**PROJECT REPORT**

# Business Intelligence and data Visualization

**CLL 232**

A logo with text and a curved orange and black design

Description automatically generated with medium confidence

Faculty name: Mrs. Poonam Chaudhary Student name: shivam

vibhav

Roll No.: 21csu325

21csu347

Section: DS-IV-B

**Department of Computer Science and Engineering**

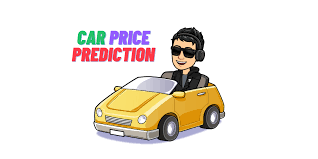
**NorthCap University, Gurugram- 122001, India**

**Session 2023-24**

| **CONTENTS** | | | | |
| --- | --- | --- | --- | --- |
| S.N. | **Details** | Page No. | | |
| 1 | **Introduction** | 3 | | |
| 2 | **About The Project** | 4 | | |
| 3 | **Model Experiment** | 6 | |
| 4 | **Reasons for Using Multiple Models** | 10 | |
| 5 | **Dataset Overview** | | 11 | |
| 6 | **Screenshots** | | 12 | |
| 7 | **References** | 13 | |

Project Title:

Car Price Prediction



The goal of a car price prediction project is to develop an accurate model that estimates the market value of a car based on its features. This model aims to assist buyers and sellers by providing informed pricing decisions, offering insights into the factors influencing car prices, and contributing to a better understanding of market dynamics. The project involves creating a user-friendly interface for easy accessibility, ensuring continuous model improvement to adapt to changing conditions, and addressing ethical considerations related to fairness and transparency. Ultimately, the project aims to enhance decision-making in the automotive market and potentially find practical applications in business strategies for entities like car dealerships.

About The Project:

Car Price Prediction

Problem Statement: This project aims to tackle the challenge of accurately predicting car prices by addressing issues such as price variability due to diverse features, ensuring high-quality and unbiased data representation, optimizing model accuracy through suitable algorithms, creating a user-friendly interface for accessibility, adapting the model to dynamic market conditions, addressing ethical considerations, and exploring potential business applications for the predictive model.A white car with yellow text

Description automatically generated

Approach:The approach to solving the car price prediction problem involves a systematic series of key steps:

1. Data Collection:Gather a comprehensive dataset encompassing various car features, conditions, and pricing information.
2. Data Preprocessing:Clean and preprocess the dataset, handling missing data, encoding categorical variables, and ensuring data quality.

3. Feature Engineering:Create new features and extract meaningful information to enhance the model's understanding of car pricing factors.

4. Model Selection:Choose an appropriate regression model based on the dataset characteristics. Options include linear regression, decision tree regression, randomforest regression, or gradient boosting regression.

5. Training-Testing Split:Split the dataset into training and testing sets to train the model on one subset and evaluate its performance on another.

6. Model Training:Train the selected model using the training dataset, adjusting hyperparameters for optimal performance.

7. Model Evaluation:Evaluate the model's performance using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), or R-squared on the testing set.

8. User-Friendly Interface:Develop an intuitive user interface for easy input of car details and retrieval of price predictions.

9. Continuous Improvement:Establish a framework for continuous model improvement, considering updates based on changing market conditions and user feedback.

10. Ethical Considerations:Implement measures to address ethical considerations, ensuring fairness, transparency, and responsible use of the predictive model.

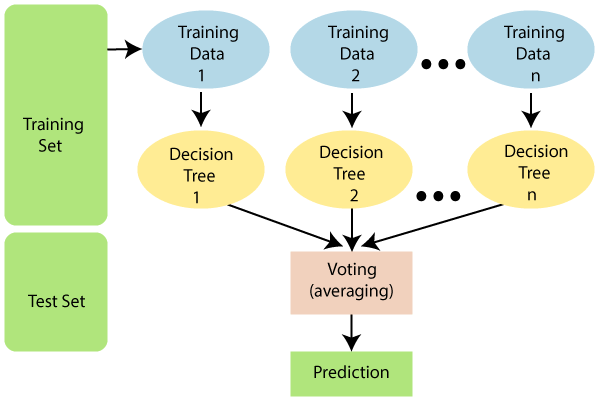
11. Educational Value:Integrate educational components to enhance user understanding of the factors influencing car prices.

Machine learning model

Here's an explanation of each model and the potential reasons for using them in the context of hate speech recognition:

**Random Forest:**

**Type:** Ensemble Learning (specifically, Bagging)  
**Use Case:** Random Forest is often used for classification and regression tasks.  
**Working Principle:** Random Forest builds multiple decision trees during training and merges them together to get a more accurate and stable prediction. It introduces randomness in the tree-building process by considering random subsets of features and data points.  
**Key Features:**Robust and less prone to overfitting.  
Can handle large datasets with high dimensionality.  
Provides feature importance.



**Polynomial Regression:**

**Type:** Regression, Polynomial

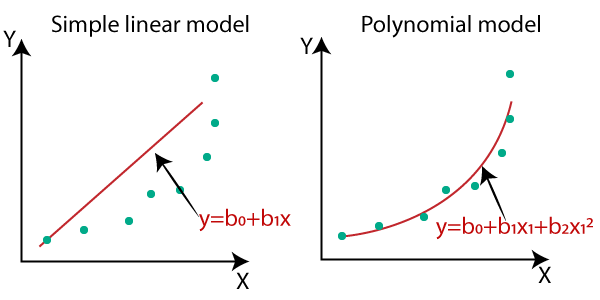
**Use Case:** Applied when the relationship between the independent and dependent variables is better represented by a polynomial function rather than a linear one. Useful for capturing non-linear patterns in data.

**Working Principle:** Polynomial regression models the relationship between the independent variable(s) and the dependent variable as an nth degree polynomial. This allows for a more flexible fit to the data compared to linear regression.

**Key Features:**

**Degree Selection:** The degree of the polynomial determines the complexity of the model. Balancing higher degrees for better fit with the risk of overfitting is crucial.

**Overfitting and Underfitting:** Prone to overfitting, especially with higher-degree polynomials. Regularization techniques or cross-validation can be employed to mitigate overfitting.



**Linear Regression:**

**Type:** Regression, Linear

**Use Case:** Primarily used for predicting the relationship between a dependent variable and one or more independent variables when the relationship is assumed to be linear. Widely applied in various fields for tasks such as price prediction, sales forecasting, and trend analysis.

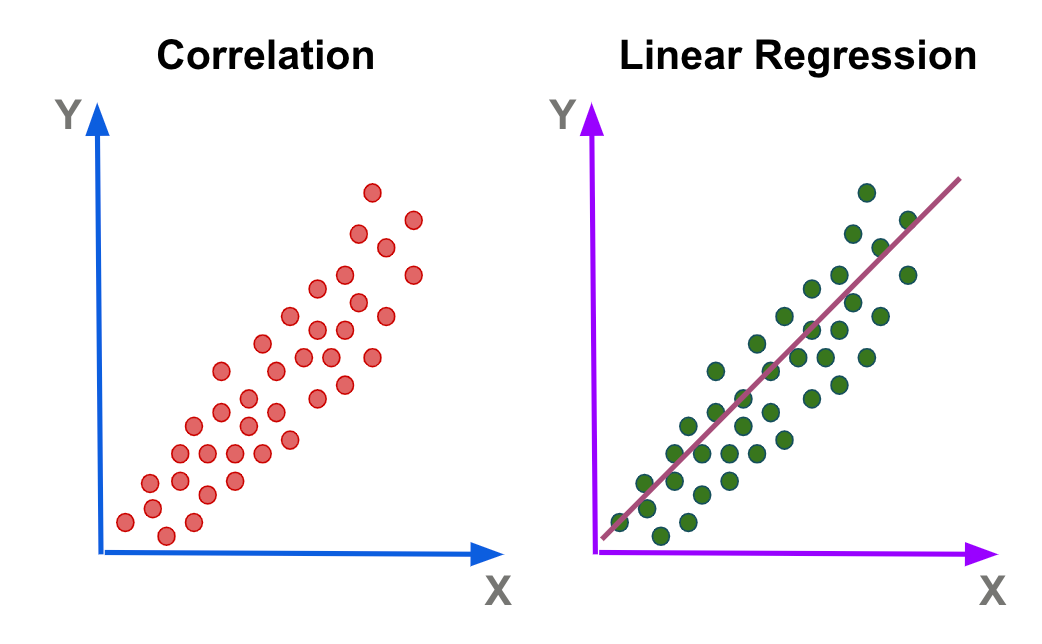
**Working Principle:** Linear regression models the relationship between the independent variable(s) and the dependent variable as a linear equation. The goal is to find the coefficients that minimize the difference between the observed and predicted values.

**Key Features**:

**Simplicity:** Linear regression is straightforward and easy to interpret.

**Interpretability:** Coefficients in the model represent the change in the dependent variable for a one-unit change in the corresponding independent variable.

**Assumption:** Assumes a linear relationship between variables.



**Decision Tree:**

**Type:** Classification and Regression

**Use Case:** Widely used for both classification and regression tasks. Decision trees are employed in scenarios where decisions need to be made based on input features, leading to different outcomes.

**Working** Principle: Decision trees recursively split the dataset into subsets based on the most significant feature, creating a tree-like structure of decision nodes. Each node represents a decision based on a specific feature, and each leaf node represents the final outcome.

**Key Features:**

Interpretability: Decision trees are highly interpretable, providing a clear decision-making logic.

Non-Linearity: Can capture non-linear relationships in the data.

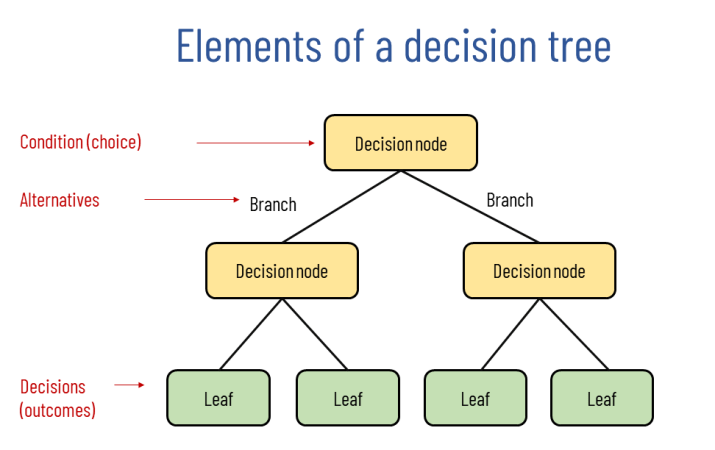
Handling Non-Numeric Data: Can handle both numeric and categorical data.

**Hyperparameters:**

Depth: The maximum depth of the tree.

Criterion: The measure used to evaluate the quality of a split (e.g., Gini impurity for classification, mean squared error for regression).

Splitting Strategy: The method for selecting the best split at each node.



**Reasons for Using Multiple Models:**

Diversity in Approaches:

Each model makes different assumptions and uses distinct mechanisms for classification. Using multiple models provides a diverse set of approaches to capturing patterns in the data.

Ensemble Learning:

Combining the predictions of multiple models can result in a more robust and accurate overall model. Ensemble methods like Random Forest naturally leverage this principle.

Model Comparison:

By implementing different models, you can compare their performances using metrics like accuracy, precision, recall, and F1-score. This helps identify the most suitable model for your specific hate speech recognition task.

Handling Complex Data:

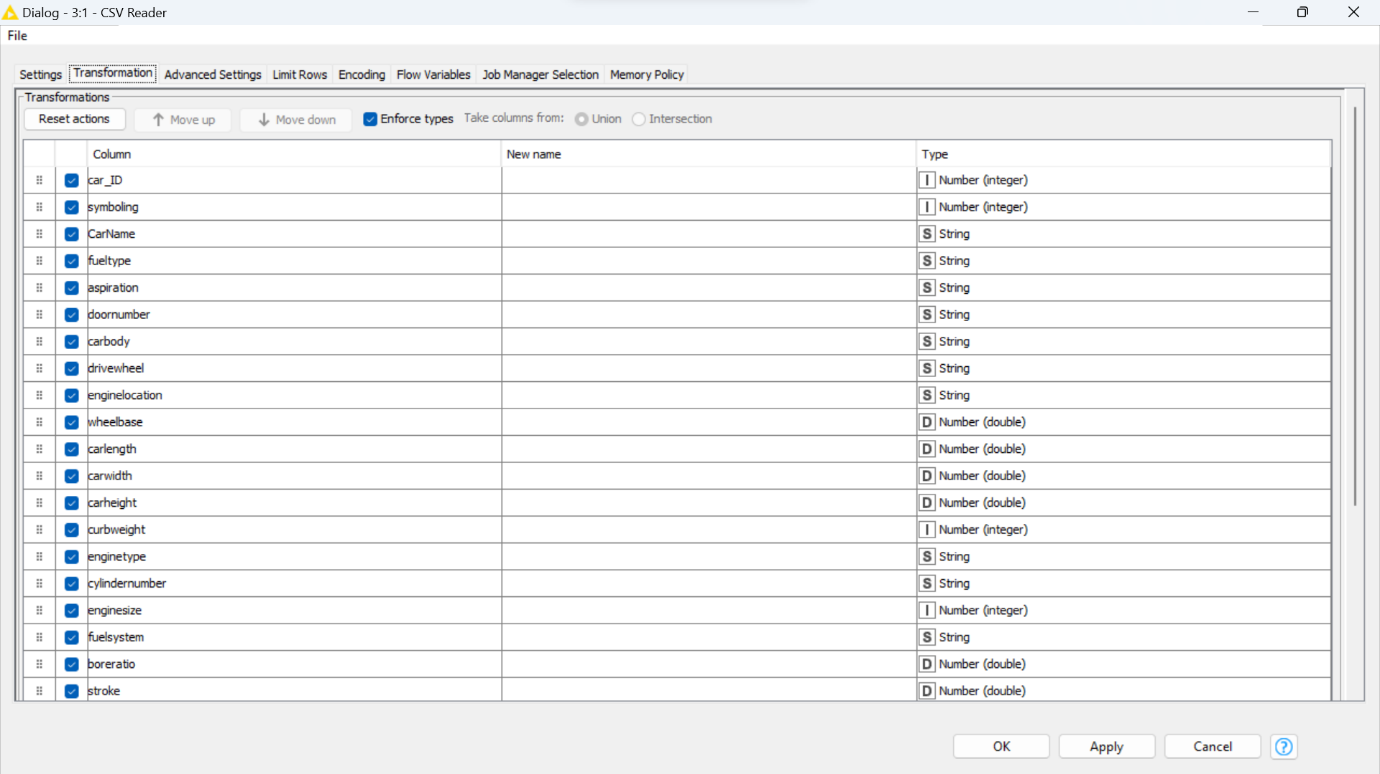
Hate speech data can be complex, and different models may excel in capturing different aspects of this complexity. Employing multiple models allows you to address the diversity of hate speech content.  
In summary, the use of SVM, Random Forest, and Naive Bayes in your hate speech recognition project reflects a thoughtful approach to model selection. These models, with their distinct characteristics, provide a comprehensive strategy for addressing the challenges inherent in identifying and classifying hate speech. The final choice may depend on empirical performance results and considerations specific to your project requirements.

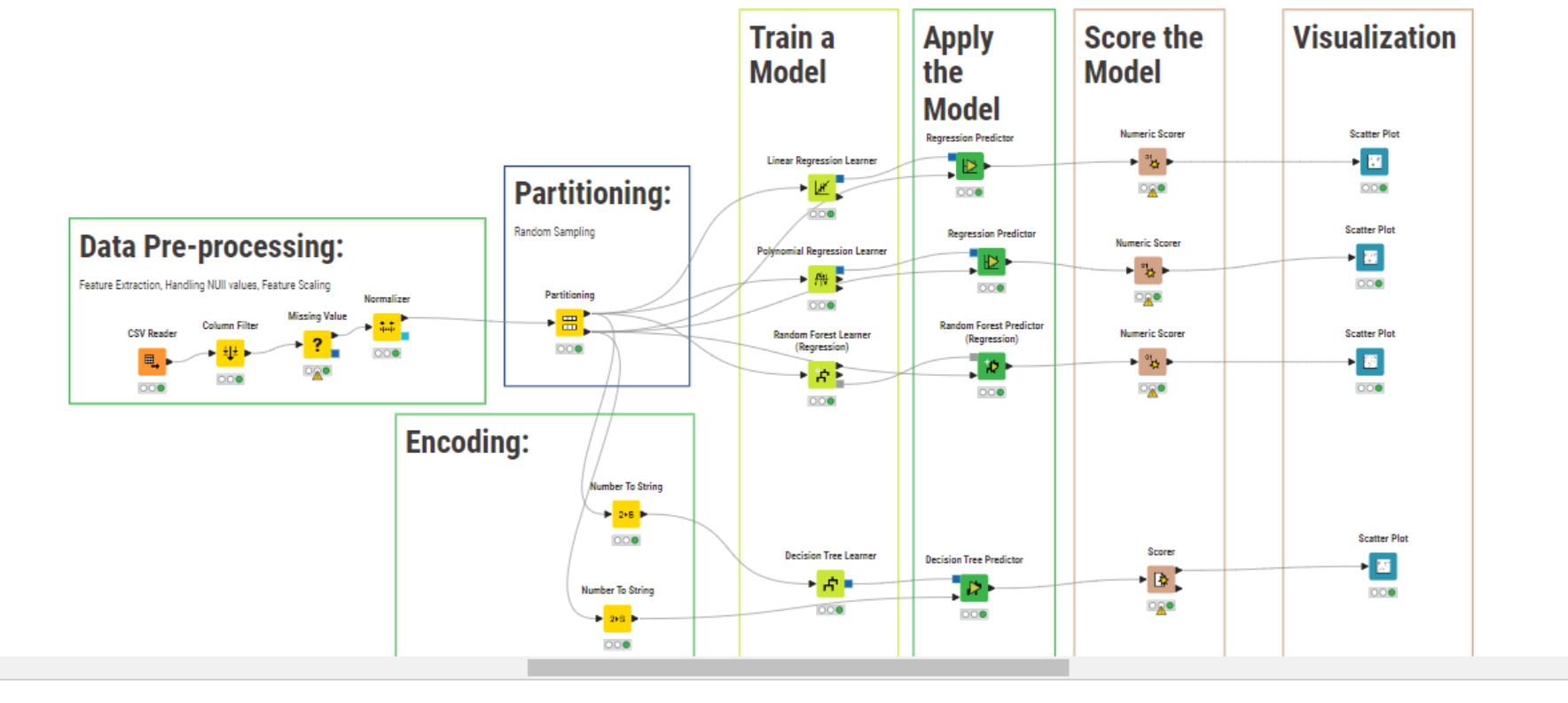
**Dataset Overview:**

**The dataset consists of the following columns:**

1. car\_ID: Identifier for each car in the dataset.
2. symboling: Risk rating associated with the car (e.g., +3 indicates a riskier car, -3 indicates a safer car).
3. CarName: Name of the car.
4. fueltype: Type of fuel the car uses (e.g., gas, diesel).
5. aspiration: Type of aspiration for the engine (e.g., std for standard, turbo for turbocharged).
6. doornumber: Number of doors on the car.
7. carbody: Body style of the car (e.g., sedan, hatchback).
8. drivewheel: Type of drive wheel the car has (e.g., 4wd for four-wheel drive, fwd for front-wheel drive).
9. enginelocation: Location of the car's engine (e.g., front, rear).
10. wheelbase: Distance between the centers of the front and rear wheels.
11. carlength: Length of the car.
12. carwidth: Width of the car.
13. carheight: Height of the car.
14. curbweight: Weight of the car without occupants or baggage.
15. enginetype: Type of engine (e.g., ohc for overhead camshaft, ohcf for overhead camshaft front).
16. cylindernumber: Number of cylinders in the engine.
17. enginesize: Size of the engine.
18. fuelsystem: Fuel system of the car (e.g., mpfi, 2bbl).
19. boreratio: Bore ratio of the engine.
20. stroke: Stroke length of the engine.
21. compressionratio: Compression ratio of the engine.
22. horsepower: Horsepower of the car.
23. peakrpm: Peak revolutions per minute.
24. citympg: Miles per gallon in the city.
25. highwaympg: Miles per gallon on the highway.
26. price: Price of the car (target variable for regression).

Screenshots:





Certainly! Here's a structured flow for your project on a car dataset:

**PROJECT FLOW:**

**1. Data Preprocessing:**

- Column Filter: Select relevant columns/features for analysis (e.g., 'price', 'enginetype', numerical attributes).

- Handling Null Values: Address missing data by imputation or deletion based on context.

- Normalization: Scale numerical features to a common range (e.g., using Min-Max or Standard Scaling) to ensure all variables contribute equally.

**2. Partitioning the Dataset:**

- Random Sampling (70-30 Split): Divide the dataset into training (70%) and testing (30%) sets to facilitate model training and evaluation.

**3. Encoding:**

- Convert categorical variables like 'enginetype' into numerical representations using techniques like one-hot encoding or label encoding.

**4. Applying Multiple Machine Learning Models:**

- Numerical Data Prediction (Price):

- Linear Regression: Utilize linear regression to predict the car prices based on numerical features.

- Polynomial Regression: Employ polynomial regression to capture non-linear relationships in predicting car prices.

- Random Forest Regression: Apply a random forest regressor to handle complex interactions and predict car prices.

- Categorical Data Prediction (Engine Type):

- Decision Tree Classifier: Use a decision tree classifier to predict 'enginetype' based on categorical attributes.

**5. Model Evaluation:**

- Accuracy Calculation: Evaluate the performance of each model separately. For regression models (Linear, Polynomial, Random Forest), assess accuracy using metrics like Mean Squared Error (MSE), R-squared, or Root Mean Squared Error (RMSE). For the Decision Tree Classifier, assess accuracy, precision, recall, and F1-score for the 'enginetype' prediction.

**6. Comparison of Models:**

- Compare the accuracy and performance of each model to determine which one best fits the dataset and problem context. This helps in selecting the most suitable model for further analysis or deployment.

**7. Visualization:**

- Create visualizations such as scatter plots, regression lines, or confusion matrices to illustrate the model's predictions, their accuracy, and any relationships discovered between features and the target variable. Visual aids help in conveying insights effectively.

This structured project flow enables a step-by-step approach to preprocess data, build models, evaluate their accuracy, compare their performance, and visualize the results, providing a comprehensive analysis of the car dataset and its predictive capabilities.

**DETAILED NODE-TO-NODE EXPLANATION:**

CSV READER:

* CSV Reader Node: Imports CSV files (car-price-prediction.csv) in KNIME workflows.
* File Path/Input: Specifies the CSV file location or URL.
* Configuration Options: Allows setting delimiters, quote characters, and column headers.
* Data Types Inference: Automatically infers column data types but allows manual specification.
* Output: Generates a KNIME data table for further processing.
* The node simplifies data import, enabling users to read CSV files, adjust settings, and manipulate data within the KNIME environment seamlessly for subsequent analysis or manipulation.

COLUMN FILTER:

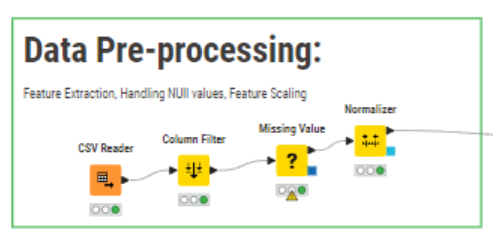
* Column Filter Node in KNIME: Selects or excludes specific columns (REMOVES Car ID) in a dataset.
* Functionality: Facilitates selection of desired columns for analysis or processing.
* Configuration Options: Allows manual selection or exclusion of columns.
* Regular Expressions: Offers flexibility to filter columns based on specific patterns or criteria.
* Efficiency Improvement: Reduces dataset clutter by outputting a refined dataset with chosen columns.
* This node simplifies data preparation by allowing users to streamline datasets, keeping only relevant columns for analysis, thereby enhancing workflow efficiency, and focusing on essential data attributes within the KNIME environment.

MISSING VALUE :

* Missing Value Node in KNIME: Manages missing data within a dataset efficiently.
* Configuration Options: Allows selection of columns for treatment and choosing imputation methods.
* Imputation Methods: Offers diverse techniques like mean, median, mode, constant value, or advanced approaches like KNN or predictive modeling.
* Exclude Rows: Optionally excludes rows with missing values from analysis.
* Column-Specific Treatment: Permits customized handling of missing values for specific columns.
* This node streamlines data preprocessing by providing various methods to handle missing values, ensuring datasets are suitably treated for further analysis or modeling within the KNIME workflow environment.

NORMALIZER :

* Normalizer Node in KNIME: Scales numerical data within a specified range or distribution.
* Functionality: Standardizes or normalizes numerical attributes for consistent analysis.
* Configuration Options: Allows selection of columns for normalization and method choice.
* Normalization Methods: Offers options like Min-Max Scaling, Z-Score Standardization, and Decimal Scaling.
* Output: Generates a dataset with scaled numerical attributes, ensuring uniformity and comparability across features.
* This node enhances data preprocessing by transforming numerical data into a standardized range or distribution, facilitating fair comparison and analysis of attributes within the KNIME workflow environment.



PARTITIONING :

* Partitioning Node in KNIME: Splits datasets into training and testing subsets for model evaluation.
* Functionality: Divides data into customizable proportions for model training and validation.
* Configuration Options: Allows percentage allocation for training and testing datasets.
* Random Sampling: Facilitates random sampling of data for unbiased training and testing sets.
* Output: Produces segregated datasets, aiding in robust model development and performance assessment.
* This node streamlines the machine learning workflow by partitioning data, enabling effective model training on one subset while evaluating performance on another, ensuring reliable model assessment within the KNIME environment.

A white paper with black text and yellow box

Description automatically generated

LINEAR REGRESSOR LEARNER:

* Linear Regression Node in KNIME: Performs linear regression modeling for predictive analysis.
* Functionality: Fits a linear relationship between input variables and a continuous target variable.
* Configuration Options: Allows selection of predictor and target columns, customization of model parameters.
* Model Interpretation: Provides coefficients indicating the impact of predictors on the target variable.
* Output: Generates a linear regression model for prediction within the KNIME workflow, aiding in understanding relationships between variables and making predictions based on the learned patterns.

A diagram of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

REGRESSOR PREDICTOR:

* Regressor Predictor Node in KNIME: Utilizes trained regression models for predictions on new data.
* Functionality: Applies regression models created in KNIME to make predictions on unseen or test datasets.
* Model Application: Enables the use of various regression models like Linear Regression, Random Forest, or Gradient Boosting for predictions.
* Input Data: Requires input of a trained regression model and new data for prediction.
* Output: Provides predictions based on the input data using the specified regression model, aiding in forecasting or estimation tasks within the KNIME environment.

A diagram of a forest predictor

Description automatically generatedA screenshot of a graph

Description automatically generated

SCORER:

* Scorer Node in KNIME: Evaluates model performance using various metrics.
* Evaluation Metrics: Calculates performance measures like accuracy, precision, recall, F1-score, RMSE, R-squared, etc.
* Input Data: Takes predicted and actual target values for model comparison.
* Configuration Options: Allows selection of specific evaluation metrics based on the model's purpose.
* Output: Provides a detailed report on model performance, aiding in model selection or refinement within the KNIME workflow environment.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

DECISION TREE LEARNER:

* Decision Tree Learner Node in KNIME: Constructs decision tree-based models for classification or regression.
* Functionality: Builds decision trees based on input features and target variables.
* Configuration Options: Allows customization of tree depth, splitting criteria, and other parameters.
* Input Data: Requires labeled datasets for supervised learning tasks.
* Output: Generates a decision tree model that partitions data based on attribute values, aiding in classification or regression tasks within the KNIME environment. The resulting tree illustrates decision pathways, facilitating interpretation and prediction on new or unseen data.

A screenshot of a computer

Description automatically generated

**References:**

**Kaggle Dataset** : car price Prediction

**Google** : <https://www.google.com/>

**Towards Data Science** : <https://www.kaggle.com/datasets/hellbuoy/car-price-prediction>

**GeeksforGeeks (To learn about ML Algorithm Models) :** <https://www.geeksforgeeks.org>