# Title :

## Feature Finder

# Subtitle:

Feature Finder is a cutting-edge machine learning tool engineered to pinpoint and prioritize key features from extensive datasets. Utilizing sophisticated algorithms, it refines the feature selection process, boosting both the efficiency and clarity of predictive models.

# Enroll. No :

**202418011 , 202418009 , 202418042 , 202418056**

**Date :** **26/08/2004**

# Introduction

## Objective:

Feature Finder is a machine learning tool designed to optimize the estimation of maintenance costs. By leveraging advanced algorithms to identify and prioritize crucial features from extensive datasets, it enhances the accuracy of cost predictions.

# Project Overview

## Dataset:

**Total Entries:** 10,000

**Columns:**

1. **Car Make** – The manufacturer of the car (object)
2. **Car Model** – The specific model of the car (object)
3. **Year of Manufacture** – The year the car was made (int64)
4. **Engine Size** – The size of the car's engine (int64)
5. **Mileage** – The total distance the car has traveled (int64)
6. **Maintenance History** – Record of past maintenance activities (object)
7. **Maintenance Cost** – The cost incurred for maintenance (int64)
8. **Repair Costs** – Expenses related to repairs (int64)
9. **Insurance Premium** – Cost of insurance for the car (int64)
10. **Coverage Type** – Type of insurance coverage (object)
11. **Location** – Geographical location of the vehicle (object)
12. **Driver Age** – Age of the car's driver (int64)
13. **Driver Experience** – Years of experience of the driver (int64)
14. **Safety Feature** – Features related to vehicle safety (object)
15. **Vehicle Condition** – Current condition of the vehicle (object)

**Data Types**:

- **Numeric (int64)**: 8 attributes
- **Categorical (object)**: 7 attributes

## Model:

Feature Finder utilizes three powerful regression models to estimate maintenance costs:

1. **Random Forest Regressor:** An ensemble method that combines multiple decision trees to improve accuracy and handle complex data patterns.

2. **Ridge Regression:** A technique that applies L2 regularization to address multicollinearity and enhance model robustness.

3. **Lasso Regression:** A method incorporating L1 regularization to perform feature selection and promote sparse, interpretable models.

## Metrics :

- MSE provides an idea of the average squared error, useful for understanding the variance of errors.
- $R^2$ reflects how well the model explains the variability of the outcome.
- MAE offers a simple interpretation of average prediction error.
- RMSE quantifies the average prediction error by taking the square root of the mean squared differences between predicted and actual values.

## Data Preparation

**Preprocessing Steps:**
Outline key preprocessing tasks with brief descriptions.

1. **Cleaning:** Removed missing values.
2. **Normalization:** Scaled features to a range of [0, 1].
3. **Data Transformation**: Raw data is converted into a format suitable for analysis or modeling.
4. **Standardization :** Mean Centering

# Model Training

## ❖ Linear Regression Model

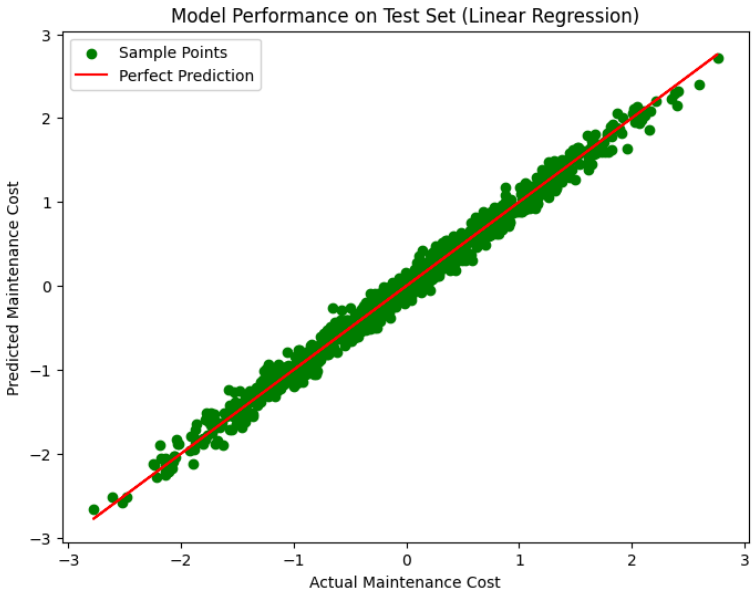
### Data Splitting:

- Divided data into training, validation, and test sets using train_test_split.

### Model Training:.

- Trained the Linear regression model using the best parameter
- Project(202418011).ipynb

### Model Evaluation :




**Validation Set Performance:**

- MSE: 0.0653
- $R^2$: 0.936
- MAE: 0.198
- RMSE: 0.255

**Test Set Performance:**

- MSE: 0.0739
- $R^2$: 0.926
- MAE: 0.215
- RMSE: 0.271

# Pseudocode

1. Import necessary libraries
   - pandas
   - numpy
   - scikit-learn modules (LinearRegression, train_test_split, PolynomialFeatures, StandardScaler, LabelEncoder, mean_squared_error, r2_score, mean_absolute_error)
   - matplotlib for plotting

2. Load dataset from Excel file
   - df = read dataset from file path

3. Define features (X) and target variable (y)
   - X = drop "Maintenance Cost" column from df
   - y = "Maintenance Cost" column from df

4. Convert categorical features to numerical using LabelEncoder
   - Initialize LabelEncoder
   - Identify categorical columns in X
   - Apply LabelEncoder to categorical columns in X

5. Scale features and target variable
   - Initialize StandardScaler
   - Scale features X using StandardScaler
   - Scale target variable y using StandardScaler

6. Create a pipeline with polynomial features and linear regression model
   - Initialize PolynomialFeatures with degree=2
   - Initialize LinearRegression
   - Create a Pipeline with steps:
     - Add polynomial features transformation
     - Add Linear Regression model

7. Split the data into training, validation, and test sets
   - Split data into training set (80%) and temporary set (20%)
   - Split temporary set into validation set (50%) and test set (50%)

8. Train the model using the training set
   - Fit the pipeline with training data

9. Make predictions on the validation and test sets
   - Predict using the validation set
   - Predict using the test set

10. Evaluate model performance on the validation set
   - Calculate and print Mean Squared Error
   - Calculate and print R^2 Score
   - Calculate and print Mean Absolute Error
   - Calculate and print Root Mean Squared Error

11. Evaluate model performance on the test set
   - Calculate and print Mean Squared Error
   - Calculate and print R^2 Score
   - Calculate and print Mean Absolute Error
   - Calculate and print Root Mean Squared Error

12. Plot actual vs. predicted values for the test set
   - Create scatter plot of actual vs. predicted values
   - Plot a line for perfect prediction
   - Add labels and title to the plot
   - Show the plot

13. Make a prediction for a new car
   - Define new car data in DataFrame format
   - Convert categorical features to numerical using LabelEncoder
   - Scale new car data using StandardScaler
   - Predict maintenance cost using the trained pipeline
   - Print the predicted maintenance cost

## ❖ Ridge Model

### Data Splitting:

- Divided data into training, validation, and test sets using train_test_split.

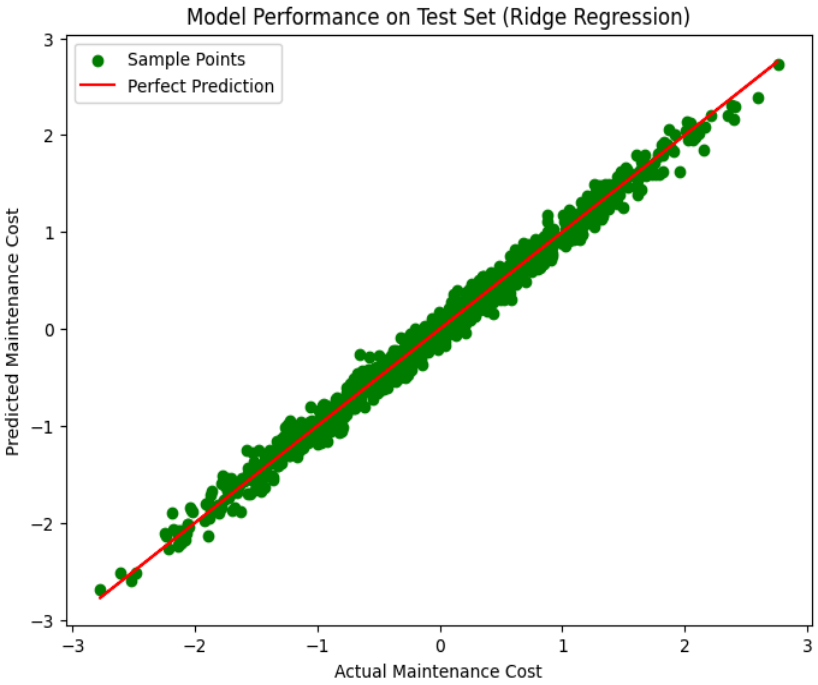
### Feature Selection:

- Employed RFE with Ridge regression to select important features.

### Model Training:

- Used GridSearchCV with Ridge regression to find the best hyperparameters.
- Trained the Ridge regression model using the best parameters.
- ∞ Project(202418011).ipynb

### Model Evaluation :




**Validation Set Performance:**

- MSE: 0.0116
- $R^2$: 0.989
- MAE: 0.0875
- RMSE: 0.1079

**Test Set Performance:**

- MSE: 0.0114
- $R^2$: 0.989
- MAE: 0.0864
- RMSE: 0.1070

# Pseudocode

1. Import Required Libraries
   - Load libraries needed for data handling (`pandas`, `numpy`), machine learning (`sklearn`), and visualization (`matplotlib`).

2. Load the Dataset
   - Read the data from an Excel file into a DataFrame.

3. Prepare the Data
   - Separate the dataset into features (inputs) and the target variable (maintenance cost).
   - Convert any categorical features into numeric values.

4. Standardize the Data
   - Scale the features and target variable so they have a standard format.

5. Create Polynomial Features
   - Generate new features based on polynomial combinations of existing features to capture more complex relationships.

6. Split the Data
   - Divide the data into training, validation, and test sets to evaluate the model.

7. Perform Feature Selection
   - Use a technique to select the most important features for the model, reducing the number of features to a manageable number.

8. Define and Search for Best Model Parameters
   - Set up a range of parameters to test for Ridge Regression.
   - Use a search method to find the best parameters for the model based on cross-validation.

9. Train the Model
   - Train the Ridge Regression model using the best parameters found from the search.

10. Make Predictions
    - Predict maintenance costs using the trained model on both the validation and test data sets.

11. Evaluate Model Performance
    - Print out performance metrics (like error rates and accuracy) for both the validation and test sets to understand how well the model is performing.

12. Visualize Predictions
    - Create a plot comparing the actual maintenance costs to the predicted values for the test set to visualize model accuracy.

13. Predict for New Data
    - Prepare a new data entry for a specific car, convert its categorical features, and scale the data.
    - Use the trained model to predict the maintenance cost for this new car and print the result.

## ❖ Lasso Model

### Data Splitting:

- Divided data into training, validation, and test sets using train_test_split.

### Model Training:

- Used GridSearchCV with Ridge regression to find the best hyperparameters.
- Trained the Lasso regression model using the best parameters.
- Project(202418011).ipynb

### Model Evaluation :


### Validation Set Performance:

- MSE: 0.0119
- $R^2$: 0.988
- MAE: 0.0882
- RMSE: 0.1093

### Test Set Performance:

- MSE: 0.0118
- $R^2$: 0.989
- MAE: 0.0879
- RMSE: 0.1086

# Pseudocode

1. Import Necessary Libraries
   - Load libraries for data handling (`pandas`, `numpy`), machine learning (`sklearn`), and visualization (`matplotlib`).

2. Load the Data
   - Read the dataset from an Excel file into a DataFrame.

3. Prepare the Data
   - Extract the features (X) and the target variable (y) from the DataFrame.
   - Convert categorical columns in the features into numeric values using a label encoder.

4. Standardize the Data
   - Scale the feature values to have a mean of 0 and a standard deviation of 1.
   - Scale the target variable in the same way.

5. Create Polynomial Features
   - Generate new features that are combinations of the original features to capture more complex patterns.

6. Split the Data
   - Divide the data into training, validation, and test sets for model evaluation.

7. Set Up and Run Grid Search
   - Define a range of possible values for the Lasso regression parameter `alpha`.
   - Use Grid Search to find the best `alpha` value by evaluating different options.

8. Train the Model
   - Train a Lasso regression model using the best `alpha` value found from the Grid Search.

9. Make Predictions
   - Use the trained model to make predictions on both the validation and test sets.

10. Evaluate Model Performance
    - Print out performance metrics (such as Mean Squared Error, R^2 Score, Mean Absolute Error, and Root Mean Squared Error) for both the validation and test sets.

11. Visualize Results
    - Create a scatter plot comparing the actual maintenance costs to the predicted values from the test set.
    - Add a reference line showing perfect predictions for better visualization.