Application Project (Group 28)
Milestone 1

Hongyi Zhang[1], Ziang Deng[1], and Nancy Wang[1]

[1] *McKelvey School of Engineering, Washington University in St. Louis, USA*
hongyi.zhang@wustl.edu, d.ziang@wustl.edu, nancy.wang@wustl.edu

1 Introduction

In this milestone, our goal is to gain initial insights into the provided dataset through a series of structured tasks, including data exploration, data preprocessing, baseline modeling, and clustering.

We begin with **Data exploration**, where we compute basic statistics, visualize the distributions of the target variable and features, and investigate the correlations between features and the target. We also examine relationships between features themselves using correlation matrices and scatter plot matrices.

Following the exploratory data analysis, we perform **Data Preprocessing**, including identifying and handling missing values, outliers, feature rescaling, and splitting the dataset to prepare the dataset for modeling.

In the **Baseline Modeling** phase, we fit 3 baseline models - Ridge Regression, Random Forest Regressor, and Neural Network Regressor to evaluate their initial performance on the dataset by using MSE and R-square. Additionally, we analyze their error distributions through paired t-tests and kernel density estimation.

Finally, we perform **Clustering** on the data with all features, using k=2. We use a linear regression model to predict the target. There is a modest improvement by incorporating the **distances to each cluster center** as additional features.We try different values of k and recluster, the elbow method is employed to determine the optimal number of clusters (k) for the K-means algorithm.

2 Data Exploration

2.1 Target Statistics and KDE

The dataset consists of 40 features and 1 target column. We first computed the descriptive statistics for the target column:

Table 2.1: Descriptive Statistics for Target Column

Statistic	Value
Count	8250
Mean	0.8672
Standard Deviation	0.4050
Minimum	0.0000
25th Percentile	0.6000
Median	0.8000
75th Percentile	1.1000
Maximum	3.5000
Range	3.5000

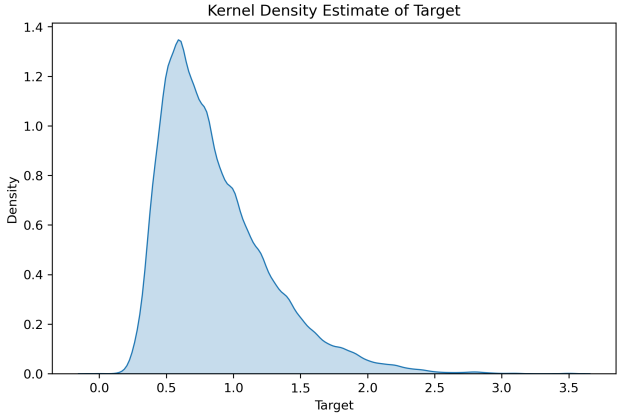

Figure 2.1: Kernel Density Estimate of Target

For the target variable, we plotted kernel density estimates (KDE) to visualize their distributions, as shown in Figure 2.1. The target variable shows a right-skewed distribution, with the majority of values concentrated between 0.5 and 1.0. The distribution has a peak near 0.6, followed by a gradual decline and a long tail extending toward larger values.

2.2 Feature Statistics and KDE

For each of the 40 features, we computed:

Table 2.2: Descriptive Statistics of Features (Simplified)

Feature	Mean	Std	Min	Max	Range
acc_rate	-1.152606e+01	259.637258	-975.000000	946.000000	1921.000000
track	-1.267297e+01	25.675733	-95.000000	94.000000	189.000000
m	1.051280e+00	0.320703	0.216536	2.886371	2.669835
n	6.038667e-02	0.118805	-0.530000	0.620000	1.150000
current_pitch	6.293709e-01	0.313628	-0.420000	2.580000	3.000000
current_roll	6.124848e-02	0.967274	-3.000000	2.900000	5.900000
absoluate_roll	-1.100485e+01	4.140399	-23.000000	-3.000000	20.000000
climb_delta	-9.203636e-01	10.334136	-44.000000	46.000000	90.000000
roll_rate_delta	-9.567273e-04	0.013203	-0.080000	0.056000	0.136000
climb_delta_diff	-4.784242e-02	1.132179	-8.800000	8.700000	17.500000
time1	2.187236e-02	0.006873	0.012000	0.078000	0.066000
time2	2.188558e-02	0.006906	0.012000	0.078000	0.066000
time3	2.188558e-02	0.006906	0.012000	0.078000	0.066000
time4	2.189782e-02	0.006914	0.012000	0.078000	0.066000
time5	2.189794e-02	0.006914	0.012000	0.078000	0.066000
time6	2.191770e-02	0.006924	0.012000	0.078000	0.066000
time7	2.191818e-02	0.006925	0.012000	0.078000	0.066000
time8	2.193745e-02	0.006933	0.012000	0.076000	0.064000
time9	2.193745e-02	0.006933	0.012000	0.076000	0.064000
time10	2.195418e-02	0.006939	0.012000	0.074000	0.062000
time11	4.390739e-02	0.013876	0.024000	0.148000	0.124000
time12	2.196885e-02	0.006953	0.012000	0.074000	0.062000
time13	2.196921e-02	0.006954	0.012000	0.074000	0.062000
time14	2.198194e-02	0.006960	0.012000	0.073000	0.061000
time1_delta	-1.320000e-04	0.000695	-0.005000	0.004000	0.009000
time2_delta	-1.212121e-07	0.000011	-0.001000	0.000000	0.001000
time3_delta	-6.266667e-05	0.000462	-0.006000	0.004000	0.010000
time4_delta	-3.636364e-07	0.000025	-0.002000	0.000000	0.002000
time5_delta	-8.072727e-05	0.000491	-0.008000	0.002000	0.010000
time6_delta	-7.272727e-07	0.000049	-0.004000	0.000000	0.004000
time7_delta	-1.774545e-05	0.000113	-0.001000	0.000600	0.001600
time8_delta	-1.212121e-07	0.000011	-0.001000	0.000000	0.001000
time9_delta	-9.890909e-05	0.000615	-0.006000	0.003000	0.009000
time10_delta	4.848485e-07	0.000070	-0.002000	0.006000	0.008000
time11_delta	8.999901e+00	0.000631	8.995000	9.005000	0.010000
time12_delta	-7.272727e-07	0.000060	-0.004000	0.002000	0.006000
time13_delta	-9.309091e-05	0.000613	-0.005000	0.007000	0.012000
time14_delta	-1.000000e+01	0.000037	-10.003000	-10.000000	0.003000
omega	-5.102791e-01	0.257113	-0.916291	0.693147	1.609438
set	2.198218e-02	0.006961	0.012000	0.073000	0.061000

The distributions of all features were visualized using Kernel Density Estimates (KDE), as shown in

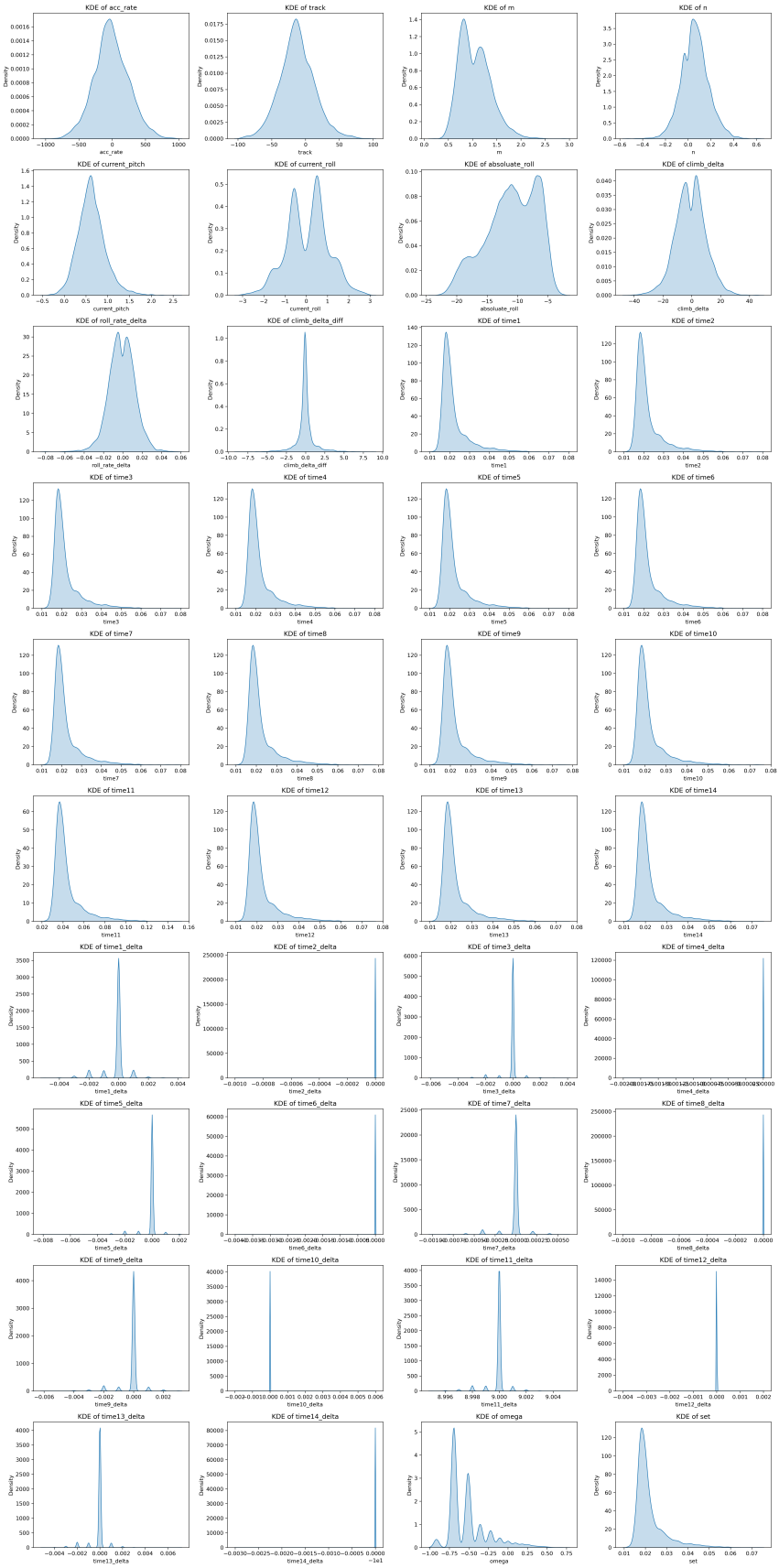

Figure 2.2: Kernel Density Estimate of Each Feature

Figure 2.2. Several insights emerged from these visualizations:

- Many features exhibit high skewness, particularly the `time` and `time_delta` series, where values are highly concentrated near zero with long tails.
- Some features, such as `m`, `n`, and `current_roll`, exhibit multi-modal distributions, indicating the presence of distinct operating regimes or system states.
- Motion-related features, such as `absoluate_roll` and `climb_delta`, show broader distributions with potential clustering.

2.3 Feature/Target Correlation

We computed the Pearson correlation coefficient between each feature and the target variable. The 3 most correlated features were shown in Table 2.3:

Table 2.3: Top 3 Correlated Features with Target

Feature	**Correlation**
absoluate_roll	0.704515
time1	0.641312
time5	0.640413

2.4 Feature/Target Correlation Scatter Plot

To better understand the relationships between the target variable and the top three most correlated features, we plotted scatter plots for each feature, shown in Figure 2.3.

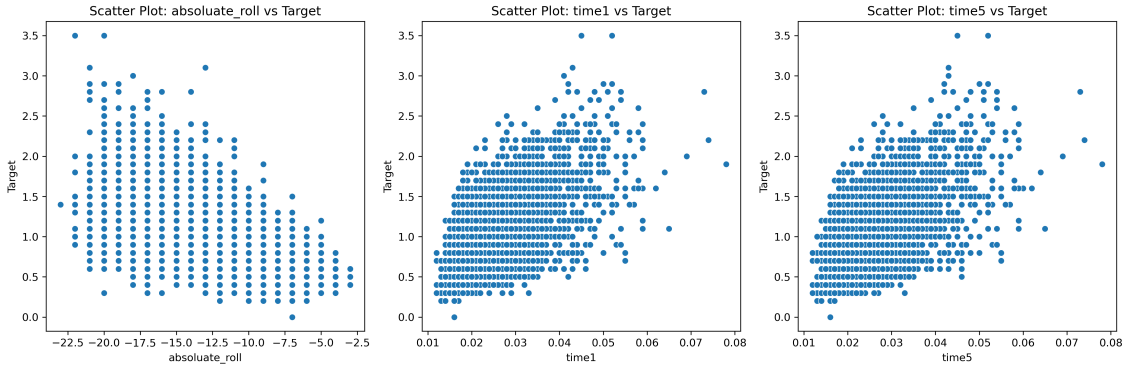

Figure 2.3: Scatter Plots for Top 3 Most Correlated Features with Target

The following insights were drawn from these scatter plots:

- A clear negative correlation is observed. As the `absoluate_roll` decreases, the target value tends to increase.
- Both the `time1` and `time5` characteristics show a positive correlation with the target. As these time-based features increase, the target tends to increase as well. This reflects potential time-dependent trends in the data.
- The similarity between `time1` and `time5` further supports the high correlation observed between time features, suggesting that some of these features may be redundant.

2.5 Feature/Feature Correlation

To better understand the relationships between features, we computed a pairwise correlation matrix, shown in Figure 2.4.

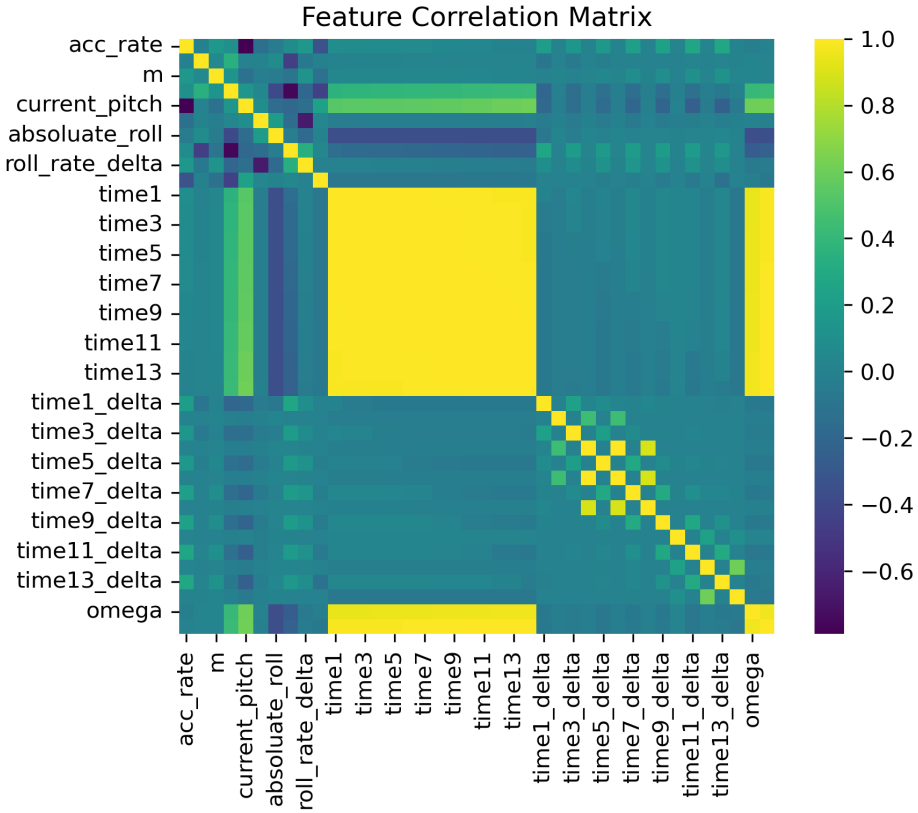

Figure 2.4: Feature Correlation Matrix

Several important observations can be made from this matrix:

- Time-based features (time1, time2, etc.) exhibit extremely high positive correlations with each other, indicating strong temporal dependencies within the dataset.
- Certain motion-related features, such as `omega`, also show positive correlations with specific time features, suggesting potential relationships between temporal progression and motion characteristics.
- Most other non-temporal features show low correlations with each other, indicating a relatively low degree of multicollinearity across these features.

2.6 Feature/Feature Correlation Scatter Plot

We selected `absoluate_roll` as the primary feature and identified the three features `n, omega, time1` most correlated with it. To better visualize the relationships between these features, we created a 4x4 pairplot matrix, which includes:

- **Diagonal elements**: Kernel Density Estimate (KDE) plots for the distribution of each feature.
- **Off-diagonal elements**: Scatter plots showing pairwise relationships between the features.

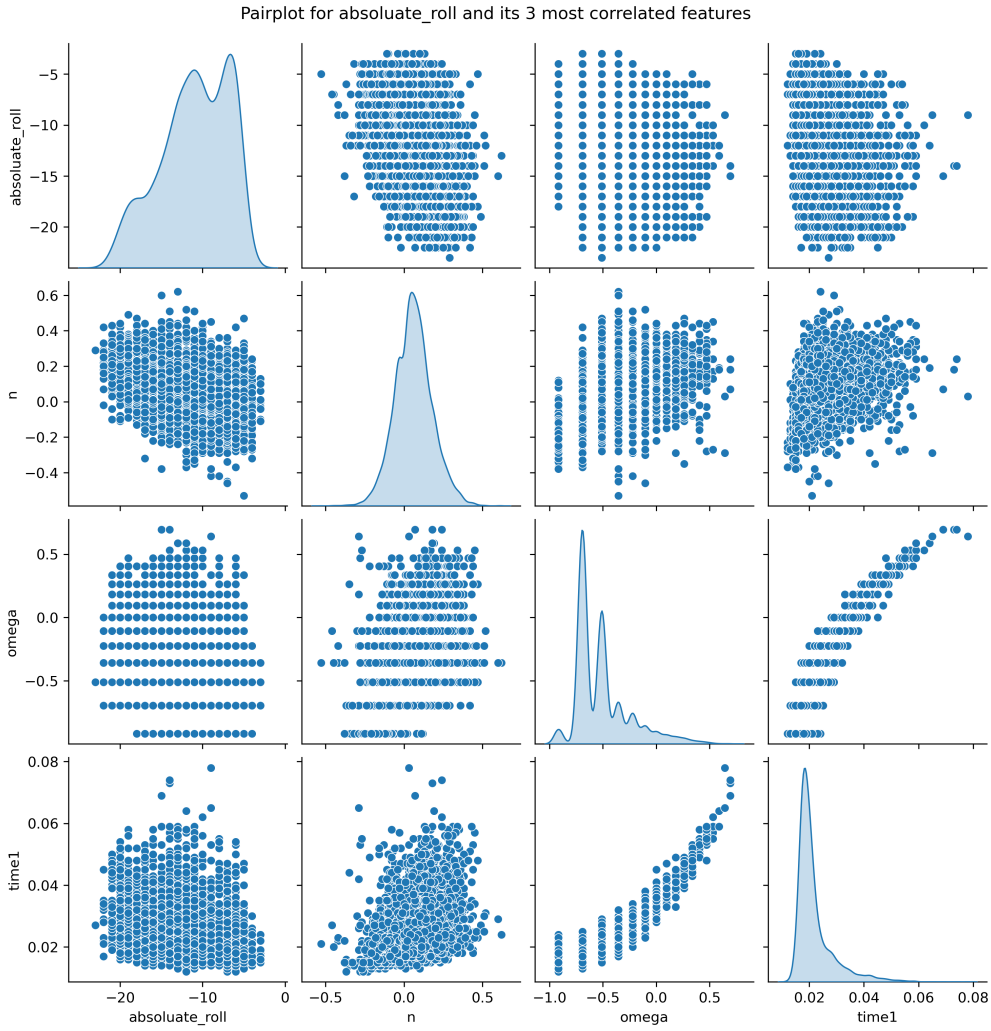

Figure 2.5: Pairplot for absoluate_roll and its 3 most correlated features

3 Data Preprocessing

3.1 Error and Missing Value Detection

To ensure the quality and reliability of the dataset, we conducted an initial analysis to detect potential errors and missing values. First, we examined the datasets size and structure, including the number of rows and columns, data types, and the presence of non-null values. This step provided a comprehensive overview of the dataset and helped identify any inconsistencies or anomalies. Through this step, I found that the data has 8250 rows and 41 columns, of which 40 columns are features and 1 column is target. And there is no non-null value in this dataset.

3.2 Outlier Removal Using Z-score

To improve the robustness of our dataset and mitigate the influence of extreme values, we employed the Z-score method to detect and remove outliers. The Z-score measures how many standard deviations a data point is away from the mean. It is calculated using the following formula:

$$Z = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the data point, μ is the mean of the dataset, and σ is the standard deviation. By computing the absolute Z-scores for each data point, we identified extreme values that significantly deviate from the majority of the data.

Typically, a threshold of $Z > 3$ is used to classify outliers. However, in our case, setting the threshold to 3 resulted in the removal of an excessive amount of data. Upon analyzing the data distribution, we observed that most features followed a roughly normal distribution with a concentrated spread. Given this observation, we opted for a more lenient threshold of $Z > 5$, which effectively eliminated extreme outliers while preserving the integrity of the dataset.

After applying this method, the dataset was reduced to 7,917 rows, ensuring that the remaining data maintains its representativeness while minimizing the impact of extreme values.

3.3 Splitting the Training and Test Sets

To ensure the model's ability to generalize to unseen data, we split the dataset into a training set and a testing set. The training set, which comprises 80% of the total data, is used to train the model, while the remaining 20% is reserved as the testing set to evaluate the models performance.

It is crucial to perform this data splitting before applying any form of standardization or normalization. If standardization is conducted prior to splitting, the computed mean and standard deviation would be influenced by the entire dataset, including the testing set. This introduces information leakage, where the model indirectly learns from the test set, leading to overly optimistic performance metrics.

3.4 Data Standardization/Normalization

3.4.1 Rescaling Data for Gradient Descent-Based Models

When using gradient descent-based optimization methods (e.g., linear regression, logistic regression, neural networks), rescaling data through standardization or normalization is essential. The key reasons are:

- **Equal Contribution of Features:** Features with larger numerical ranges can dominate gradient updates, leading to imbalance in learning. Rescaling ensures all features contribute equally to the optimization process.
- **Faster Convergence:** Standardizing data makes the loss function contours more spherical rather than elongated, improving the efficiency of gradient-based optimization and speeding up convergence.
- **Numerical Stability:** Large numerical disparities can lead to vanishing or exploding gradients, particularly in deep learning models. Normalization helps maintain numerical stability and smooth learning.

3.4.2 Rescaling Data for Regularization According to the Norm

Regularization techniques such as L1 (lasso) and L2 (ridge) impose constraints on the magnitude of model parameters. Rescaling data is crucial for ensuring fair and effective regularization:

- **Fair Regularization Effect:** Regularization penalizes larger coefficients more heavily. If features are on different scales, regularization may shrink some coefficients more aggressively, leading to biased learning.
- **Proper Hyperparameter Tuning:** When using regularization parameters (e.g., λ in ridge or lasso regression), unscaled features can make it difficult to choose appropriate values, as the penalty impact varies across features.
- **Improved Model Interpretability:** Standardized features ensure that the effect of regularization is uniform across all parameters, making the resulting coefficients easier to interpret.

3.4.3 Rescaling Data for Distance-Based Models

Models that rely on distance measures (e.g., Support Vector Machines, k-Nearest Neighbors, clustering algorithms like k-means) require rescaling for effective performance:

- **Consistent Distance Measurement:** These models compute distances (e.g., Euclidean distance), and if feature scales differ, features with larger magnitudes will disproportionately influence the calculations.
- **Better Decision Boundaries:** In SVM, the optimization process depends on distances. Without rescaling, decision boundaries may be skewed, leading to suboptimal classification.
- **Improved Clustering and Classification:** k-NN and clustering algorithms determine similarity based on distance. If features are not on the same scale, distance calculations will be biased towards high-magnitude features, leading to incorrect classifications or cluster formations.

Overall, rescaling data ensures that features contribute appropriately to learning, improving convergence, regularization effectiveness, and distance-based calculations.

3.4.4 Rescaling Techniques Applied in the Project

Data standardization is a crucial step, ensuring that different features have comparable scales and improving the performance of machine learning models. In this work, we applied three different standardization techniques based on the distribution characteristics of the features.

Standard Scaling For features that are approximately normally distributed (as shown in 3.1), we used the StandardScaler, which transforms data as follows:

$$X' = \frac{X - \mu}{\sigma} \tag{2}$$

where μ is the mean and σ is the standard deviation of the feature.

Robust Scaling For features with skewed distributions and outliers, we applied the RobustScaler (as shown in 3.1), which is based on the median and interquartile range (IQR):

$$X' = \frac{X - \text{median}(X)}{\text{IQR}(X)} \tag{3}$$

where $\text{IQR}(X) = Q_3 - Q_1$ (the difference between the 75th and 25th percentiles).

Min-Max Scaling For features with a highly concentrated distribution (e.g., needle-like distributions, as shown in 3.1), we used the MinMaxScaler to transform values into a specified range $[-1, 1]$:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \times (b - a) + a \tag{4}$$

where $a = -1$ and $b = 1$.

These approaches ensure that each feature is appropriately scaled based on its distribution, leading to better model performance.

Table 3.1: Data Normalization Methods and Their Associated Features

Scaler	**Features**
Standard Scaler	`acc_rate`, `track`, `n`, `current_pitch`, `current_roll`, `climb_delta`, `roll_rate_delta`, `climb_delta_diff`
Robust Scaler	`m`, `absolute_roll`, `time1`, `time2`, `time3`, `time4`, `time5`, `time6`, `time7`, `time8`, `time9`, `time10`, `time11`, `time12`, `time13`, `time14`, `omega`, `set`
MinMax Scaler	`time1_delta`, `time2_delta`, `time3_delta`, `time4_delta`, `time5_delta`, `time6_delta`, `time7_delta`, `time8_delta`, `time9_delta`, `time10_delta`, `time11_delta`, `time12_delta`, `time13_delta`, `time14_delta`

4 Baseline Models

4.1 K-fold Cross Validation and Grid Search

In machine learning, model selection and hyperparameter tuning play crucial roles in achieving optimal performance. Two widely used techniques for this purpose are **K-fold Cross Validation** and **Grid Search**. These methods help assess model generalization and identify the best hyperparameters, reducing the risk of overfitting and underfitting.

4.1.1 K-fold Cross Validation

K-fold Cross Validation (K-fold CV) is a resampling technique that splits the dataset into K equal-sized subsets or folds. The model is trained K times, each time using $K - 1$ folds for training and the remaining fold for validation. The process repeats until each fold has served as the validation set exactly once. The final performance is determined by averaging the evaluation metrics across all K iterations. This method mitigates bias and variance issues present in single train-test splits.

Formally, given a dataset D with N samples, the steps of K-fold Cross Validation are:

1. Shuffle the dataset D randomly.
2. Split D into K equal-sized subsets: $D_1, D_2, ..., D_K$.
3. For each fold k, train the model on $D_{train} = D \setminus D_k$ and validate on D_k.
4. Compute performance metrics for each fold and average them to obtain the final model evaluation.

The choice of K is problem-dependent, with common values being $K = 5$ or $K = 10$. A higher K leads to a lower bias but a higher computational cost.

4.1.2 Grid Search

Grid Search is a brute-force hyperparameter optimization technique that systematically evaluates a predefined set of hyperparameter combinations. It involves:

1. Defining a set of hyperparameter values for tuning, such as learning rates, regularization parameters, or kernel functions.
2. Exhaustively searching through all possible combinations of the given hyperparameters.
3. Evaluating each combination using K-fold Cross Validation to determine the best-performing set.

Let $\Theta = \{\theta_1, \theta_2, ..., \theta_m\}$ be the set of hyperparameters and $\mathcal{M}(\theta)$ be the machine learning model parameterized by θ. The optimal hyperparameters θ^* are found as:

$$\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{K} \sum_{k=1}^{K} L(\mathcal{M}(\theta), D_k) \tag{5}$$

where $L(\cdot)$ represents a loss function measuring model performance.

By combining K-fold Cross Validation with Grid Search, we ensure robust model evaluation while systematically finding the optimal hyperparameters, leading to improved generalization and performance.

4.2 Ridge Regression

Ridge Regression is a linear regression model that incorporates L2 regularization to prevent overfitting by penalizing large coefficients. The model's formulation is given by:

$$\hat{y} = Xw + \epsilon \tag{6}$$

where w is the weight vector, and the objective function with L2 regularization is:

$$\min_{w} \sum_{i=1}^{n} (y_i - X_i w)^2 + \alpha \sum_{j=1}^{p} w_j^2 \tag{7}$$

where α is the regularization strength that controls the trade-off between bias and variance.

Since the Ridge Regression model has only one hyperparameter (α), we performed a GridSearchCV with a 5-fold cross-validation to select the best α from:

$$\alpha \in \{0.1, 1.0, 10.0\} \tag{8}$$

This ensures that the model generalizes well to unseen data while preventing overfitting.

4.3 Random Forest Regressor

Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting. Each tree is trained on a bootstrap sample of the dataset, and the final output is computed as the average prediction across all trees.

The prediction for a given input X is:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(X) \tag{9}$$

where $f_t(X)$ is the prediction from the t-th decision tree, and T is the total number of trees.

To optimize the performance of Random Forest, we performed a GridSearchCV with a 5-fold cross-validation over the following hyperparameters:

- Number of trees (`n_estimators`): {50, 100, 150, 200}
- Maximum tree depth (`max_depth`): {None, 10, 20, 30}

The final model was selected based on the lowest MSE.

4.4 Neural Network Regressor

Neural networks are powerful function approximators that can capture complex relationships in data. We employed a multi-layer perceptron (MLP) regressor, which consists of multiple fully connected layers:

$$\hat{y} = f(W_L \cdot f(W_{L-1} \cdots f(W_1 X + b_1) + b_{L-1}) + b_L) \tag{10}$$

where:

- W_i are the weight matrices at layer i,
- b_i are the bias terms at layer i,
- $f(\cdot)$ is the activation function, which we selected as either ReLU or Tanh.

We tuned the following hyperparameters using GridSearchCV with a 5-fold cross-validation:

- Hidden layer sizes: {(50,), (50,50), (50,50,50), (50,50,50,50)}
- Activation functions: {ReLU, Tanh}
- Solvers: {Adam, LBFGS}

The best-performing network was chosen based on the lowest MSE.

4.5 Performance Analysis

The performance of the models was evaluated using:

- Mean Squared Error (MSE): Lower values indicate better predictive accuracy.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{11}$$

- R-Squared (R^2): Measures the proportion of variance explained by the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{12}$$

- Training Time: Important for assessing computational efficiency.

$$T_{\text{train}} = T_{\text{end}} - T_{\text{start}} \tag{13}$$

The results are summarized in Table 4.1 and visualized in Figure 4.1.

Model	**MSE**	R^2	**Training Time**
Ridge Regression	0.0279	0.8157	7.3s
Random Forest	0.0282	0.8141	2m 48.5s
Neural Network	0.0266	0.8243	22.8s

Table 4.1: Comparison of Model Performance: Mean Squared Error (MSE), R^2, and Training Time.

From the results, we observe:

- The Neural Network achieves the lowest MSE (0.0266) and the highest R^2 (0.8243), suggesting the best overall predictive performance.
- Ridge Regression performs comparably, with an MSE of 0.0279 and R^2 of 0.8157, but with a significantly lower training time (7.3s), making it a strong candidate for fast inference.

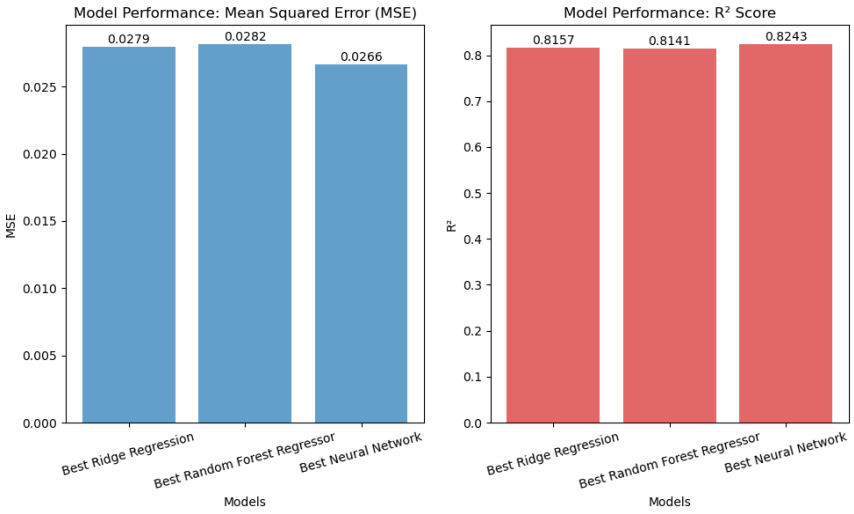

Figure 4.1: Comparison of Model Performance: Mean Squared Error (MSE) and R-Squared (R^2).

- Random Forest performs similarly to Ridge Regression, with an MSE of 0.0282 and R^2 of 0.8141, but has the longest training time (2m 48.5s), making it computationally expensive.

While the Neural Network achieves the best predictive performance, its training time (22.8s) is significantly longer than Ridge Regression but much shorter than Random Forest. Ridge Regression emerges as an efficient alternative for scenarios where interpretability and low computational cost are prioritized.

4.6 Paired t-Test and p-Value Calculation

The paired t-test is a statistical test used to compare the means of two related groups. It evaluates whether the mean difference between paired observations is statistically significant. The paired t-test statistic is given by:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{14}$$

where:

- $\bar{d}$ is the mean of the differences between paired samples.
- s_d is the standard deviation of the differences.
- n is the number of paired observations.

The p-value is the probability of obtaining the observed test statistic under the null hypothesis. It is computed using the cumulative distribution function (CDF) of the t-distribution:

$$p = 2 \times P(T \geq |t|) = 2 \times (1 - F_{t,n-1}(|t|)) \tag{15}$$

where:

- T follows a t-distribution with $n-1$ degrees of freedom.
- $F_{t,n-1}(|t|)$ is the cumulative distribution function (CDF) of the t-distribution.

A smaller p-value (typically $p < 0.05$) indicates strong evidence against the null hypothesis, suggesting that the two models have significantly different error distributions.

Based on the theorem above, we applied 5-fold cross-validation to compute the in-sample and out-of-sample errors and performed paired t-tests comparing the in-sample and out-of-sample errors of Ridge Regression, Random Forest, and MLP models.

4.6.1 Paired t-Test (In-Sample Error)

The paired t-test results for in-sample errors are summarized as follows:

- Ridge vs. Random Forest: $t = 8.33$, $p = 0.0011$
- Ridge vs. MLP: $t = 19.08$, $p = 4.45 \times 10^{-5}$
- Random Forest vs. MLP: $t = 0.12$, $p = 0.91$

Since the p-values for Ridge vs. Random Forest and Ridge vs. MLP are both below 0.05, we reject the null hypothesis, indicating that there is a significant difference in in-sample error between these models. However, the p-value for Random Forest vs. MLP is 0.91, suggesting no statistically significant difference between these two models.

4.6.2 Paired t-Test (Out-of-Sample Error)

The paired t-test results for out-of-sample errors are:

- Ridge vs. Random Forest: $t = -1.00$, $p = 0.373$
- Ridge vs. MLP: $t = -1.65$, $p = 0.175$
- Random Forest vs. MLP: $t = -0.94$, $p = 0.399$

All p-values are greater than 0.05, meaning we fail to reject the null hypothesis. This suggests that there is no statistically significant difference in out-of-sample performance among the models.

4.7 Kernel Density Estimation

Kernel Density Estimation (KDE) is a non-parametric method for estimating the probability density function (PDF) of a dataset. Given a set of observations $x_1, x_2, ..., x_n$, KDE is computed as:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{16}$$

where:

- $K(\cdot)$ is the kernel function (typically a Gaussian function).
- h is the bandwidth parameter, controlling the smoothness of the estimate.
- x_i are the data points.

The KDE helps visualize the distribution of in-sample error and out-of-sample error for each model, allowing for comparison of their performance.

Figure 4.2 presents the kernel density estimation of in-sample errors for Ridge Regression, Random Forest, and MLP models. The distributions of Random Forest and MLP nearly overlap, suggesting that both models achieve very low training errors and exhibit similar error distributions. In contrast, Ridge Regression has a noticeably higher and more dispersed error distribution, indicating greater variability in its in-sample performance. This difference suggests that Random Forest and MLP may have overfitted the training data, capturing fine-grained patterns that may not generalize well to unseen data. The extremely low in-sample errors for these models indicate a high capacity to fit the training data but also raise concerns about potential overfitting.

Figure 4.3 shows the kernel density estimation of out-of-sample errors, revealing key differences in model generalization. Ridge Regression exhibits a sharper and more concentrated peak, suggesting lower variance in its predictions and more stable generalization to unseen data. In contrast, the error distributions of Random Forest and MLP are broader and more dispersed, indicating higher variability in their predictions on the test set. This suggests that while these models performed exceptionally well on the training data, their performance on unseen data is less consistent, further supporting the possibility of overfitting. To mitigate this, techniques such as regularization, early stopping for MLP, or limiting the depth of Random Forest trees could help improve generalization.

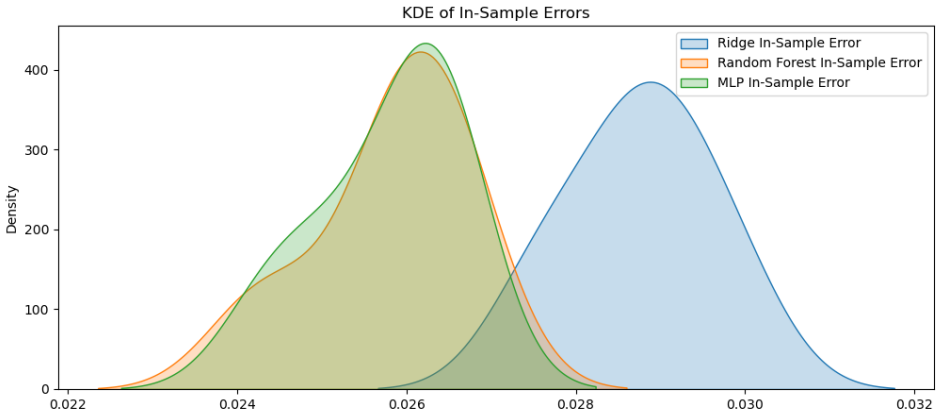

Figure 4.2: KDE of In-Sample Errors for Ridge Regression, Random Forest, and MLP.

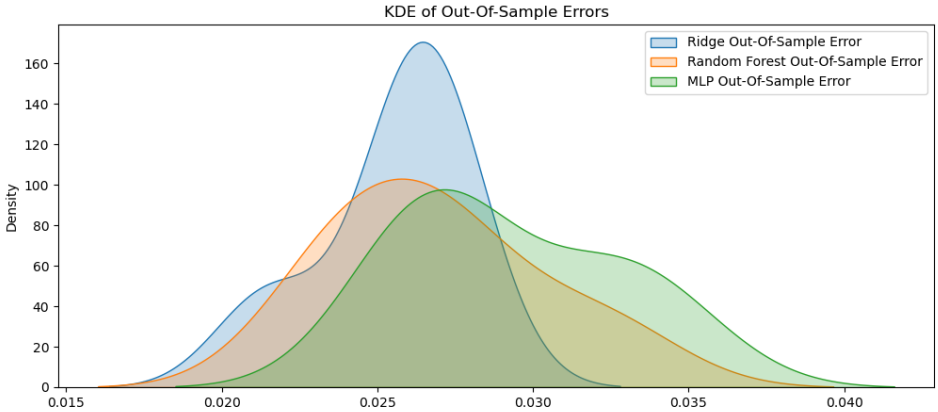

Figure 4.3: KDE of Out-of-Sample Errors for Ridge Regression, Random Forest, and MLP.

5 K-Means Clustering

5.1 K-Means with k=2 and Information Gain Analysis

Data outliers exceeding 5 standard deviations from the mean were handled according to the 5 criterion.Normalization was performed using the data normalization methods mentioned earlier (applying methods for different feature distributions). K-means clustering with k=2 was then performed on the processed and normalized data, and a scatter plot illustrating the distance to each cluster centroid (Figure 5.1) was generated.

Next, the distance-to-centroid values were introduced as an additional feature in the original dataset to construct a linear regression model, with the original dataset (without the new feature) serving as the baseline. As shown in Table 5.1, the correlation coefficient for the linear regression model increased after incorporating the cluster distance feature, suggesting that the clustering information provides an information gain. However, the improvement was modest, likely due to the simplicity of the linear regression model.

5.2 Clustering Analysis and Feature Incorporation

To determine the optimal hyperparameter k for the K-means algorithm, we can employ the elbow method. Figure 5.2 plots inertia as a function of k, alongside the corresponding Calinski–Harabasz Score and silhouette coefficient. From this figure, the elbow (or turning point) occurs roughly in the range $2 \leq k \leq 4$. If k is

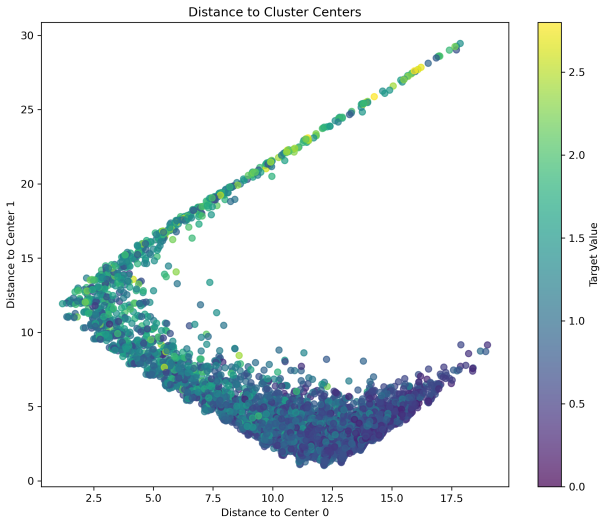

Figure 5.1: Scatter plot illustrating the distance to each cluster centroid.

Table 5.1: Comparison of Baseline and Clustering-Enhanced Models

Method	**R^2**	**MSE**	**R^2 Gain**
Baseline	0.8163	0.0281	–
With Clustering Features	0.8177	0.079	0.0014

chosen below this range, the models predictive capability declines substantially; if k is increased, there is a modest improvement in predictive performance, but at the cost of greater model complexity. Within the [2, 4] interval, both the Calinski–Harabasz Score and the silhouette coefficient reach their maximum at $k = 2$. Therefore, considering these metrics collectively, $k = 2$ is concluded to be the optimal number of clusters for this dataset when applying K-means.

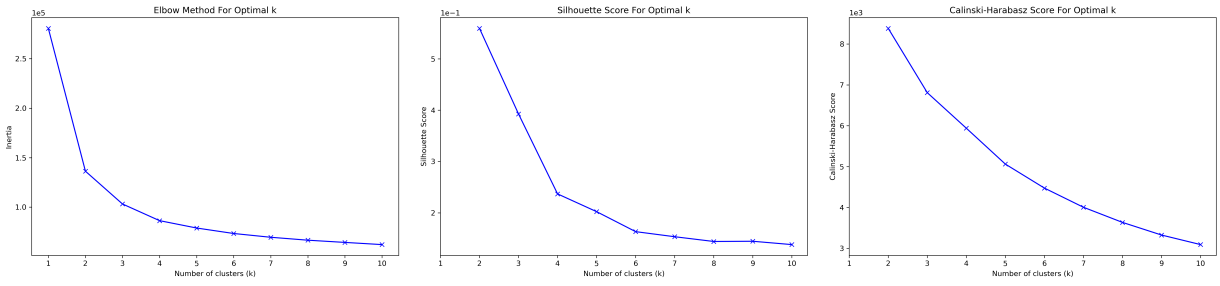

Figure 5.2: Elbow Method for Determining the Optimal Number of Clusters: Inertia, CalinskiHarabasz Score, and Silhouette Coefficient.

The observed improvement in regression coefficients after adding the cluster distance feature implies that the original data may contain latent group structures not explicitly captured by the baseline model. These clusters, identified by K-means, exhibit systematic relationships with the target variable. A simple linear regression model might not adequately capture more complex featuretarget relationships, whereas the distance-based feature encodes distributional patterns in cluster space and may therefore represent higher-order interactions or nonlinear effects. For subsequent work, incorporating cluster-distance features may

enhance overall model accuracy and performance. It may also be beneficial to explore alternative clustering methods to identify more suitable cluster-based features.

6 Conclusion

6.1 Summary of Work

In this milestone, we conducted a comprehensive analysis encompassing data exploration, data preprocessing, baseline modeling, and clustering. We began by performing extensive exploratory data analysis to understand the dataset's characteristics, distribution patterns, and relationships between variables. Subsequently, we implemented a lot of preprocessing techniques, including outlier removal, normalization, dataset splitting, and so on, to enhance data quality and prepare it for modeling. Three baseline modelsRidge Regression, Random Forest Regressor, and Neural Network Regressorwere developed and evaluated for predictive performance. Additionally, clustering analysis using the K-means algorithm was conducted to uncover latent structures and assess their impact on prediction accuracy.

6.2 Key Findings and Results

The key insights derived from this milestone include:

1. **Data Characteristics**: The target variable exhibits a right-skewed distribution, while time-related features show substantial redundancy. These observations underscore the importance of carefully addressing multicollinearity among temporal attributes.

2. **Model Performance Benchmark**: The results of baseline models part indicate that the Neural Network model achieved the best predictive performance, with the lowest MSE (0.0266) and highest R^2 (0.8243). However, it required a significantly longer training time (22.8s) compared to Ridge Regression (7.3s). Ridge Regression exhibited slightly lower predictive performance but trained much faster, making it a practical choice for scenarios requiring efficient computation. Random Forest performed similarly to Ridge Regression in terms of MSE and R^2, but it had the longest training time (2m 48.5s), which may limit its applicability in time-sensitive tasks. Thus, idge Regression balanced efficiency and interpretability better.

3. **T-test and Kernel Density Estimation**: Paired t-tests showed that Ridge Regression had significantly different training errors compared to Random Forest and Neural Network, while the latter two had no significant difference. However, out-of-sample errors showed no statistically significant differences among all models. KDE analysis further revealed that Random Forest and Neural Network had lower training errors but more dispersed test errors, suggesting potential overfitting, whereas Ridge Regression had a more stable error distribution, indicating more balanced generalization ability.

4. **Clustering Feature Gains**: Incorporating K-means clustering features yielded a slight improvement for linear models (R^2 gain of 0.0014), indicating the existence of latent substructures in the data. However, this enhancement was modest, partly due to K-means sensitivity to initial centroids and the inherent difficulty of linear models in capturing nonlinear relationships.

6.3 Limitations

Nonetheless, several limitations persist at this stage of the work:

1. Outlier handling relies on a fixed threshold (Z-score = 5), potentially introducing subjective bias.

2. Coupling between the linear model and clustering features remains modest, leaving potential nonlinear interactions underexplored.

3. The baseline models include Ridge Regression, Random Forest, and Neural Network, but more complex architectures were not explored.

4. The hyperparameter tuning was conducted over a limited search space, which may not have identified the most optimal model settings.

5. Employing K-means as the sole clustering method makes results susceptible to initialization effects.

6.4 Future Work and Improvements

Looking ahead to subsequent milestones, our efforts will extend in the following directions:

1. **Data Processing and Feature Engineering**: Integrate more flexible outlier detection methods (e.g., Isolation Forest) and automated feature selection or construction to enhance data quality and feature representation.

2. **Exploration of More Complex Models**: Experiment with ensemble learning techniques such as XGBoost or deeper neural network architectures to better capture nonlinear patterns and potential interactions.

3. **Optimization of Clustering Strategies**: Introduce additional clustering algorithms (e.g., DBSCAN, spectral clustering) or compare multiple clustering approaches to obtain more robust and discriminative cluster features.

6.5 Milestone 1 Work Distribution

The responsibilities within our group were distributed as follows:

- **Hongyi Zhang**: Data Preprocessing, Baseline Modeling, Report Writing.
- **Ziang Deng**: Clustering Analysis, Report Writing.
- **Nancy Wang**: Data Exploration, Report Writing.