Automobile Price Prediction

Prepared by

Atharva Shinde

Internship Program

Spinnaker Analytics

Date

09/ 03/ 2024

ABSTRACT

An Automobile price prediction has been a high interest research area, as it requires noticeable effort and knowledge of the field expert. Considerable number of distinct attributes are examined for the reliable and accurate prediction. To build a model for predicting the price of cars, we applied machine learning techniques (Linear Regression, K Neighbor regressor, Gradient Boosting Regressor and Random Forest, Regressor). However, the mentioned techniques were applied to work as an ensemble. Respective performances of different algorithms were then compared to find one that best suits the available data set. The final prediction model was integrated into Java application. Furthermore, the model was evaluated using test data and the accuracy of 93% was obtained. Cars of a particular make, model, year, and set of features start out with a price set by the manufacturer. Using Machine Learning algorithms to better utilize data on all the less common features of a car can more accurately assess the value of a vehicle. This study compares the performance of Linear Regression, Ridge Regression, Lasso Regression, and Random Forest Regression ML algorithms in predicting the price of used cars. An important qualification of a price prediction tool is that depreciation can be represented to better utilize past data for current price prediction. The study has been conducted with a large public dataset of cars. The results show that Random Forest Regression demonstrates the highest price prediction performance across all metrics used. It was also able to represent average depreciation much more closely than the other algorithms, at 7% predicted annual geometric depreciation for the dataset.

ACKNOWLEDGEMENT

I would like to sincerely thanks the author and express gratitude to Kaggle Platform for providing the dataset used in our Automobile Price Prediction project. The dataset, sourced from Kaggle's website, served as the foundation for our analysis and model development. We acknowledge the contributors who curated and made the dataset publicly available, enabling researchers and practitioners to explore and innovate in the field of machine learning and automotive analytics.

My sincere thanks to Spinnaker Analytics for providing me with an opportunity to deep dive into the world of problem solving, leveraging the skillset and analytical mindset. Your assistance has been crucial in enabling to achieve the project goals and objectives.

I extend the appreciation to the academic and research community for their contributions to the field of Automobile Price Prediction detection. The insights, methodologies, and best practices shared by researchers and practitioners have been instrumental in informing our approach and guiding our decision-making process.

CONTENTS

INTRODUCTION

The automobile market is a growing business with a market value that has nearly doubled itself in previous years. The rise of online websites and other tools like it have made it easier for both buyers and sellers to get a better understanding of the factors that determine the market value of a used car. Based on a set of factors, Machine Learning algorithms may be used to forecast the price of any automobile. The data set will include information on a variety of automobiles. There will be information regarding the vehicle's technical elements, such as the engine type, fuel type, the kilometers per liter, and more, for each car. There is no universal mechanism for establishing the retail price of used automobiles because different websites employ different methods to create it. By using statistical models to anticipate pricing, it is possible to obtain a preliminary price estimate without having to enter all of the details into the desired website. The main purpose of this study is to compare the accuracy of two different prediction models for estimating a used car's retail price.

Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors. Typically, most significant ones are brand and model, age, horsepower and mileage. The fuel type used in the car as well as fuel consumption per mile highly affect price of a car due to a frequent change in the price of fuel. Different features like exterior color, door number, type of transmission, dimensions, safety, air condition, interior, whether it has navigation or not will also influence the car price. In this paper, we applied different methods and techniques in order to achieve higher precision of the used car price prediction. This project aims to deliver:

- price prediction model to the public
- new methods to evaluate used cars prices and to compare their accuracies.

to help guide the individuals looking to buy or sell cars and to give them a better insight into the automotive sector. Buying a used car from a dealer can be a frustrating and an unsatisfying experience as some dealers are known to deploy deceitful sale tactics to close a deal. Therefore, to help consumers avoid falling victims to such tactics, this study hopes to equip consumers with right tools to guide them in their shopping experience. Considering this is an interesting research topic in the research community, and in continuing their footsteps, we hope to achieve significant results using more advanced methods of previous work

LITERATURE REVIEW

With the recent arrival of internet portals, buyers and sellers may obtain an appropriate status of the factors that ascertain the market price of a used automobile. Lasso Regression, Multiple Regression, and Regression Trees are examples of machine learning algorithms. We will try to develop a statistical model that can forecast the value of a pre-owned automobile based on prior customer details and different parameters of the vehicle. This paper aims to compare the efficiency of different models' predictions to find the appropriate one. On the subject of used automobile price prediction, several previous studies have been conducted. To anticipate the value of automobiles employed Multiple Linear Regression, K-Nearest Neighbor, Random Forest Regression and Decision Trees. However, because there were fewer cars observed, their results were not good for prediction. In his article, it is concluded that Random Forest, Decision Trees and K Nearest Neighbor are effective for continuous-valued variables. To anticipate the price of a vehicle, employed Multiple Linear Regression which used a variable selection methodology to determine the variables that had the highest influence and then eliminated the remainder. Only a few variables are included in the data, which were utilized to create the linear regression model. With an R-square of 72 percent, the outcome was good. However, especially on higher-priced cars, the estimated value is not very close to the real price. In forecasting the price of a car, they found that Random Forest Regression out performed K Neighbor and Linear Regression by a little margin. To accurately anticipate the price of a car, many different approaches have been used in the digital world, ranging from Machine Learning approaches like Multiple Linear Regression, K-Nearest Neighbor, Random Forest and Decision Tree to the SAS enterprise miner. In all of these solutions took into account distinct sets of attributes when making predictions based on the historical data used to train the model and using a model for prediction based on the factors that have the greatest impact on vehicle prices.

DATA COLLECTION

The dataset contains automobile features including technical terms and terminologies. This dataset presents features that are market oriented, where we have total of 11,915 input records. The dataset is highly unbalanced, as it contains missing values, duplicates and some outlier values resulting in the unscaled dataset.

https://www.kaggle.com/datasets/CooperUnion/cardataset/data

```
# First 5 rows
dt.head()
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

Fig. 3.1 Overview of the dataset

Make: manufacturer / brand of the automobile

Model: specific model name or designation

Year: manufacturing year of the model

Fuel: type of fuel used by the vehicle

HP: engine power measured in horsepower

Cylinders: number of cylinders in the engine

Transmission: type of transmission system

Drive: drive type of the vehicle

Doors: number of doors

Market Category: vehicle market segment or category

Size: size or category of the vehicle

Style: body style or design of the automobile

Highway MPG: fuel efficiency on the highway

City MPG: fuel efficiency of the vehicle in city

Popularity: popularity or demand for the model

Price: retail price or cost of the automobile

DATA PREPROCESSING

Missing Values: Filling missing values involves replacing empty or null entries in the dataset with appropriate values to ensure completeness and accuracy of the data. For numerical features like Engine HP, Engine Cylinders, Market Category missing values can be filled using statistical measures such as mean, median, or mode of the respective column. For categorical features like Make, Model, Year, Fuel, Transmission, Drive, Doors, Market Category, and Style, missing values can be filled with the most frequent category or a new category denoting missing values. The missing values were filled with the mean for the numerical features and with the mode for the categorical features using the function '.fillna()'.

```
# Missing values
print('Number of missing values :')
df.isnull().sum()
```

```
Number of missing values :
Make                    0
Model                   0
Year                    0
Engine Fuel Type        3
Engine HP              69
Engine Cylinders       30
Transmission Type       0
Driven_Wheels           0
Number of Doors         6
Market Category      3376
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

Fig. 4.1 Missing value count in the data

The irrelevant feature Market category was dropped from the data using the function '.drop()' as it was making no sense in terms of future aspects and model building.

```
# Dropping irrelevant feature
df.drop('Market Category', axis = 1, inplace = True)
```

```
# Filling the missing values
df['Engine Fuel Type'] = df['Engine Fuel Type'].fillna('regular unleaded')
df['Engine HP'] = df['Engine HP'].fillna(0)
df['Engine Cylinders'] = df['Engine Cylinders'].fillna(0)
df['Number of Doors'] = df['Number of Doors'].fillna(df['Number of Doors'].mean())
```

Fig. 4.2 Filling of missing data

Duplicate Values: Dropping duplicated values eliminates duplicate rows from the dataset, ensuring data integrity and consistency. Similarly, dropping irrelevant columns removes features that do not contribute to the analysis or prediction task. Duplicated rows can be identified using the 'duplicated()' function and removed using the 'drop_duplicates()' method. Irrelevant columns, such as Make and Model (if not required for analysis), can be dropped using the '.drop()' method.

```
# Duplicate values
df.duplicated().sum()
```

```
715
```

```
# Dropping the Duplicate values
df = df.drop_duplicates()
```

```
df.shape
```

```
(11199, 16)
```

Fig. 4.3 Deletion of duplicate values

Column Formatting: Column formatting involves converting the data type or format of columns to a suitable representation for analysis and visualization. Numeric columns like Year, HP, Cylinders, Highway MPG, City MPG, Popularity, and Price should be converted to numeric data types (integer or float) for mathematical operations and analysis. Categorical columns like Fuel, Transmission, Drive, Market Category, Size, and Style should be converted to categorical data types for better memory utilization and efficient processing. The feature names were transformed into a better and simple format for better understanding and analysis of the data.

```
# Renaming the columns
df.rename(columns={'year' : 'Year', 'model' : 'Model', 'make' : 'Make', 'Engine Fuel Type' : 'Fuel',
                   'Engine HP' : 'HP', 'Engine Cylinders' : 'Cylinders', 'Transmission Type' : 'Transmission',
                   'Driven_Wheels' : 'Drive', 'Number of Doors' : 'Doors', 'Vehicle Size' : 'Size',
                   'Vehicle Style' : 'Style', 'highway MPG' : 'Highway MPG', 'city mpg' : 'City MPG', 'MSRP' : 'Price'}, inplace = True)
```

Fig. 4.4 Formatting the feature name

Outlier Detection: Outlier deletion aims to remove extreme or erroneous data points that deviate significantly from the rest of the dataset, which can distort analysis and modeling results. Outliers can be identified using statistical methods such as Z-score, IQR (Interquartile Range), or visualization techniques like box plots and scatter plots. Once identified, outliers can be removed from the dataset using filtering or truncation methods

to ensure the robustness and reliability of the analysis. In this data, we used Inter-Quartile Method for the identification and removal of outliers.

```
# Box Plot Distribution for Outlier detection
for i in num_col:
    fig = px.box(df, x = df[i])
    fig.update_traces(fillcolor = '#C9A26B')
    fig.show()
```

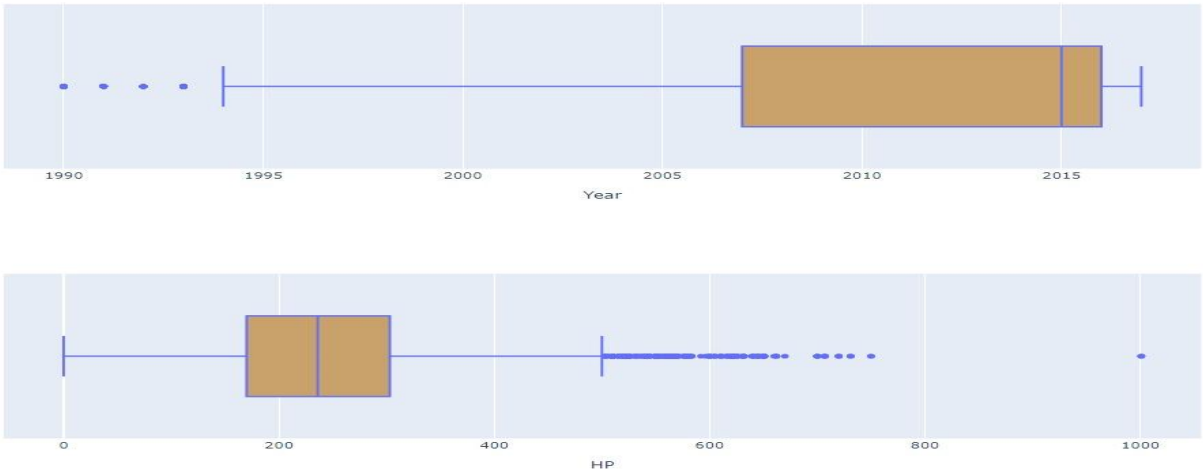


Fig. 4.5 Identification of Outlier values

```
# Deletion of Outliers
s1 = df.shape
clean = df[['HP', 'Cylinders', 'Highway MPG', 'City MPG', 'Price']]
for i in clean.columns:
    qt1 = df[i].quantile(0.25)
    qt3 = df[i].quantile(0.75)
    iqr = qt3 - qt1
    lower = qt1-(1.5*iqr)
    upper = qt3+(1.5*iqr)
    min_in = df[df[i]<lower].index
    max_in = df[df[i]>upper].index
    df.drop(min_in, inplace = True)
    df.drop(max_in, inplace = True)
s2 = df.shape
outliers = s1[0] - s2[0]
print("Deleted outliers are : ", outliers)
```

```
Deleted outliers are :  1403
```

```
df.shape
```

```
(9784, 15)
```

Checking the changes after outliers deletion from a dataset column by plotting the boxplot again

```
fig = px.box(df, x = df['HP'])
fig.update_traces(fillcolor = '#C9A26B')
```

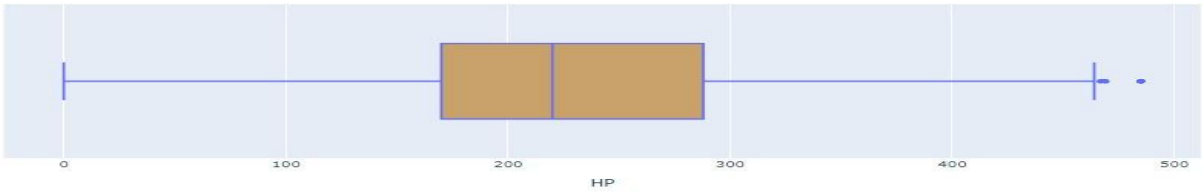


Fig. 4.6 Deletion of Outliers

Created a Quartile range from 0.25 to 0.75 for the identification of the outliers. The deletion outliers resulted in removal of redundant data and achievement of scaled data.

EXPLORATORY DATA ANALYSIS

In statistics, **Exploratory Data Analysis** (EDA) is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling and thereby contrasts traditional hypothesis testing. Exploratory Data Analysis (EDA) is an analysis approach that identifies general patterns in the data. These patterns include outliers and features of the data that might be unexpected. EDA is an important first step in any data analysis. Understanding where outliers occur and how variables are related can help one design statistical analyses that yield meaningful results.

Fig. 5.1 Advantages of EDA

Univariate Analysis and Bivariate Analysis are two fundamental methods used in EDA to gain insights into the data and understand its characteristics. Multivariate analysis deals with the interactions and relationships among three or more variables. The choice of analysis method depends on the research question, the nature of the data, and the goals of the analysis.

Univariate Analysis: Univariate analysis focuses on analyzing individual variables or features in the dataset. It helps in understanding the distribution, central tendency, and spread of each variable independently. For the given automobile dataset, univariate analysis can involve the following tasks:

For numerical variables like Year, HP, Cylinders, Highway MPG, City MPG, Popularity, and Price: Calculate descriptive statistics such as mean, median, mode, standard deviation, minimum, and maximum values. Create histograms, box plots, or density plots to visualize the distribution of each numerical variable.

For categorical variables like Make, Model, Fuel, Transmission, Drive, Doors, Market Category, Size, and Style: Generate frequency tables or bar charts to display the count or percentage of each category within the variable.

Univariate Analysis

```
# Distribution of Features
for i in df:
    fig = px.histogram(df, x= i, color_discrete_sequence = ['#C9A26B'])
    fig.show()
```

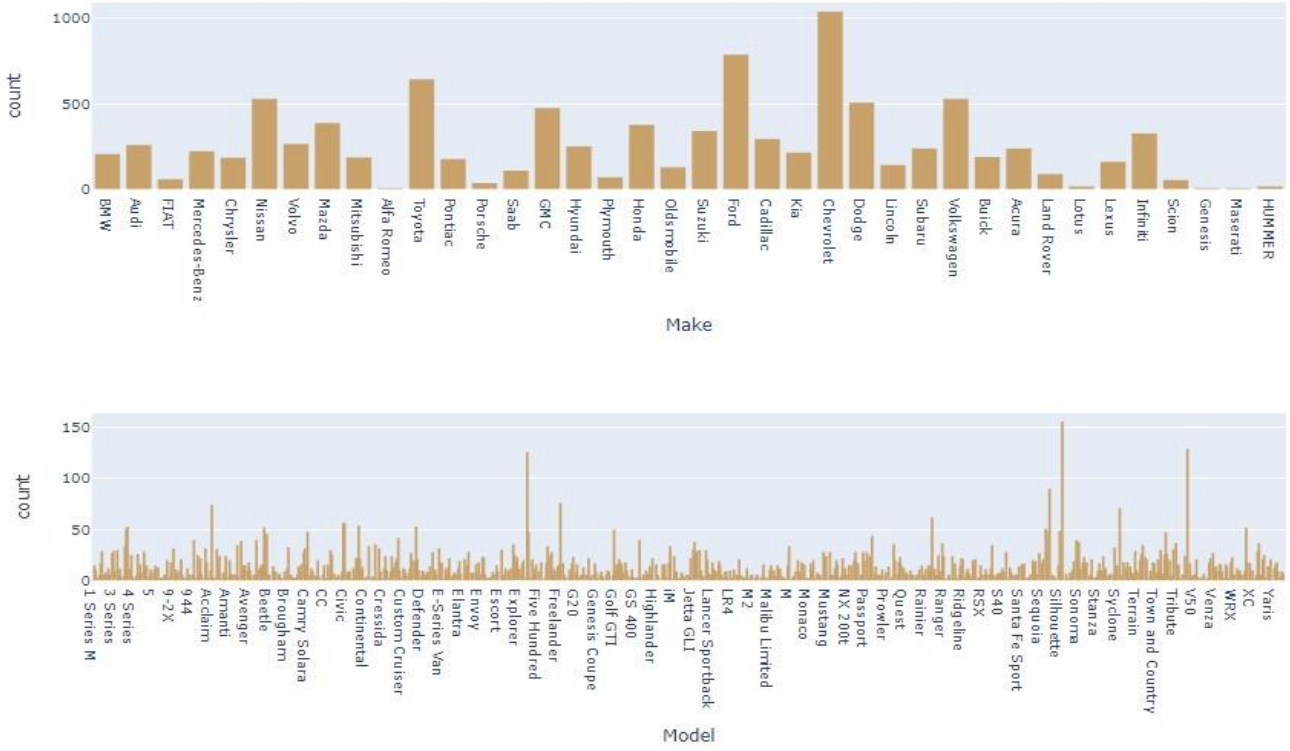


Fig. 5.2 Univariate Analysis of the features

- Make: The least count of vehicles belongs to Maserati, while Chevrolet boasts the maximum count, indicating a diverse range of manufacturers present in the dataset.
- Model: The data comprises numerous models, with Silverado 1500 being the most prevalent among them.
- Year: The dataset spans from 1990 to 2016, with the highest count recorded for vehicles manufactured in 2016.
- Fuel: Regular Unleaded is the most common fuel type, while Natural Gas is the least prevalent.
- HP and Cylinders: The dataset features a wide range of horsepower and cylinder configurations, with a maximum count of 200 horsepower and 4 cylinders.
- Transmission and Drive: Automatic transmission and front-wheel drive are the predominant configurations among the vehicles.
- Doors and Size: Vehicles with 4 doors and a Compact size are the most abundant in the dataset.
- Style: Sedans are the most common style, while Convertible SUVs are the least prevalent.
- Highway MPG and City MPG: The dataset covers a range of MPG values, with 24 MPG on the highway being the most frequent, and 17 MPG in the city.
- Popularity and Price: The dataset reflects varying levels of popularity and price points, with a maximum count of popularity attributed to a value of 1385, and a maximum price of 2000.

Bivariate Analysis: Bivariate analysis involves analyzing the relationship between two variables in the dataset. It helps in understanding how one variable impacts another and identifies any correlations or associations between them. The bivariate analysis can include the following approaches:

For numerical variables: Use scatter plots or correlation matrices to visualize the relationship between pairs of numerical variables, such as HP vs. Price or Highway MPG vs. City MPG. Calculate correlation coefficients (e.g., Pearson correlation) to quantify the strength and direction of the linear relationship between numerical variables.

For numerical-categorical combinations: Use box plots or violin plots to compare the distribution of numerical variables across different categories of categorical variables, such as Price across different Makes or HP across different Transmission types.

For categorical-categorical combinations: Create contingency tables or stacked bar charts to analyze the relationship between two categorical variables, such as Fuel type vs. Drive type or Market Category vs. Style.

Bivariate Analysis

According to HP

```
# Distribution of Feature w.r.t. to other feature
fig = go.Figure()
fig.add_trace(go.Bar(x = df['Make'].unique(), y = df.groupby('Make')['HP'].mean().sort_values()[:5], marker_color = '#C9A26B'))
fig.update_layout(xaxis_title = "Car Brand", yaxis_title = "HP", width = 700, height = 500)
```

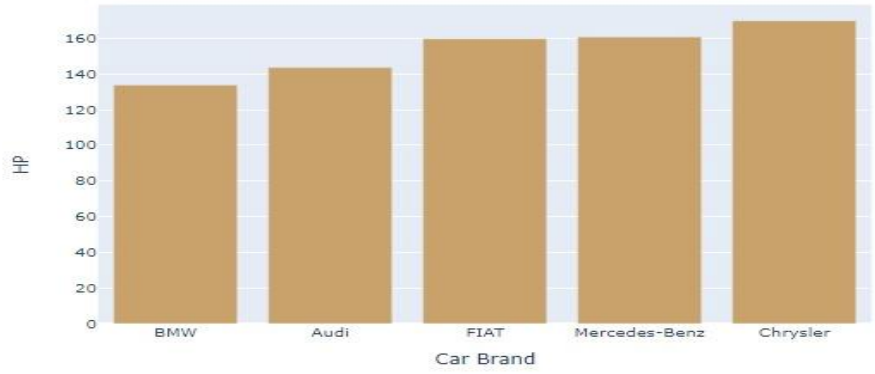


Fig. 5.3 Bivariate Analysis between the features

- In terms of car brands, Chrysler stands out with the highest average price, reaching around 20K. Luxury models like the 190-Class command premium prices, averaging around 2000.
- Diesel-powered cars tend to have the highest prices, peaking at approximately 39.1K, indicating a preference for fuel efficiency and power.
- Cars with manual transmission boast prices around 18.3K, while those with automated-manual transmission reach significantly higher prices, averaging around 36K.
- Cars featuring rear-wheel drive commanding prices of about 23K, while four-wheel drive models average around 38.2K.
- Compact vehicles are priced competitively, averaging around 22.6K, whereas larger vehicles such as SUVs and sedans have higher average prices, reaching approximately 35.9K.
- Style of the car plays a role in pricing, with coupes typically priced around 14.8K, while passenger vans command higher prices, averaging around 37.1K.

```
fig = px.scatter(df, x = 'HP', y = 'Price', color = 'Cylinders')
fig.update_layout(width = 900, height = 500)
fig.show()
```

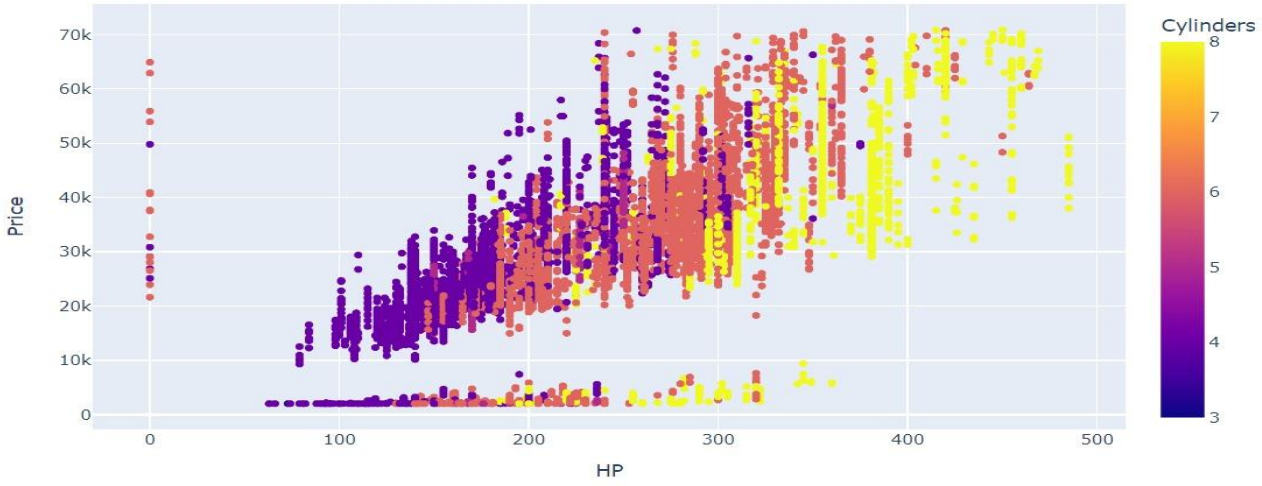
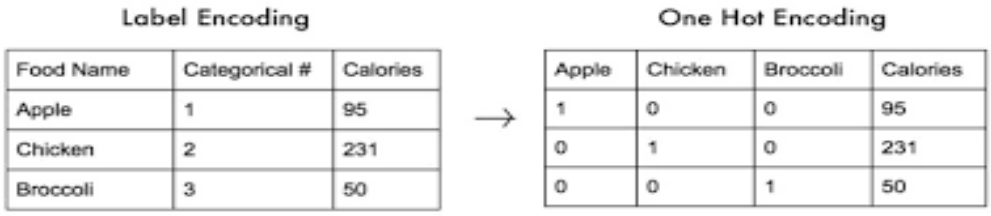


Fig. 5.4 Bivariate Analysis of HP and Price

- Increase in the number of cylinders from 2001, with 4 cylinders being predominant in 2015-2017.
- Car prices surged notably in 2001 and continued to rise steadily thereafter.
- Positive correlation observed between horsepower (HP) and price, with higher HP correlating to higher prices.
- Cars with over 290 HP tend to have 8 cylinders, while those under 180 HP are more likely to have 4 cylinders.
- Prices for cars with 4, 6, and 8 cylinders range from 2000 to 70K.
- Cars offering city MPG of 10 and 11 typically have 8 cylinders, while those with 17 MPG often have 6 cylinders.
- Cars with 4 cylinders exhibit excellent city MPG.
- Cars achieving highway MPG of 12-18 mostly have 8 cylinders, while those with 19-27 MPG typically have 6 cylinders.
- Cars with highway MPG of 28 or more often have 4 cylinders.
- Cars with popularity ratings of 204 and 6 cylinders range from 35K to 57K, while those with popularity ratings of 549 and 8 cylinders range from 42K to 70.2K.
- Cars with popularity ratings of 1385 and 8 cylinders are priced above 44K, while those with popularity ratings of 5657 and prices above 21K are more likely to have 6 cylinders.

FEATURE ENGINEERING

Feature engineering, a preprocessing step in supervised machine learning and statistical modeling, transforms raw data into a more effective set of inputs. It involves transforming raw data into a format that is suitable for training machine learning models. Each input comprises several attributes, known as features. By providing models with relevant information, feature engineering significantly enhances their predictive accuracy and decision-making capability.



Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Fig. 6.1 Encoding functions

Converting String Columns to Category: The string columns such as Make, Model, Fuel, Transmission, Drive, Category, Size, and Style need to be converted into categorical data types. This conversion helps in reducing memory usage and can improve the performance of certain machine learning algorithms.

```
# Covnerying the feature columns into category
columns_to_convert=['Make', 'Model', 'Fuel', 'Transmission', 'Drive', 'Size', 'Style']
df[columns_to_convert] = df[columns_to_convert].astype('category')
```

```
# Checking the datatypes
df.dtypes
```

```
Make            category
Model           category
Year               int64
Fuel            category
HP               float64
Cylinders        float64
Transmission    category
Drive           category
Doors            float64
Size            category
Style           category
Highway MPG        int64
City MPG           int64
Popularity         int64
Price              int64
dtype: object
```

Fig. 6.2 Conversion of features into category

Converting Category Columns to Numerical using Label Encoding: Once the string columns are converted to categorical data types, they can be further encoded into numerical values using label encoding. Label encoding assigns a unique numerical label to each category within a column. For example, the 'Transmission' column with categories like 'Manual', 'Semi-automatic', and ‘Automatic' can be encoded as 0, 1, and 2, respectively.

```
# Converting the Categorical features into numerical for Model Building
label_encoder = LabelEncoder()

df['Make'] = label_encoder.fit_transform(df['Make'])
df['Model'] = label_encoder.fit_transform(df['Model'])
df['Fuel'] = label_encoder.fit_transform(df['Fuel'])
df['Transmission'] = label_encoder.fit_transform(df['Transmission'])
df['Drive'] = label_encoder.fit_transform(df['Drive'])
df['Size'] = label_encoder.fit_transform(df['Size'])
df['Style'] = label_encoder.fit_transform(df['Style'])
```

```
df
```

	Make	Model	Year	Fuel	HP	Cylinders	Transmission	Drive	Doors	Size	Style	Highway MPG	City MPG	Popularity	Price
0	3	1	2011	7	335.0	6.0	2	3	2.0	0	8	26	19	3916	46135
1	3	0	2011	7	300.0	6.0	2	3	2.0	0	6	28	19	3916	40650
2	3	0	2011	7	300.0	6.0	2	3	2.0	0	8	28	20	3916	36350
3	3	0	2011	7	230.0	6.0	2	3	2.0	0	8	28	18	3916	29450
4	3	0	2011	7	230.0	6.0	2	3	2.0	0	6	28	18	3916	34500

Fig. 6.3 Label Encoding of features

By performing these feature engineering steps, the dataset is transformed into a format that is more suitable for training machine learning models. It helps improve model performance and ensures that the algorithms can effectively learn from the data to make accurate predictions, such as predicting the price of automobiles based on their features.

MODEL SELECTION

Linear Regression: Linear regression is a supervised machine learning method that finds a linear equation that best describes the correlation of the explanatory variables with the dependent variable. This is achieved by fitting a line to the data using least squares. In statistics, linear regression is a statistical model which estimates the linear relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable. If the explanatory variables are measured with error, then errors-in-variables models are required, also known as measurement error models.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

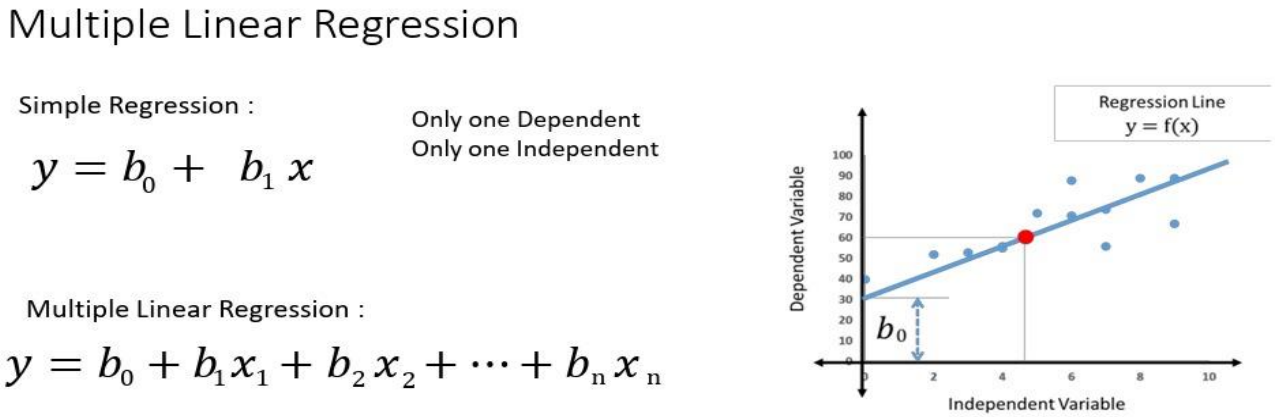


Fig. 7.1 Linear Regression

Given a data set $\{y_i,\ x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of *n* statistical units, a linear regression model assumes that the relationship between the dependent variable *y* and the vector of regressors **x** is linear. This relationship is modeled through a *disturbance term* or *error variable* ε — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\mathsf{T}}$ denotes the transpose, so that $\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$.

Often these *n* equations are stacked together and written in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

Fig. 7.2 Linear Regression mechanism

The model's performance is evaluated using the mean squared error (MSE) or mean absolute error (MAE), which measures the difference between the predicted and actual values. The goal is to minimize the cost function by adjusting the coefficients using optimization algorithms like gradient descent.

Ridge Regression:

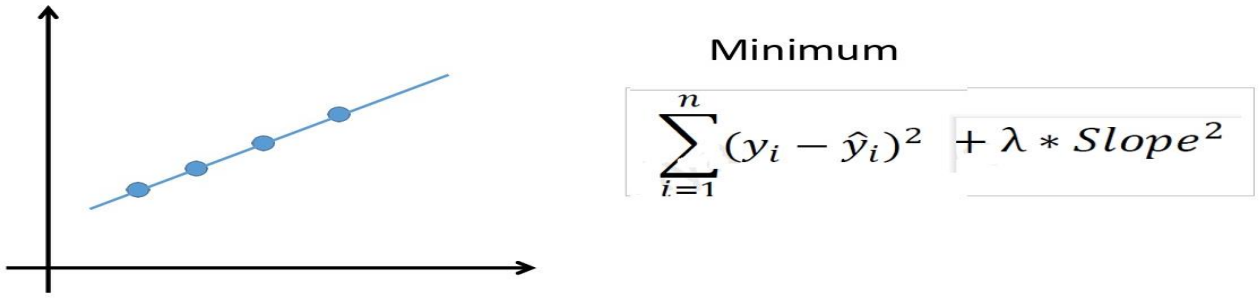
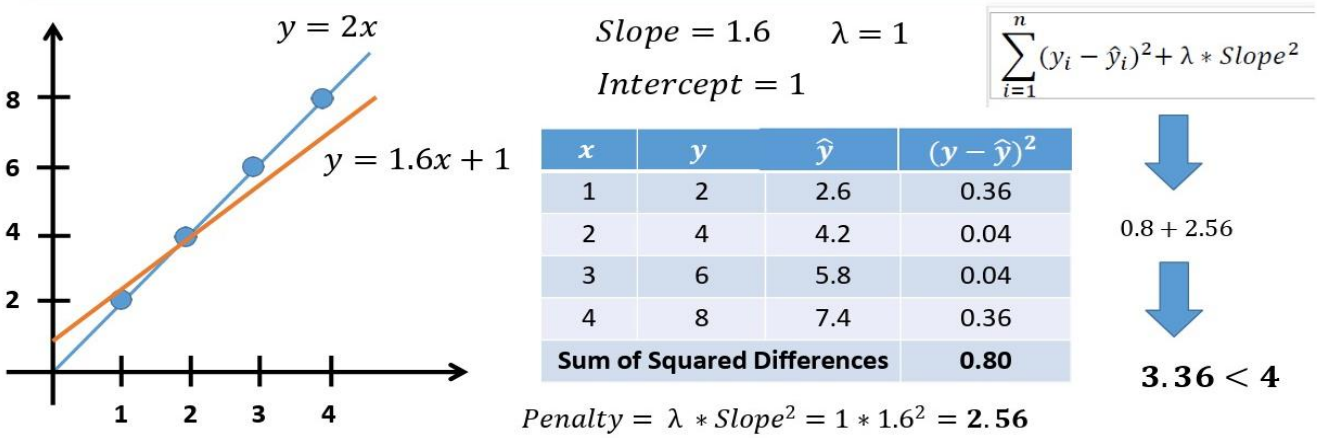


Fig. 7.3 Ridge Regression



x	y	$\hat{y}$	$(y - \hat{y})^2$
1	2	2.6	0.36
2	4	4.2	0.04
3	6	5.8	0.04
4	8	7.4	0.36
Sum of Squared Differences			**0.80**

Fig. 7.4 Ridge Regression working mechanism

Ridge regression adds a penalty term to the cost function, where λ is the regularization parameter and βi are the coefficients. This penalty term shrinks the coefficients, reducing model complexity. Ridge regression strikes a balance between reducing variance (overfitting) and increasing bias (underfitting) by controlling the complexity of the model. The regularization term is added to the cost function, and the model is optimized using techniques like gradient descent. Ridge regression is effective when dealing with multicollinearity (high correlation among predictor variables) and helps improve the model's generalization performance.

Lasso regression: Lasso regression, similar to ridge regression, is a regularization technique used to prevent overfitting by adding a penalty term to the cost function. However, lasso regression uses the L1 norm penalty instead of the L2 norm used in ridge regression.

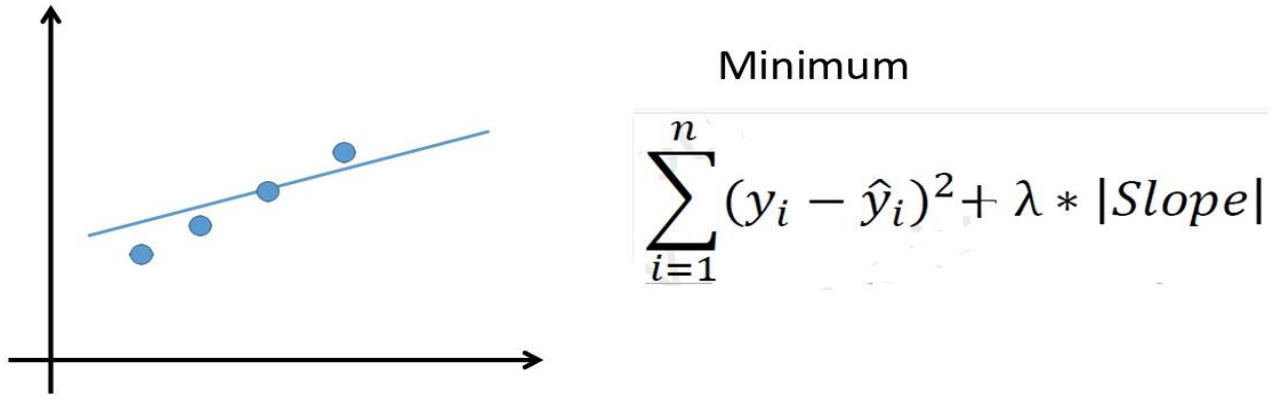


Fig. 7.5 Lasso regression

Lasso regression adds a penalty term to the cost function, where λ is the regularization parameter and βi are the coefficients. The L1 norm penalty encourages sparsity by forcing some coefficients to be exactly zero. Lasso regression can automatically perform feature selection by setting some coefficients to zero, effectively eliminating irrelevant features. The cost function with the L1 regularization term is minimized using optimization techniques like gradient descent. Lasso regression is beneficial in scenarios with a large number of features, where feature selection and sparsity are desirable.

Decision Tree Regressor: Decision tree learning is a supervised learning approach used in statistics, data mining and machine learning. In this formalism, a classification or regression decision tree is used as a predictive model to draw conclusions about a set of observations.

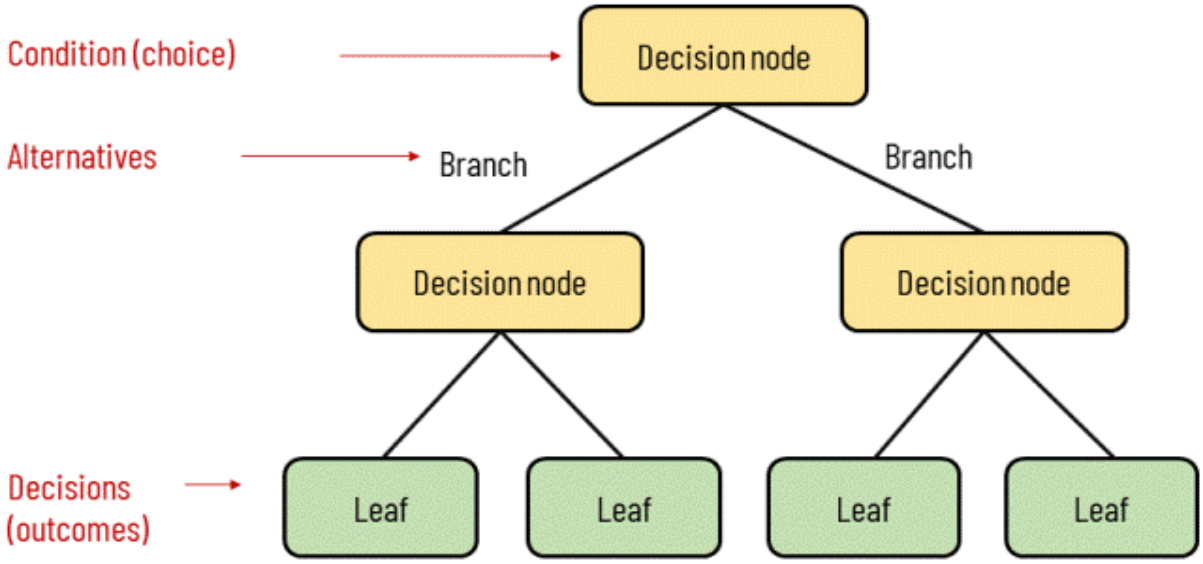


Fig. 7.6 Decision Tree Regressor

Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. More generally, the concept of regression tree can be extended to any kind of object equipped with pairwise dissimilarities such as categorical sequences.

Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making).

Fig. 7.7 Decision Tree Regressor mechanism

Advantages: Able to handle both numerical and categorical data.

- Requires little data preparation. Since trees can handle qualitative predictors, there is no need to create dummy variables.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Non-parametric approach that makes no assumptions of the training data or prediction residuals; e.g., no distributional, independence, or constant variance assumptions
- Performs well with large datasets. Large amounts of data can be analyzed using standard computing resources in reasonable time.
- Accuracy with flexible modeling. These methods may be applied to healthcare research with increased accuracy.
- In built feature selection. Additional irrelevant feature will be less used so that they can be removed on subsequent runs.

Limitations: Trees can be very non-robust. A small change in the training data can result in a large change in the tree and consequently the final predictions.

- Decision-tree learners can create over-complex trees that do not generalize well from the training data. This is known as overfitting.
- The average depth of the tree that is defined by the number of nodes or tests till classification is not guaranteed to be minimal or small under various splitting criteria.
- For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favor of attributes with more levels.
- This biases the decision tree against considering attributes with a large number of distinct values, while not giving an unfair advantage to attributes with very low information gain.

Random Forest Regressor: Random Forest is a supervised learning algorithm, meaning that the data on which it operates contains labels or outcomes. It works by creating many decision trees, each built on randomly chosen subsets of the data. The model then aggregates the outputs of all of these decision trees to make an overall prediction for unseen data points. In this way, it can process larger datasets and capture more complex associations than individual decision trees. Each tree in the forest builds from a different subset of the data and makes its own independent prediction. The final prediction for input is based on the average or weighted average of all the individual trees' predictions**.**

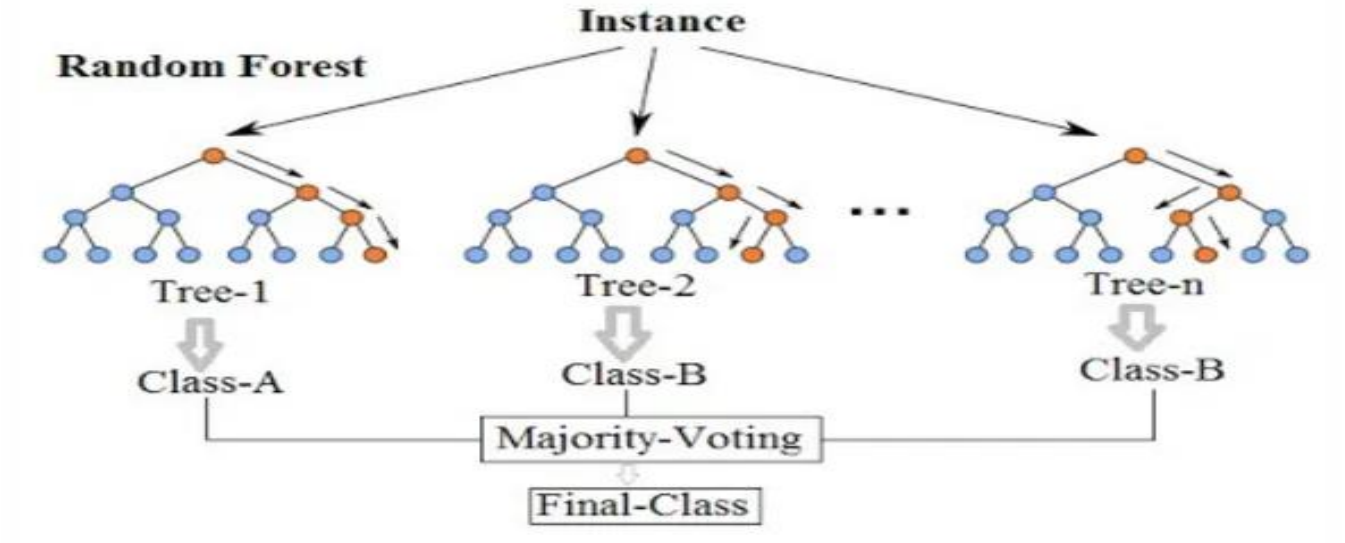


Fig. 7.8 Random Forest Regressor

The individual decision tree models are constructed using a technique called bagging random forest. It involves randomly selecting subsets of the training data and building smaller decision trees from them. After the bagging random forest step, we combine the smaller models to form the random forest model, which outputs a single prediction value. The technique helps reduce variance and improve accuracy by combining the predictions from several decision trees**.** When building a random forest regression model, you need to identify several variables that represent potential features for your dataset. Furthermore, these variables should be related to the outcomes in some form so that they can provide meaningful information about how different features influence the predictions.

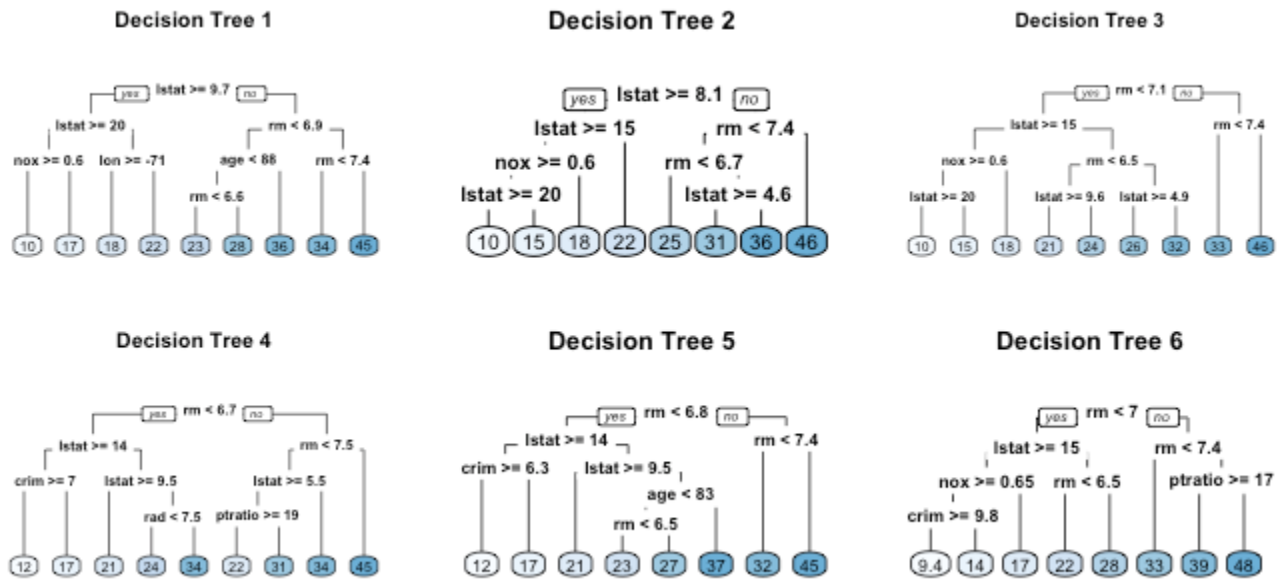


Fig. 7.9 Random Forest regressor mechanism

Features: Random forests are more efficient than other ensemble methods and can process large datasets with fewer parameters making it fast and accurate.

- offer high accuracy when predicting outcomes due to their ability to create highly individualized decision trees
- selects the most important features from a dataset for improved accuracy.
- can be easily automated, making it an ideal choice for efficient and reliable outcomes.
- are resilient to noise in the data, they continue producing accurate results despite corrupted data points.

- reduce the risk of overfitting since they create multiple independent decision trees.
- each tree has its unique attribute, purpose, and variety concerning other trees.
-

Advantages: It is less prone to overfitting than other linear models, such as multiple linear regression or support vector machines. This means that random forests can accurately predict outcomes on unseen data, reducing the risk of errors in predictive modeling.

- Trees are built incrementally and not by a single formula, so they naturally handle non-linearities in the data more effectively than linear models. This makes them better equipped for complex problems that involve multiple variables with varying degrees of importance or interaction.
- Random forest models are also computationally efficient and require fitting fewer parameters compared to other algorithms, such as neural networks or support vector machines. Additionally, the algorithm itself is relatively easy to implement and requires minimal user tuning. This makes it a great tool for quickly building models with good predictive accuracy without spending too much time tweaking parameters.
- They can be used as an ensemble method to create more robust models. Combining multiple trees can reduce bias and variance in your predictions. It is especially beneficial when dealing with high-dimensional data where overfitting may be problematic. Additionally, ensembles can capture interactions between variables that individual trees might miss.

Gradient Boosting Regressor: Gradient Boosting is an ensemble technique. It is primarily used in classification and regression tasks. It provides a forecast model consisting of a collection of weaker prediction models, mostly decision trees. It builds a better model by merging earlier models until the best model reduces the total prediction error. Also referred to as a statistical forecasting model, the main idea of gradient boosting is to attain a model that eliminates the errors of the previous models. Gradient Boosting is named so that the set target outcomes depend on the gradient of the inaccuracy vs the forecast. Every new model created using this method moves closer to the path that lowers prediction error in the range of potential outcomes for every ML training case.

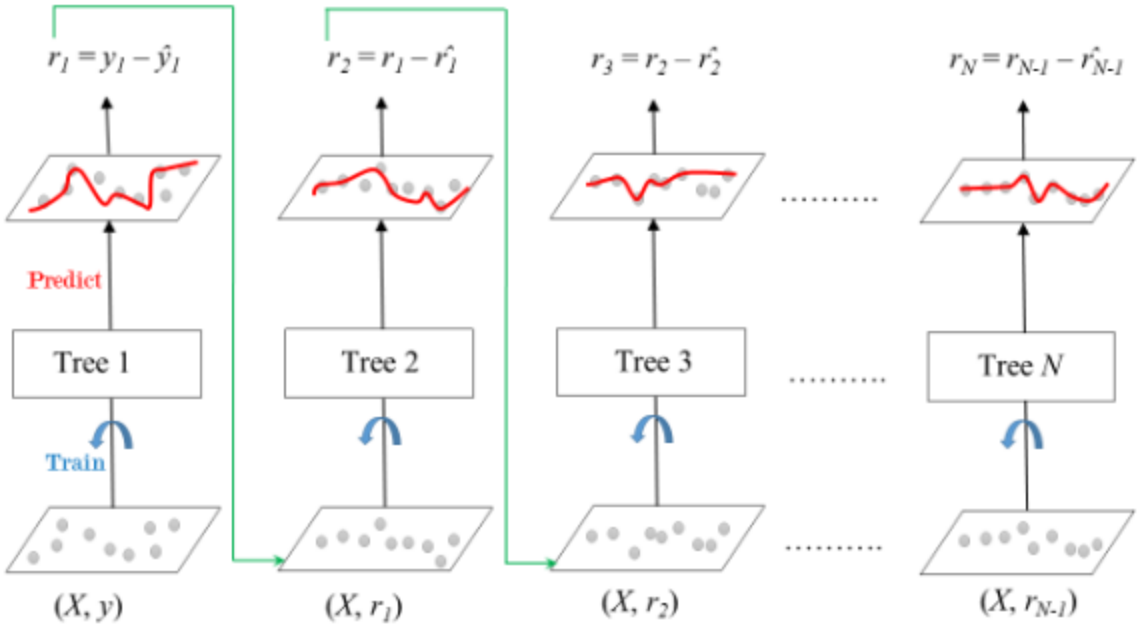


Fig. 7.10 Gradient Boosting Regressor

Gradient Boosting is mainly of two types depending on the target columns:

- **Gradient Boosting Regressor**: It is used when the columns are continuous
- **Gradient Boosting Classifier**: It is used when the target columns are classification problems

The "Loss Function" acts as a distinguisher for them. It is among the three main elements on which gradient boosting works.

- Loss Function: The primary goal in this situation is to maximize the loss function, which is not constant and changes according to the problems. It is simple to create one's own standard loss function, however, it must be differentiable.
- Weak Learners: These are used mainly for predictions. A decision tree is an example of weak learners. For the real output values needed for splits, specific regression trees are applied.
- Additive Model: There are more trees added at once, but no changes are made to the model's already-existing trees. A gradient descent approach reduces the losses when the trees are added.

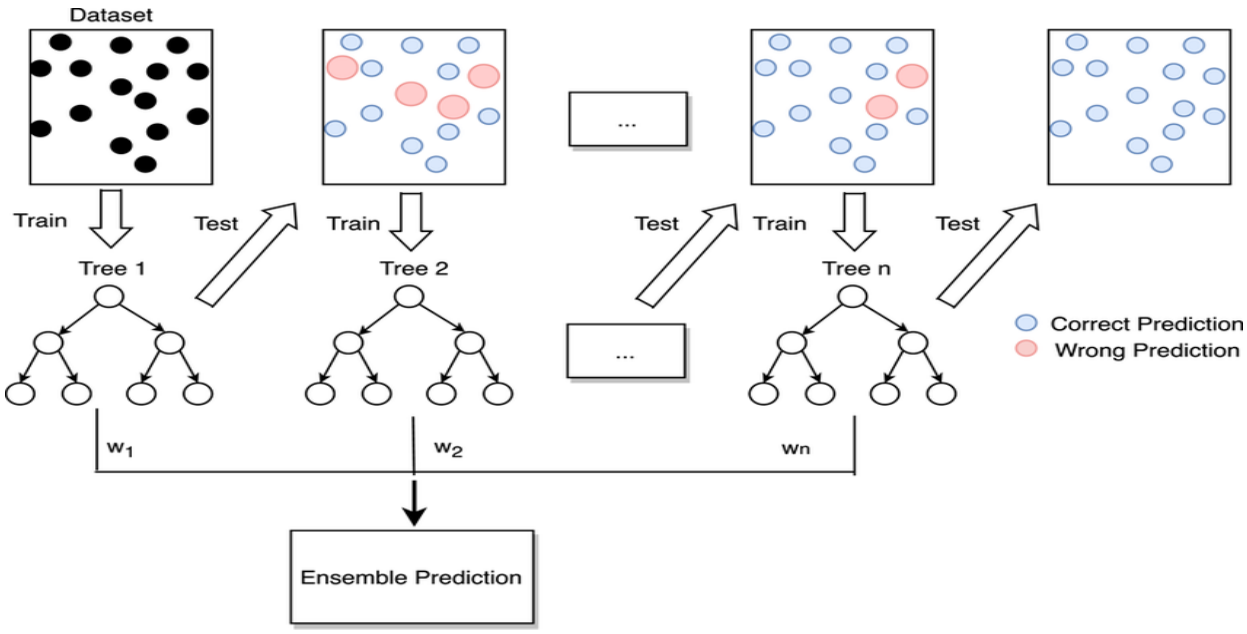


Fig. 7.11 Gradient Boosting Regressor mechanism

Advantages: As the gradient boosting model follows ensemble learning, it is easier to interpret and handle the data.

- It is more accurate than many other algorithms and with cutting-edge methods like bagging, random forest, and decision tree, accurate results are available. Also, one of the best algorithms to handle larger datasets and compute with the weak learners at least loss.
- This algorithm not only supports numerical datasets but is also efficient for categorical data handling.
- is a resilient method that checks over-fitting training datasets quite easily.

Disadvantages: As there can be multiple outliers in a dataset, gradient boosting cannot avoid them completely. As the gradient boosting classifier tends to correct the errors, it accepts the outlying values as well.

- Another disadvantage is the tendency to solve each error of predecessor nodes that results in overfitting of the model. However, this can also be solved with the L1 and L2 regulation method
- Gradient boosting models can be computationally expensive and can take a longer time to train the complete model on CPUs.

Applications: Reducing bias error in an ML model: The gradient boosting algorithm in machine learning is applied to cut down the bias error in the training dataset of the model. Usually, a biased dataset is one in which the linear regression line does not fit the training data with a more significant margin. This inability to capture the true relationship of the training dataset is called bias. And, higher bias denotes underfit training dataset in a machine learning model. For a highly-biased dataset, the gradient boosting algorithm increases the number of stages at a low learning rate to optimize the decision stumps in order to make the training dataset precise.

Resolves problems related to regression: Regression in machine learning is the approach to predicting continuous outcomes by observing the relationship between independent variables or features and dependent variables. Gradient boosting regression algorithm can be used to predict the target variables with the help of decision stumps like AdaBoost.

K Nearest Neighbor Regressor: KNN stands for K Nearest Neighbor. K refers to an undefined number of neighbors we have to find (as the accuracy of the KNN model in machine learning greatly depends on it). In contrast, "Nearest Neighbor," as the name suggests, refers to the closest data points i.e., neighbors from a new point in the data space**.**

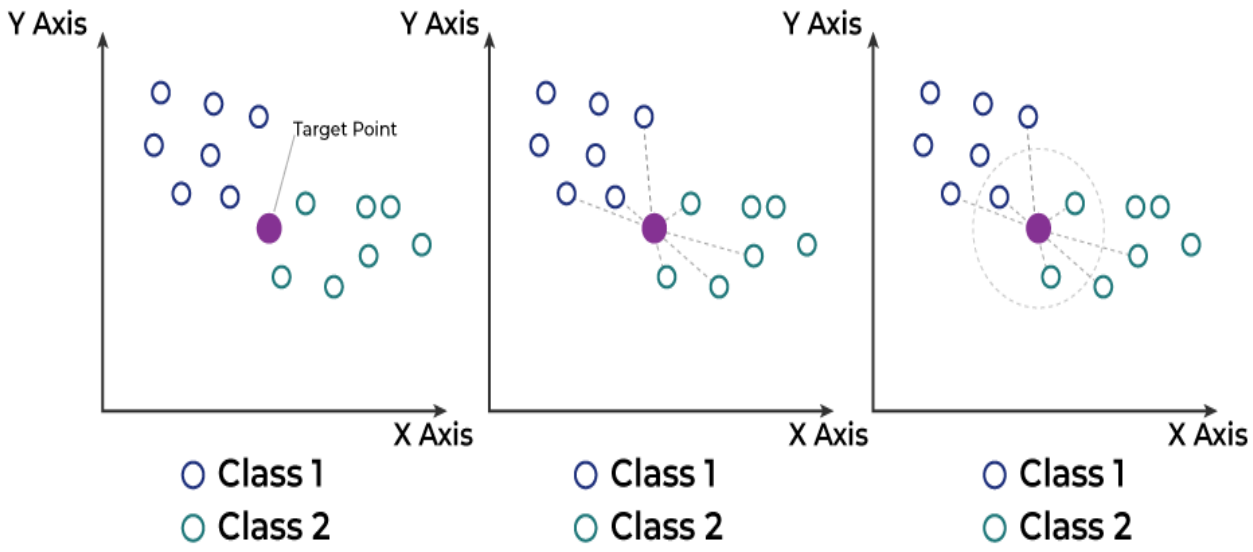
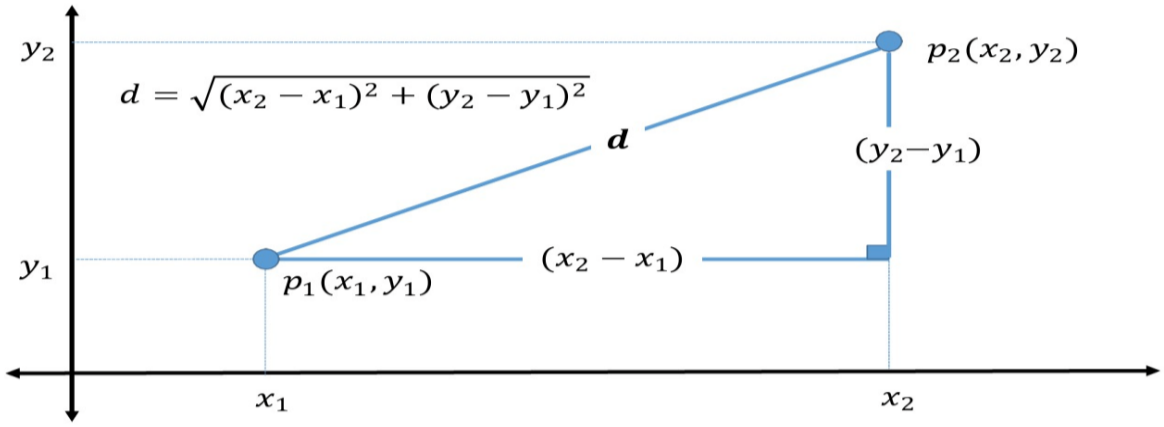


Fig. 7.12 K Nearest Neighbor Classifier

Euclidean Distance: This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object



$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d}(x_j - X_{i_j})^2]}$$

Fig. 7.13 Euclidean Distance

Manhattan Distance: Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d\,(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Fig. 7.14 Manhattan Distance

In statistics, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric supervised learning method used for classification and regression. In both cases, the input consists of the *k* closest training examples in a data set. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors. If *k* = 1, then the output is simply assigned to the value of that single nearest neighbor.

k-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

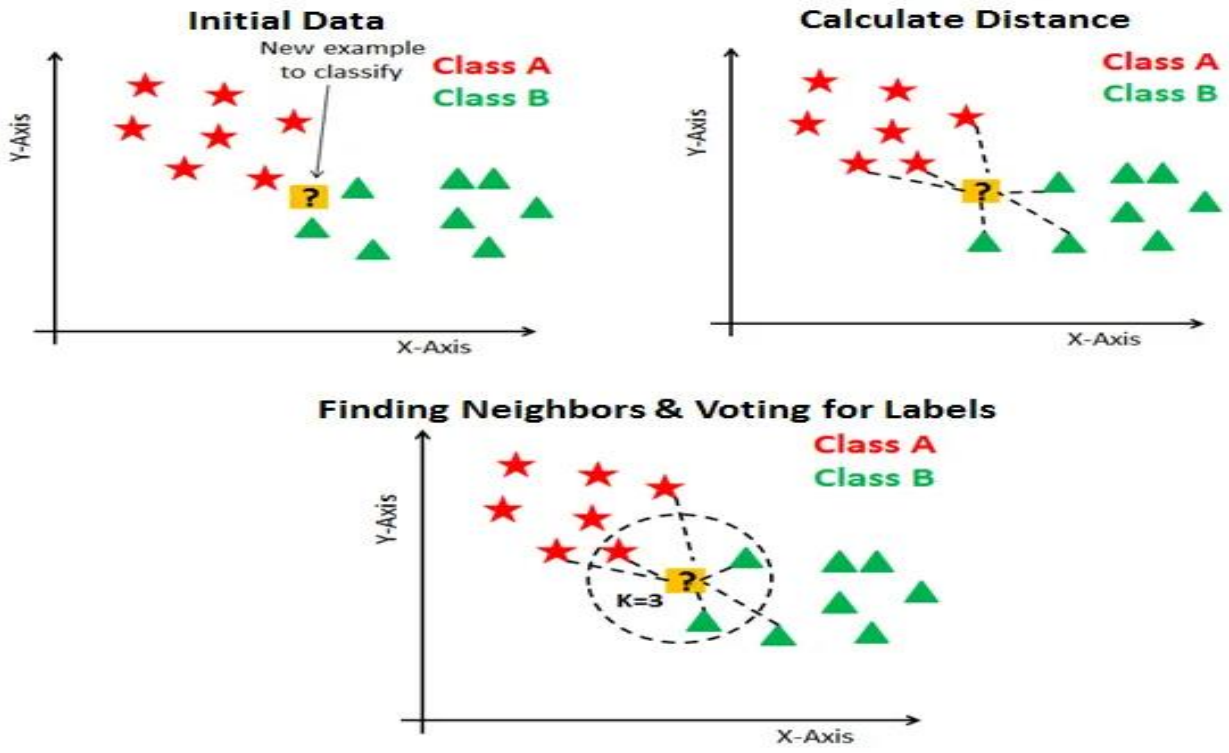


Fig. 7.15 KNN mechanism

Advantages: Being a non-parametric algorithm, KNN doesn't have to worry about assumptions related to the data distribution for it to work correctly.

- can solve both regression and classification problems, works in a supervised learning setup, but other distance-based models can be used for segmentation, and they work in an unsupervised learning setup such as K-means.
- solves multiclass problems naturally because it simply performs majority voting and provides an answer.

- non-linear algorithm and can solve complex problems that traditional statistical algorithms like logistic regression may not easily solve.
- simple to comprehend and implement.

Disadvantages: As KNN uses a distance metric to calculate proximity, it is susceptible to outliers as they can significantly skew the results.

- In testing phase, KNN has to calculate the distance from each testing data point to all other training data points this causes KNN to become slow in the testing phase
- Majority voting is a simplistic process, all observations are treated equally, even though the distance of each observation from the new data point might differ.

DATA SPLITTING

Feature scaling is a preprocessing technique used to standardize the range of independent features or variables in the data. It ensures that all features have the same scale, preventing certain features from dominating the others during model training. Common methods for feature scaling include Min-Max scaling and Standardization (Z-score normalization). Features like Year, HP, Cylinders, Highway MPG, City MPG, and Popularity were scaled using Standard Scaler function to ensure that they have comparable magnitudes.

```
# Defining the x and y data
x = df[['Popularity','Year','HP','Cylinders','Highway MPG','City MPG']].values
y = df['Price'].values

print(x.shape)
print(y.shape)
```

```
(9784, 6)
(9784,)
```

```
# Scaling and transforming the x any y data
sc_x = StandardScaler()
sc_y = StandardScaler()

x = sc_x.fit_transform(x)
y = sc_y.fit_transform(y.reshape(-1,1))
```

Fig. 8.1 Data Scaling and Splitting

The following data was split into train and test set with a proportion of 80% train set and 20% test set.

```
# Splitting the train and test data
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(7827, 6)
(7827, 1)
(1957, 6)
(1957, 1)
```

Fig. 8.2 Train and Test data

MODEL TRAINING

```
# Define models to iterate over
models = {
    "Linear Regression": LinearRegression(),
    "Ridge Regression": Ridge(),
    "Lasso Regression": Lasso(),
    "Decision Tree Regressor": DecisionTreeRegressor(),
    "Random Forest Regressor": RandomForestRegressor(),
    "Gradient Boosting Regressor": GradientBoostingRegressor(),
    "K Neighbor Regressor": KNeighborsRegressor()
}

# Results table
results = []

# Iterate over models
for name, model in models.items():
    # Fit model
    model.fit(x_train, y_train)

    # Make predictions
    y_pred = model.predict(x_test)

    # Calculate evaluation metrics
    accuracy = model.score(x_test, y_test)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)


    # Print or store the evaluation metrics
    print(f"Evaluation Metrics:")
    print(f"Algorithm: {name}")
    print(f"Accuracy: {accuracy}")
    print(f"Mean Absolute Error: {mae}")
    print(f"Mean Squared Error: {mse}")
    print()


    # Append results to table
    results.append([name, accuracy, mae, mse])

    # Plot regression
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, color='blue')
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
    plt.xlabel('Actual Price')
    plt.ylabel('Predicted Price')
    plt.title(f'{name} - Regression Plot')
    plt.show()
```

Fig. 9.1 Model Evaluation function

1. **Linear Regression**: Linear regression is a simple and commonly used supervised learning algorithm used for regression tasks. It assumes a linear relationship between the independent variables (features) and the dependent variable (target). In the context of automobile price prediction, linear regression models the relationship

between the features (such as Year, HP, Highway MPG, City MPG, Popularity, etc.) and the target variable (Price). The algorithm fits a linear regression line to the data, minimizing the sum of squared errors between the actual and predicted values. The regression curve generated by linear regression is a straight line in the multidimensional feature space.

2. **Ridge Regression**: Ridge regression is a regularized version of linear regression that adds a penalty term to the linear regression cost function. The penalty term helps to reduce overfitting by penalizing large coefficient values. It is particularly useful when the dataset has multicollinearity (high correlation between features). Ridge regression tends to shrink the coefficients towards zero, leading to a smoother regression curve compared to linear regression.
3. **Random Forest Regressor**: Random Forest Regressor is an ensemble learning technique based on decision trees. It builds multiple decision trees during training and aggregates their predictions to make the final prediction. Random Forest Regressor is robust to overfitting and works well with both categorical and numerical features. It can capture complex nonlinear relationships between the features and the target variable, resulting in a more flexible regression curve.
4. **Decision Tree Regressor**: Decision Tree Regressor is a non-parametric supervised learning algorithm used for regression tasks. It partitions the feature space into regions based on feature values and predicts the average target value within each region. Decision trees can capture complex interactions between features but are prone to overfitting, especially with deep trees. The regression curve generated by decision tree regression consists of step-like segments corresponding to the partitioned feature space.
5. **Gradient Boosting Regressor**: Gradient Boosting Regressor is an ensemble learning technique that builds decision trees sequentially, each one correcting the errors of its predecessor. It combines multiple weak learners (typically shallow decision trees) to create a strong learner. Gradient Boosting Regressor minimizes the loss function using gradient descent optimization. It is robust to overfitting and can capture complex nonlinear relationships, resulting in a flexible regression curve.
6. **K Nearest Neighbor Regressor**: K Nearest Neighbor (KNN) Regressor is a non-parametric regression algorithm that predicts the target variable by averaging the values of its k nearest neighbors in the feature space. KNN Regressor does not make any assumptions about the underlying data distribution. It is sensitive to the choice of the distance metric and the value of k. The regression curve produced by KNN Regressor is non-linear and depends on the distribution of the training data in the feature space.

MODEL EVALUATION

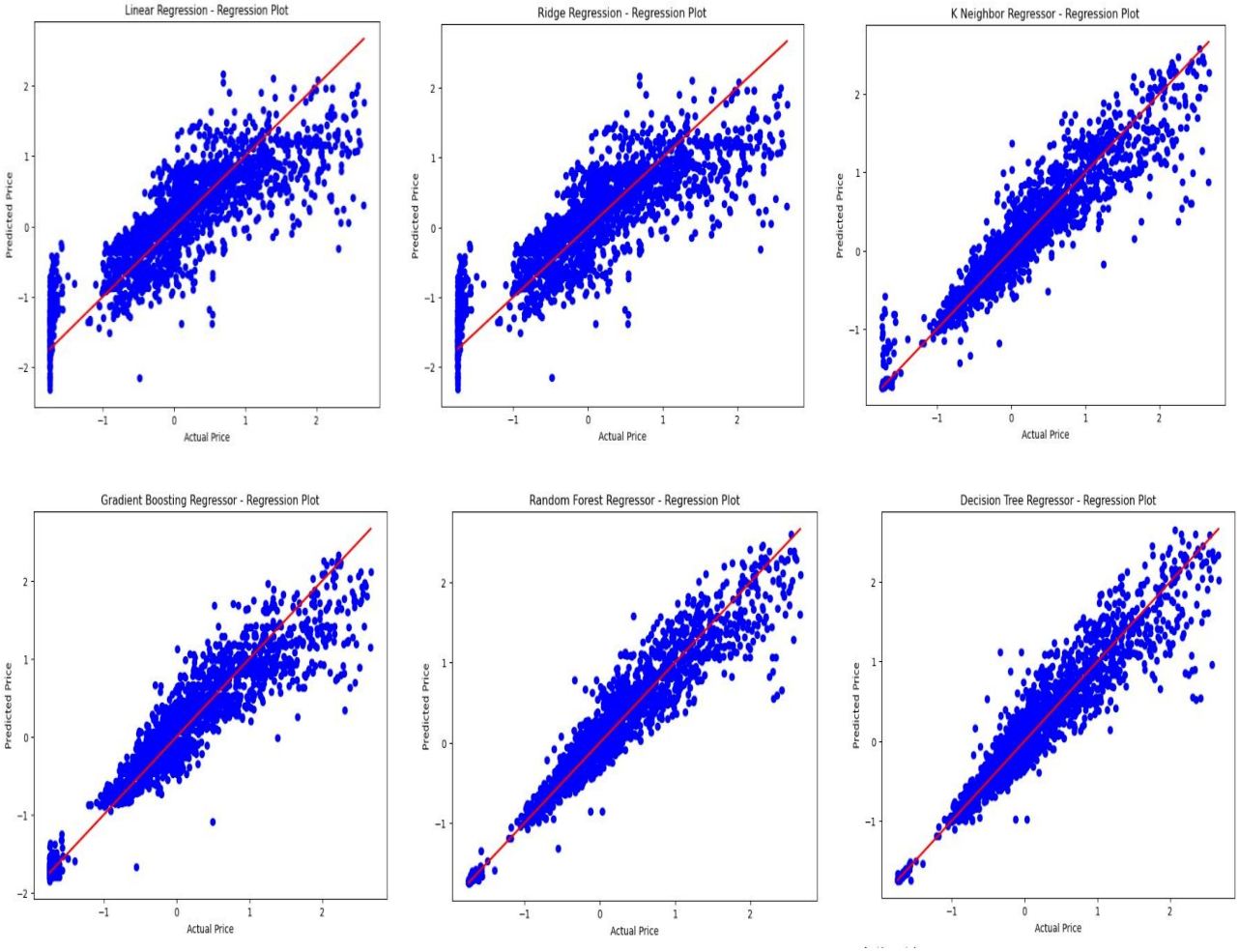


Fig. 10.1 Regression plot of algorithms

- Linear regression achieves a moderate level of accuracy in predicting automobile prices. The MAE and MSE values indicate that the model's predictions deviate from the actual prices by an average of 0.41 and 0.28 units, respectively.
- Decision tree regression demonstrates high accuracy (0.92) and low MAE and MSE values (0.19 and 0.08, respectively). It excels in capturing nonlinear relationships within the data, leading to more accurate price predictions.
- Random forest regression performs slightly better than the decision tree model, with a higher accuracy of 0.93 and lower MAE and MSE values (0.18 and 0.07,

respectively). Ensemble learning helps reduce overfitting and improves prediction accuracy.

- Gradient boosting regression achieves an accuracy of 0.89, with moderate MAE and MSE values (0.24 and 0.11, respectively). It sequentially improves upon the weaknesses of previous models, resulting in accurate predictions.
- KNN regression delivers competitive performance, with an accuracy of 0.90 and relatively low MAE and MSE values (0.20 and 0.10, respectively). It relies on the similarity of neighboring data points to make predictions, making it effective for local pattern recognition.

Model	Accuracy	Mean Square Error	Mean Absolute Error
Linear Regression	0.72	0.41	0.28
Ridge Regression	0.72	0.41	0.28
Lasso Regression	-0.01	0.77	1.0
Decision Tree Regressor	0.92	0.19	0.08
Random Forest Regressor	0.93	0.18	0.07
Gradient Boosting Regressor	0.89	0.24	0.11
K Nearest Neighbor	0.90	0.20	0.10

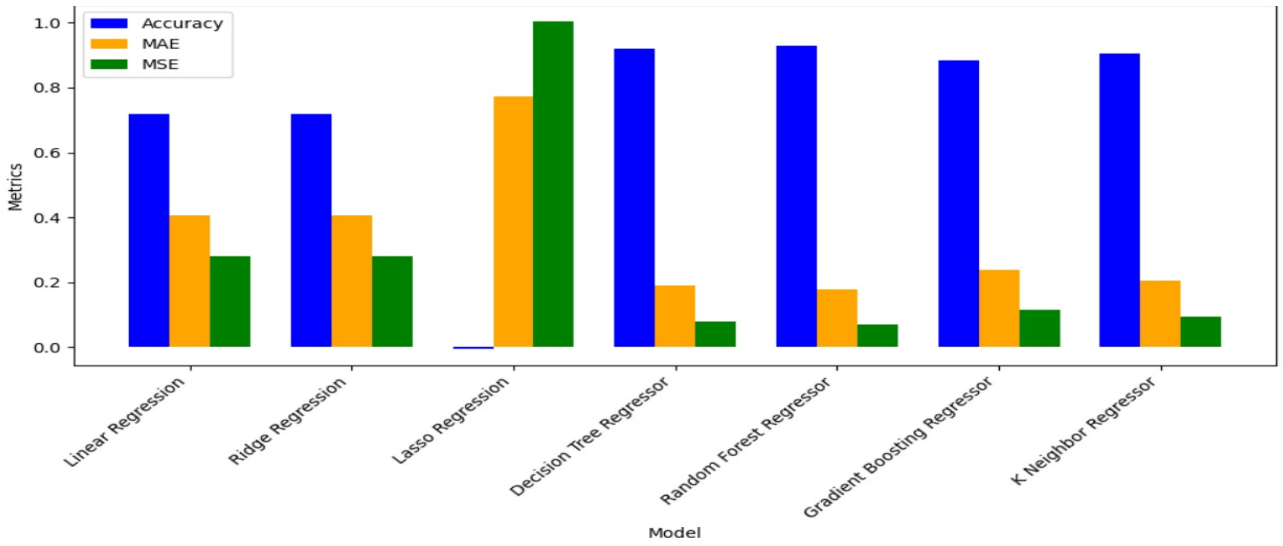


Fig. 10.2 Result of Models

CONTINUAL IMPROVEMENT

Continual improvement is a fundamental aspect of deploying machine learning models in real-world scenarios. As organizations strive to leverage data-driven insights for decision-making, the pursuit of ongoing refinement and optimization becomes imperative.

Continual improvement involves an iterative process of refining and enhancing machine learning models based on feedback, new data, and evolving business requirements. It encompasses various stages, including model training, evaluation, deployment, and monitoring. Establishing robust feedback loops is essential for capturing insights from model performance in production environments. By analyzing prediction errors, user feedback, and business outcomes, organizations can identify areas for improvement and iterate on model parameters, features, and algorithms accordingly.

It also entails ensuring data quality, integrity, and governance throughout the model lifecycle. Addressing issues such as missing values, data drift, and bias requires proactive measures such as data validation, cleansing, and monitoring to maintain model accuracy and reliability. Techniques such as online learning, transfer learning, and reinforcement learning enable models to incorporate fresh insights and adapt their predictions over time, leading to more accurate and relevant outputs. Real-world scenarios present diverse challenges and complexities that necessitate tailored optimization strategies. In dynamic market environments, pricing models may require frequent recalibration to reflect changing consumer preferences, competitor actions, and economic trends.

Continual improvement extends beyond technical optimization to encompass ethical and responsible AI practices. Organizations must prioritize transparency, fairness, and accountability in their model development and deployment processes, mitigating risks associated with unintended biases, privacy violations, and algorithmic discrimination. Successful continual improvement initiatives rely on cross-functional collaboration and alignment across business, data science, engineering, and governance teams.

In conclusion, continual improvement in machine learning model deployment is a dynamic and ongoing endeavor that requires a holistic approach encompassing technical innovation, data governance, ethical considerations, and organizational collaboration.

CONCLUSION

The performance evaluation of various machine learning algorithms for automobile price prediction reveals valuable insights into their effectiveness and suitability for real-world applications. Among the models tested, decision tree, random forest, and KNN regressors demonstrate superior accuracy and lower error metrics compared to linear regression, ridge regression, lasso regression, and gradient boosting regressor. These algorithms leverage different techniques such as ensemble learning and local pattern recognition to achieve more accurate predictions.

Decision tree and random forest regressors emerge as top performers, offering high accuracy and robustness in predicting automobile prices. KNN regression also proves to be competitive, especially for datasets with local patterns. Decision tree-based models offer transparency and interpretability, allowing stakeholders to understand the underlying decision-making process. This attribute is crucial for building trust and confidence in the model's predictions, particularly in industries with stringent regulatory requirements.

While the evaluated models show promise in controlled environments, deploying them in real-world scenarios presents challenges. Factors such as data quality, feature engineering, and model interpretability need careful consideration. Additionally, ongoing model monitoring and updates are essential to ensure optimal performance and relevance over time. As the volume and complexity of data increase, scalability and computational efficiency become critical factors. Random forest and gradient boosting regressors, while effective, may pose scalability challenges due to their computational overhead and resource requirements.

Further research and experimentation are warranted to explore advanced techniques such as neural networks and deep learning for automobile price prediction. Additionally, integrating external data sources and refining feature selection processes could enhance model performance and generalization capabilities.

REFERENCE

- Rochester Institute of Technology Rochester Institute of Technology RIT Digital Institutional Repository RIT Digital Institutional Repository Theses 12-2021 Used Cars Price Prediction and Valuation using Data Mining Techniques
- BIELSKI, V., & RAMARATHNAM, S. (2020, July 16). UAE's used car sales set to surge past 1 million mark by 2025. Retrieved from gulfbusiness: https://gulfbusiness.com/uaes-usedcar-sales-set-surge-past-1-million
- Bridge, S. (2020, January 10). Why the value of used cars is rising for the first time in the UAE. Retrieved from arabianbusiness: https://www.arabianbusiness.com/retail/435520-why the-value-of-used-cars-is-rising-for-the-first-time-in-the-uae
- Ceriottia, M. (2019). Unsupervised machine learning in atomistic simulations, between predictions and understanding. 150-155.
- Gegic, E., Isakovic, B., Keco, D., Masetic, Z., & Kevric, J. (2019, February). Car Price Prediction using Machine. TEM Journal, 8(1), 113-118. doi:10.18421/TEM81-16
- Gongqi, S., Yansong, W., & Qiang, Z. (2011). A New Model for Residual Value Prediction of the Used Car Based on BP Neural. Third International Conference on Measuring Technology and Mechatronics Automation (pp. 682-685). Shanghai: IEEE. doi:10.1109/ICMTMA.2011.455