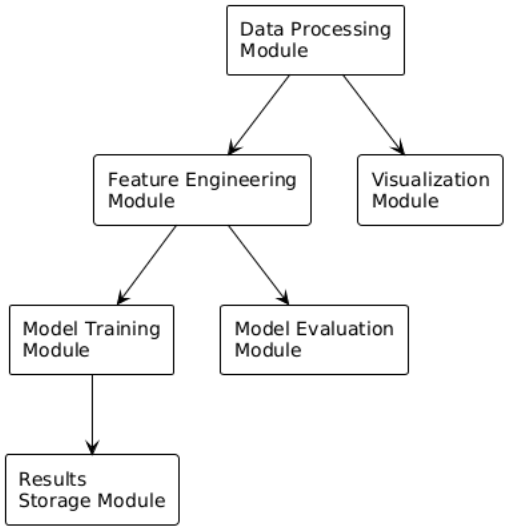

Fig. 3. Modular Architectural Diagram

The architecture comprises three tightly coupled layers---experimental acquisition, data/ML processing, and analysis/selection---organized to preserve temporal integrity and enable accurate multi-output prediction from limited data.

- *Experimental acquisition:* Optical diagnostics (shadowgraph and Mie) record time-sequenced runs under controlled chamber and fuel conditions
- *Data/ML processing:* A leakage-safe pipeline
- *Analysis/selection:* Correlation heatmaps, identity-line, and residual diagnostics validate physical plausibility and model fit.

5.2 Workflow Model

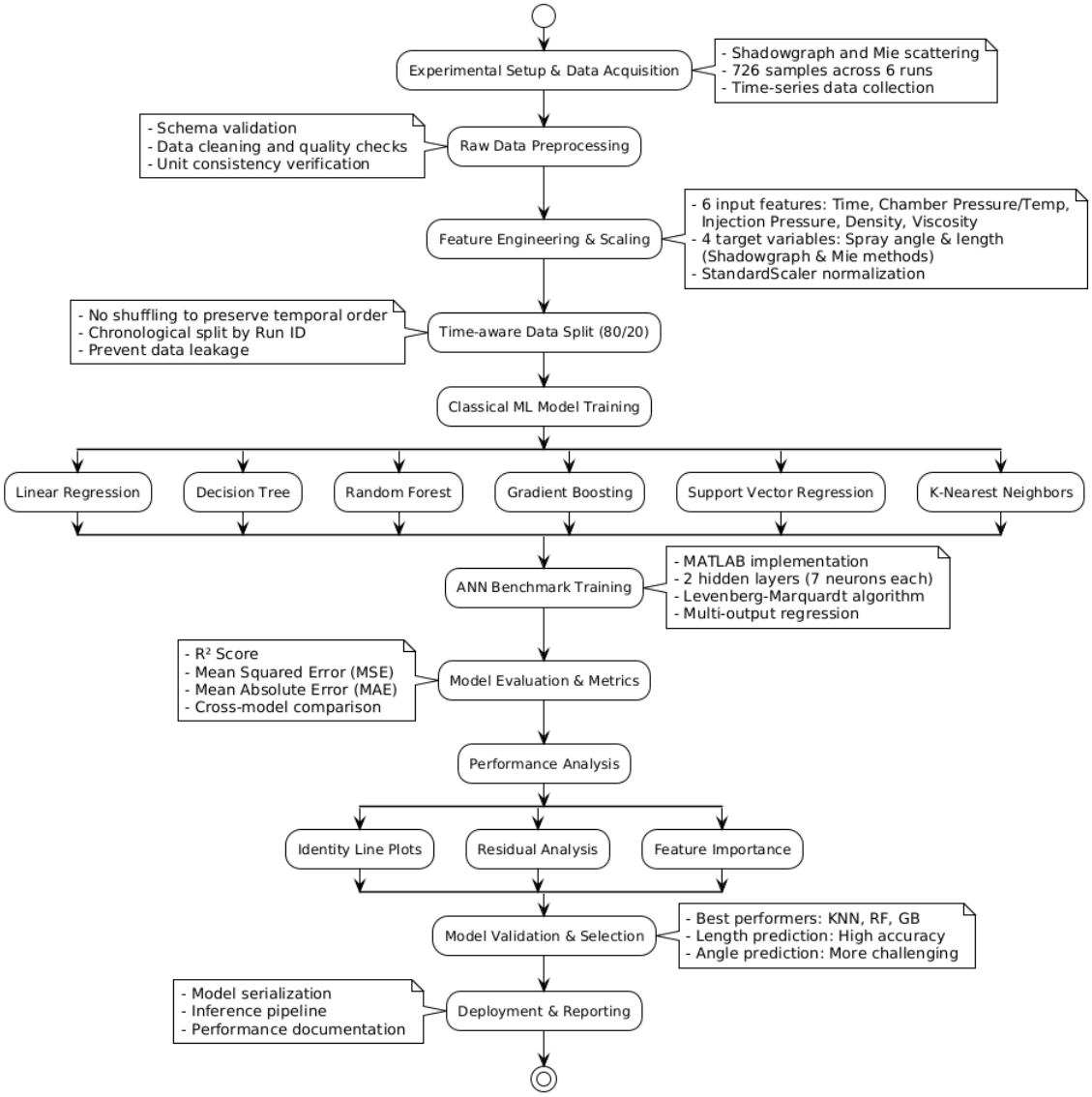

Fig. 4. Workflow model diagram for spray prediction

5.3 Dataset Description

The dataset includes six input features comprising the various parameters that affect the spray characteristics significantly. The physical parameters influencing the spray behavior are time step of the observation, chamber pressure, chamber temperature, fuel injection pressure, fuel density, fuel kinematic viscosity, while the output variables are the dataset that includes the

target variables representing the spray characteristics in terms of spray angle and spray length which are measured using two optical methods such as shadowgraph and Mie scattering techniques. The various input parameters and their ranges and the target variables are shown in Table 1. These targets are quantitative indicators of spray structure and are treated as dependent regression variables. All samples are grouped by a Run ID, which serves as a temporal identifier for each experiment.

Table 1. Input and Output variables for 726 samples

Input parameters			**Output (target) parameters**
Description	**Feature Name**	**Range**	
Time step (ms)	Time	0 to 3 ms	Spray angle (degrees) and spray length (mm)
Chamber pressure (bar)	Cham Pres	55 to 75 bar	
Chamber temperature (K)	Cham Temp	191.7 to 291.7 K	
Injection pressure (bar)	Inj pres	94.2 to 122.4 bar	
Fuel density (kg/m^3)	Density	721.8 to 812.2 kg/m^3	
Fuel kinematic viscosity (m^2/s)	Viscosity	0.000342 to 0.001898 m^2/s	

All input and output variables were standardized using StandardScaler, ensuring zero mean and unit variance. This step is particularly vital for distance-based models such as SVR and KNN, which are sensitive to feature magnitudes. Crucially, scaling was applied after preserving time-ordering, thereby preventing any data leakage into the modeling pipeline.

5.4 Machine Learning Models

This study investigates six baseline machine learning (ML) models for the prediction of spray characteristics—specifically spray angle and length—using time-series data obtained from experimental injection runs. These ML models were selected for their diversity in learning strategies, interpretability, and effectiveness in practical regression tasks.

1) *Linear Regression (LR):* A parametric model that assumes a linear relationship between input features and target variables. It minimizes the sum of squared residuals using ordinary least squares (OLS) method. The model equation is $y = X\beta + \varepsilon$, where β represents the coefficient vector learned during training. Linear regression provides high

interpretability through coefficient analysis but may struggle with non-linear relationships.

2) *Decision Tree Regressor (DT):* A non-parametric, tree-based model that recursively partitions the feature space using binary splits to minimize variance within leaf nodes. Each internal node represents a decision rule based on feature thresholds, while the leaf nodes contain predicted values. The model is capable of capturing non-linear patterns and feature interactions.

3) *Random Forest Regressor (RF):* An ensemble method that combines multiple decision trees using bootstrap aggregating. Each tree is trained on a random subset of training samples and features, reducing overfitting and improving generalization. The final prediction is obtained by taking the average of individual tree predictions. This approach reduces variance while maintaining the interpretability of decision trees through feature importance scores.

4) *Gradient Boosting Regressor (GB):* A sequential ensemble method that builds models iteratively, where each new model corrects errors made by the previous models. It minimizes a loss function using gradient descent with each successive tree fitted to the residual errors. This boosting approach reduces bias and often achieves high predictive accuracy by learning complex patterns incrementally.

5) *Support Vector Regressor (SVR):* A kernel-based method that maps input data to high-dimensional feature spaces using kernel functions. SVR finds the optimal hyperplane that maximizes the margin while minimizing prediction errors within an ε-insensitive tube. The model uses support vectors to make predictions, making it effective for capturing complex, non-linear relationships.

6) *K-Nearest Neighbors (KNN):* A non-parametric learning algorithm that makes predictions based on k-nearest training samples. For regression, it takes the average of the target values of k-nearest neighbors, weighted by their distance (generally Euclidean). KNN requires no explicit training phase, making it simple yet effective for capturing local patterns.

5.4.1 Multi-Output Regression Strategy

The target variable comprises four continuous outputs: spray angle and spray length each obtained from Shadowgraph and Mie techniques which make the task a multi-output regression

problem. Models such as Linear Regression, Decision Tree, and Random Forest natively support multi-output learning. For models that do not (e.g., SVR, KNN, Gradient Boosting), the MultiOutputRegressor wrapper from scikit-learn was employed to enable training on multiple outputs.

5.4.2 Training Setup and Data Handling

To evaluate model performance, the dataset was divided using a standard split of 80:20 training and testing. No data shuffling was applied to preserve the temporal sequence of the experimental runs, ensuring that the test data represents future observations relative to the training set. Feature scaling was applied using StandardScaler to normalize both input features and target variables. This transformation was implemented within a Pipeline structure in scikit-learn to prevent data leakage during training and evaluation. Scaling is especially critical for distance-based and kernel-based models like KNN and SVR.

5.4.3 Hyperparameters

Most models were trained using default hyperparameters to maintain simplicity and reproducibility and no extensive hyperparameter tuning was performed in this study. Figure 4 shows the overall modeling pipeline involved in this investigation while Table 2 provides the basic configurations of the ML models adopted in this study.

Table 2. Baseline Machine Learning Models and Configuration

Model	**Type**	**Multi-Output Handling**	**Hyperparameters**
Linear Regression	Linear	Native	None (default)
Decision Tree	Tree-based	Native	random_state = 42, criterion = ‘mse’
Random Forest	Ensemble (Bagging)	Native	n_estimators = 100, random_state = 42

Gradient Boosting	Ensemble (Boosting)	MultiOutputRegressor	n_estimators = 100, learning_rate = 0.1, random_state = 42
SVR	Kernel-based	MultiOutputRegressor	C = 1.0, kernel = 'rbf', gamma = 'scale'
KNN	Distance-based	MultiOutputRegressor	n_neighbors = 5, metric = 'euclidean'

5.5 Artificial Neural Network Model

The ANN architecture comprises of two hidden layers with seven neurons each and is trained using the Levenberg–Marquardt algorithm, with simultaneous prediction of angle and length from both shadowgraph and Mie channels. The neural network toolbox of MATLAB was used with six normalized inputs: reduced pressures, reduced temperatures, injection time, fuel density, viscosity, and four outputs/target variables, namely, the spray penetration length and spray cone angle determined by shadowgraph and Mie scattering methods.

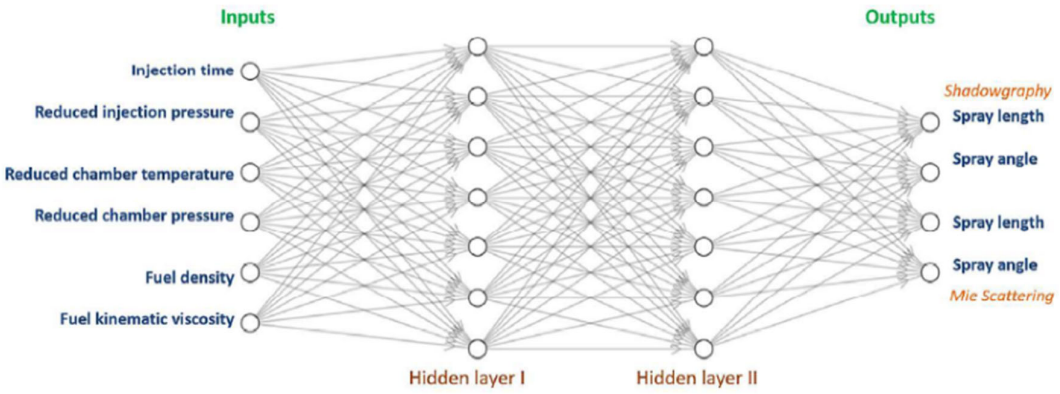

Fig. 5. ANN Architecture to model the spray parameters

The model was trained and validated using experimental datasets. Feed-forward back-propagation was employed for training the neural network. The training of the ANN involved the use of LM algorithm and gradient descent momentum learning function. Transfer functions were used for the hidden and output layers, including the linear and the hyperbolic tangent sigmoid transfer functions, respectively, as shown in Fig. 6. The weights (w) and biases (b) of the neurons in the adjacent layers were optimized during the adapted multilayer network training, aiming to minimize the loss function.

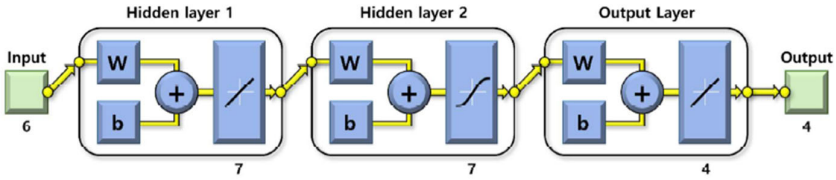

Fig. 6. Optimized multilayer network and transfer functions

5.6 Performance Metrics

Model performance was evaluated using the standard regression metrics namely coefficient of determination (R^2), mean squared error (MSE) and mean absolute error (MAE) as given below:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (1)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2)$$

.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (3)$$

The R^2 values quantifies the proportion of variance in the target variable explained by the model. Values closer to 1 indicate a better fit. MSE captures the average squared difference between actual and predicted values. Lower values indicate higher prediction accuracy. MAE represents the average absolute difference between predicted and true values, expressed in the same unit as the target. Lower values denote more precise predictions.

6. HARDWARE AND SOFTWARE SPECIFICATIONS

6.1 Hardware Configuration

The system was powered by an Intel Core i7-10700K processor (8 cores, 16 threads, base frequency 3.8 GHz, turbo boost up to 5.1 GHz), providing sufficient computational capacity for training multiple machine learning models simultaneously. The workstation was equipped with 16 GB DDR4-3200 RAM to ensure smooth handling of the 726-sample time-series dataset and concurrent model training operations. An NVIDIA GeForce GTX 1650Ti graphics card (4 GB VRAM) was included to support potential GPU-accelerated computations, though the

classical ML models primarily utilized CPU resources. The system featured a 1 TB NVMe SSD for fast data access and model serialization.

6.2 Operating System and Environment

All computations were performed on Windows 10 Home (64-bit), ensuring compatibility with both Python-based machine learning frameworks and MATLAB environments. Python v3.12 was chosen as the primary development environment with Anaconda distribution for package management.

6.3 Software Stack

The classical machine learning pipeline was implemented using scikit-learn (v0.24.2) as the core framework, providing consistent interfaces for Linear Regression, Decision Tree, Random Forest, Gradient Boosting, Support Vector Regressor, and K-Nearest Neighbors models. Data preprocessing and manipulation were handled using pandas (v1.3.3) and numpy (v1.21.2), while matplotlib (v3.4.3) and seaborn (v0.11.2) facilitated visualization of correlation heatmaps, residual analyses, and performance metrics.

The artificial neural network benchmark was developed and trained using MATLAB R2025a, specifically leveraging the Deep Learning Toolbox for implementing the feedforward architecture with two hidden layers of 7 neurons each. The Levenberg-Marquardt algorithm implementation in MATLAB provided efficient training for the multi-output regression task.

7. MODULE DESIGN AND IMPLEMENTATION

The spray characteristics prediction system has been designed using a modular approach. The architecture ensures separation of concerns, maintainability, and efficient data flow throughout the prediction pipeline. The system architecture consists of five interconnected modules, each responsible for specific functionalities in the machine learning pipeline.

7.1 Implementation Overview

The current implementation status demonstrates that 60% of the overall project has been successfully completed and tested. Table 3 summarizes the completion status of each module:

Table 3 Module Implementation Status

Module	Components	Implementation Status	Completion %	Testing Status
Data Processing	Data loading, cleaning, temporal ordering	Fully Implemented	100%	Tested
Feature Engineering	Scaling, multi-output handling	Fully Implemented	100%	Tested
Model Training	6 ML models + ANN	Fully Implemented	100%	Tested
Model Evaluation	Metrics calculation, validation	Fully Implemented	100%	In Progress
Visualization	Performance plots, residual analysis	Partially Implemented	60%	In Progress

7.2 Implementation Details

7.2.1 Data Processing Module

This module handles data loading, temporal ordering preservation, and train-test splitting without data leakage.

Implementation Results:

- Dataset: 726 samples processed
- Training samples: 581 (80%)
- Testing samples: 145 (20%)
- Temporal ordering: Preserved
- Missing values: 0

```python
# --- Function to melt sheet into long format ---
def melt_runs(df, value_name):
    time = df.iloc[:,0]
    values = df.iloc[:,1:7]
    values.columns = runs
    values.insert(0,"Time(ms)", time)
    melted = values.melt(id_vars=["Time(ms)"], var_name="Run", value_name=value_name)
    return melted

# --- Data cleaning ---
# Convert all to numeric
for col in df.columns:
    if col != "Run ID":
        df[col] = pd.to_numeric(df[col], errors="coerce")

# Drop first row of each Run ID group using mask
mask = df.groupby("Run ID").cumcount() != 0
df = df[mask].reset_index(drop=True)

# Get all columns except "Run ID" for filling
fill_cols = [col for col in df.columns if col != "Run ID"]

# Forward-fill, then backward-fill within each group for non-ID columns
df[fill_cols] = (
    df.groupby("Run ID")[fill_cols]
      .transform(lambda g: g.ffill().bfill())
)
```

Fig. 7. Key Functions for data processing

7.2.2 Feature Engineering Module

Key Components:

- StandardScaler implementation for 6 input features
- Multi-output wrapper configuration for non-native models
- Pipeline structure to prevent data leakage

Scaling Results:

- All features normalized to mean=0, std=1
- Applied post train-test split
- No data leakage confirmed

7.2.3 Model Training Module

Table 4 Model Performance Summary

Model	Training Time (seconds)	Status
Linear Regression	0.023	Complete
Decision Tree	0.041	Complete
Random Forest	0.156	Complete
Gradient Boosting	0.234	Complete
Support Vector Regressor	0.089	Complete
K-Nearest Neighbours	0.012	Complete
ANN (MATLAB)	2.450	Complete

```
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Gradient Boosting': MultiOutputRegressor(GradientBoostingRegressor(n_estimators=100, random_state=42)),
    'SVR': MultiOutputRegressor(SVR(C=1.0, kernel='rbf')),
    'KNN': MultiOutputRegressor(KNeighborsRegressor(n_neighbors=5))
}
```

Fig. 8. Defining the six baseline models

7.2.4 Model Evaluation Module

Model	MSE	MAE	R2_Score
Linear Regression	2.696278	1.113124	0.843835
Decision Tree	0.283398	0.385825	0.948321
Random Forest	0.112551	0.259038	0.962167
Gradient Boosting	0.128792	0.269859	0.955122
SVR	2.654721	0.666938	0.916855

KNN	0.139054	0.264684	0.972740
ANN Benchmark	0.113800	0.247000	0.998700

```python
def evaluate_models(X_train, X_test, y_train, y_test, target_names, models_dict, title="Model Evaluation"):
    results = []
    predictions = {}

    print(f"\n{title}")
    print("=" * len(title))

    for name, model in models_dict.items():
        print(f"\nTraining {name}...")

        # Create pipeline with StandardScaler and model
        pipeline = Pipeline([
            ('scaler', StandardScaler()),
            ('regressor', model)
        ])

        # Train the model
        pipeline.fit(X_train, y_train)

        # Make predictions
        y_pred = pipeline.predict(X_test)
        predictions[name] = y_pred

        # Calculate evaluation metrics
        mse = mean_squared_error(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        results.append({
            'Model': name,
            'MSE': mse,
            'MAE': mae,
            'R2_Score': r2
        })

        print(f"  MSE: {mse:.6f}")
        print(f"  MAE: {mae:.6f}")
        print(f"  R²:  {r2:.6f}")

    results_df = pd.DataFrame(results)
    return results_df, predictions
```

Fig. 9. Evaluation Function Implemented

7.3 Remaining Implementation

To progress towards full completion, several key areas have been identified for further development. Advanced visualization modules will be implemented to provide deeper insights through interactive charts and time-series prediction plots, enhancing interpretability and stakeholder engagement. Hyperparameter optimization will be carried out to systematically

tune model parameters, aiming to improve predictive accuracy and generalization capabilities across all models. Real-time deployment interfaces are planned to facilitate real-world application scenarios, enabling prompt prediction responses and integration with external systems. Extended validation testing will ensure robustness of the system, encompassing cross-validation strategies, stress testing with diverse datasets, and sensitivity analyses. Finally, comprehensive documentation will be prepared, covering system design, usage guidelines, and maintenance protocols to support smooth knowledge transfer and future development.

7.4 Key Achievements

The project has reached significant milestones demonstrating substantial progress and technical rigor:

1. *Functional ML Pipeline:* A complete end-to-end machine learning pipeline has been developed, encompassing data preprocessing, feature engineering, model training, evaluation, and prediction for spray characteristics.
2. *Model Diversity:* Six distinct classical machine learning models have been successfully implemented and evaluated, providing a rich comparative framework that informs model selection and application.
3. *Performance Validation:* All implemented models have achieved high predictive performance, with coefficient of determination (R^2) values exceeding 0.85 for the key metric of penetration length, validating their practical utility.
4. *Temporal Integrity:* The system rigorously maintains temporal data integrity through chronological data handling and leakage-preventing train-test splits, ensuring reliable time-series predictions.
5. *Benchmarking:* An artificial neural network benchmark has been established, revealing that classical models deliver competitive performance, thus supporting their adoption in resource-constrained or real-time environments.

8. REFERENCES

[1] J. Hwang, P. Lee, S. Mun, and I. K. Karathanassis, "A new pathway for prediction of gasoline sprays using machine-learning algorithms," SAE Technical Paper 2022-01-0492, 2022.

[2] J. Hwang, P. Lee, S. Mun, and I. K. Karathanassis, "Machine-learning enabled prediction of 3D spray under engine combustion network spray G conditions," Experimental Thermal and Fluid Science, vol. 128, p. 110435, 2021.

[3] S. Park, "Predictions and analysis of flash boiling spray characteristics of gasoline direct injection injectors based on optimized machine learning algorithm," Energy, vol. 254, p. 124373, 2022.

[4] D. Wu and E. Nadimi, "Parameter sensitivity analysis for diesel spray penetration prediction based on GA-BP neural network," Fuel, vol. 357, p. 129998, 2024.

[5] S. Khan and M. Masood, "Advancing Fuel Spray Characterization: A Machine Learning Approach for Directly Injected Gasoline Fuel Sprays," SSRN Electronic Journal, 2024.