TRAINING REPORT

OF

SUMMER TRAINING, UNDERTAKEN

AT

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY, ROPAR

ON

"CAR PRICE PREDICTION USING MACHINE LEARNING"

SUBMITTED IN THE PARTIAL FULFILLMENT OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY: ARSHDEEP KAUR

ROLL NO.:12101126

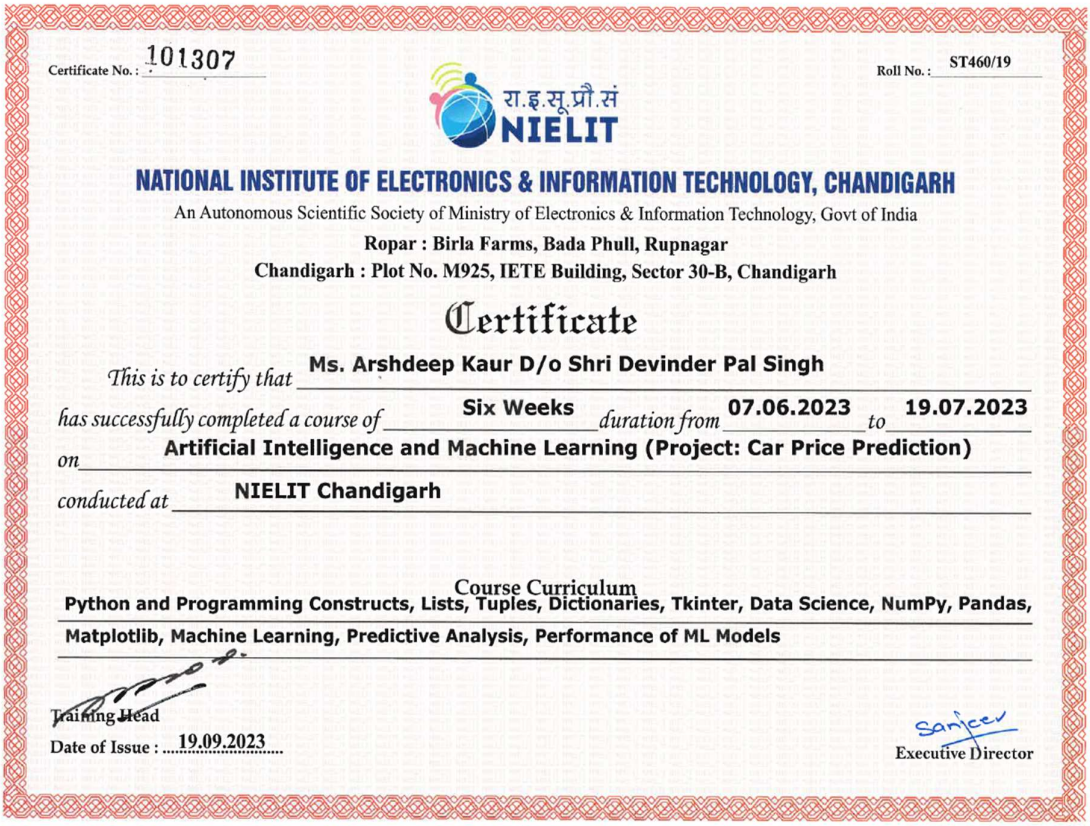
BATCH:2021-2025

DEPARTMENT OF COMPUTER SCIENCE &

ENGINEERING PUNJABI UNIIVERSITY

PATIALA-147002

CERTIFICATE BY THE INSTITUTE



Certificate No. : 101307

Roll No. : ST460/19

रा.इ.सू.प्रौ.सं
NIELIT

NATIONAL INSTITUTE OF ELECTRONICS & INFORMATION TECHNOLOGY, CHANDIGARH

An Autonomous Scientific Society of Ministry of Electronics & Information Technology, Govt of India

Ropar : Birla Farms, Bada Phull, Rupnagar

Chandigarh : Plot No. M925, IETE Building, Sector 30-B, Chandigarh

Certificate

This is to certify that **Ms. Arshdeep Kaur D/o Shri Devinder Pal Singh**

has successfully completed a course of **Six Weeks** *duration from* **07.06.2023** *to* **19.07.2023**

on **Artificial Intelligence and Machine Learning (Project: Car Price Prediction)**

conducted at **NIELIT Chandigarh**

Course Curriculum

Python and Programming Constructs, Lists, Tuples, Dictionaries, Tkinter, Data Science, NumPy, Pandas, Matplotlib, Machine Learning, Predictive Analysis, Performance of ML Models

Training Head

Date of Issue : 19.09.2023

Sanjeev

Executive Director

DEPARTMENT OF COMPUTER

SCIENCE AND ENGINEERING PUNJABI

UNIVERSITY, PATIALA

DECLARATION

I, **ARSHDEEP KAUR** hereby declare that I have undertaken summer training at **"NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY, ROPAR" (NIELIT)** during a period from **7th June,2023 to 19th July,2023** in partial fulfilment of requirements for the degree of B. Tech (Department of Computer Science & Engineering) at Punjabi University, Patiala. The work which is being presented in the training report submitted to department of computer science & engineering at Punjabi University, Patiala is an authentic record of training work.

Signature Of Student

The summer training viva-voice examination of _____ has been held on _____ and accepted.

Signature Of Examiner

ABSTRACT

In recent years, the automotive industry has witnessed a surge in data-driven approaches to optimize various aspects of the business. One prominent application is the prediction of car prices using machine learning algorithms. This research aims to develop a robust and accurate car price prediction model that can assist buyers, sellers, and other stakeholders in making informed decisions.

The proposed machine learning model leverages a comprehensive dataset comprising various features such as car brand, model, year, kilometres driven, fuel type, seller type, owner, transmission, selling price. The dataset is pre-processed to handle missing values, normalize numerical attributes, and encode categorical variables to make it suitable for training the model.

A hybrid ensemble of machine learning algorithms, including Random Forest, Linear Regression& Logistic Regression is employed to capture complex relationships between car attributes and their corresponding prices.

The evaluation of the model is conducted on a diverse set of car data, representing a wide range of brands, models, and price ranges. Performance metrics such as R-squared are utilized to assess the model's accuracy and robustness.

The developed car price prediction model can serve as a valuable tool for prospective car buyers to estimate fair prices and make cost-effective decisions.

As car prices continue to fluctuate with changing market dynamics, the proposed model can be continuously updated and fine-tuned to adapt to emerging trends, ensuring its relevance and applicability in the ever-evolving automotive landscape.

ACKNOWEDGEMENT

I, Arshdeep Kaur, express my heartfelt appreciation to all the individuals who have contributed to the successful completion of my summer training and the preparation of this report.

First and foremost, I am deeply grateful to the Head of Department, **Dr. HIMANSHU AGGARWAL,** for providing me with the opportunity to participate in this training. Their encouragement and belief in my abilities have been a source of inspiration throughout this journey.

I extend my gratitude to **Dr. SARWAN SINGH** (Joint Director at NIELIT) for their unwavering support, mentorship, and guidance. Their valuable insights and constructive feedback throughout the training period have been instrumental in my personal and professional growth.

I would also like to extend my sincere thanks to the Training Coordinator, **Dr. NIRVAIR NEERU**, for their efforts in coordinating the training program and ensuring a conducive learning environment. Their dedication and assistance have been crucial in making this experience smooth and rewarding.

I am indebted to my family and friends for their continuous encouragement, understanding, and motivation during this training period. Their support has been the driving force behind my dedication to excel in every aspect of this endeavour.

In conclusion, this summer training has been a transformative experience for me, and I am grateful to each and every individual who has contributed to my growth and learning.

ABOUT THE INSTTIUTE

National Institute of Electronics & Information Technology (NIELIT), (erstwhile DOEACC Society), an Autonomous Scientific Society under the administrative control of Ministry of Electronics & Information Technology (MoE&IT), Government of India, was set up to carry out Human Resource Development and related activities in the area of Information, Electronics & Communications Technology (IECT). NIELIT is engaged both in Formal & Non-Formal Education in the area of IECT besides development of industry-oriented quality education and training programmes in the state-of-the-art areas. NIELIT has endeavoured to establish standards to be the country's premier institution for Examination and Certification in the field of IECT. It is also one of the National Examination Body, which accredits institutes/organizations for conducting courses in IT in the non-formal sector.

As on date, NIELIT has forty seven (47) centres located at Agartala, Aizawl, Ajmer, Alawalpur (Saksharta Kendra), Aurangabad, Bhubaneswar, Calicut, Chandigarh, Chennai, Chuchuyimlang, Chur Chandpur, Daman, Delhi, Dibrugarh, Dimapur, Gangtok, Gorakhpur, Guwahati, Haridwar, Imphal, Itanagar, Jammu, Jorhat, Kargil, Kohima, Kolkata, Kokrajhar, Kurukshetra, Lakhanpur (Saksharta Kendra), Leh, Lucknow, Lunglei, Majuli, Mandi, Pasighat, Patna, Pali, Ranchi, Ropar, Senapati, Shillong, Shimla, Silchar, Srinagar, Tezpur, Tura and Tezu with its Headquarters at New Delhi. It is also well networked throughout India with the presence of about 700 + institutes.

Over the last two decades, NIELIT has acquired very good expertise in IT training, through its wide repertoire of causes, ranging from 'O' Level (Foundation), 'A' Level (Advance Diploma), 'B' Level (MCA equivalent), 'C' Level (M-Tech level), IT literacy courses such as CCC (Course on Computer Concept), BCC (Basic Computer Course) and other such long term and short term course in the non-formal sector like courses on Information Security, ITeS-BPO(Customer Care/Banking), Computer Hardware Maintenance (CHM-O/A level), Bio-Informatics(BI-O/A/B level), ESDM etc, besides, high end courses offered by NIELIT Centres at Post-Graduate level (M.Tech) in Electronics Design & Technology, Embedded Systems etc. which are not normally offered by Universities/Institutions in the formal sector, in association with the respective state Universities.

The basket of activities of NIELIT is further augmented by the wide range of projects that it undertakes. NIELIT has demonstrated its capability and capacity to undertake R&D projects, consultancy services, turnkey projects in office automation, software development, website development etc. NIELIT is also the nodal implementing agency on behalf of MeitY for Data Digitization of the population of 15 assigned States and 2 Union Territories for the creation of National Population Register (NPR) project of Registrar General of India (RGI).

NIELIT is also successfully executing the Agriculture Census and Input Survey project under which tabulation of about 10 crore data records have to be done. NIELIT has planned a roadmap for adopting appropriate pedagogy for metamorphosing NIELIT into an Institute of National Importance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF CODES

LIST OF GRAPHS

CHAPTER-1

INTRODUCTION

1.1. TECHNOLOGIES USED

1.1.1. PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured ,object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

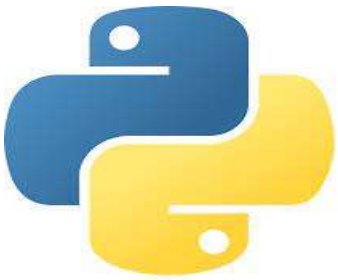


Figure-1.1
PYTHON LOGO

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular programming languages. Python users are colloquially called Pythonistas.

1.1.1.1. HISTORY OF PYTHON

Python was conceived in the late 1980s by **Guido van Rossum** at **Centrum Wiskunde & Informatica (CWI) in the Netherlands** as a successor to the ABC programming language, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In

Figure-1.2
GUIDO VAN ROSSAM (Creator of Python)

January 2019, active Python core developers elected a five-member Steering Council to lead the project.

Python 2.0 was released on 16 October 2000, with many major new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

Python 2.7's end-of-life was initially set for 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No further security patches or other improvements will be released for it. Currently only 3.7 and later are supported. In 2021, Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues leading to possible remote code execution and web cache poisoning.

In 2022, Python 3.10.4 and 3.9.12 were expedited and 3.8.13, and 3.7.13, because of many security issues. When Python 3.9.13 was released in May 2022, it was announced that the 3.9 series (joining the older series 3.8 and 3.7) would only receive security fixes in the future. On September 7, 2022, four new releases were made due to a potential denial-of-service attack: 3.10.7, 3.9.14, 3.8.14, and 3.7.14.

As of November 2022, Python 3.11 is the stable release. Notable changes from 3.10 include increased program execution speed and improved error reporting.

1.1.1.2. WHY THE NAME PYTHON?

There is a fact behind choosing the name python. **GUIDO VAN ROSSUM** was reading the script of a popular bbc comedy series "**Monty Python's Flying Circus**". It was late on-air 1970s.

Van Rossum wanted to select a name which unique, sort, and little-bit mysterious. So, he decided to select naming Python after the **"Monty Python's Flying Circus"** for their newly created programming language.

Python is also versatile and widely used in every technical field, such as Machine Learning, Artificial Intelligence, Web Development, Desktop Application, etc.

1.1.1.3. FEATURES OF PYTHON

Python provides many useful features which make it popular and valuable from the other programming languages.

- EASY TO LEARN AND USE
- INTERPRETED LANGUAGE
- FREE AND OPEN SOURCE
- OBJECT-ORIENTED LANGUAGE
- EXTENSIBLE
- LARGE STANDARD LIBRARY
- GUI PROGRAMMING SUPPORT
- INTEGRATED
- EMBEDDABLE
- DYNAMIC MEMORY ALLOCATION

1.1.1.4. PYTHON LIBRARIES

Python libraries are a collection of helpful functions that allow us to write code without having to start from scratch. With more than 137,000 libraries, Python can be used to create applications and models in a variety of fields, for instance, machine learning, data science, data visualization, image and data manipulation, and many more. Some of the known libraries are:

1. PANDAS

Pandas is a BSD (Berkeley Software Distribution) licensed open-source library. This popular library is widely used in the field of data science. They are primarily used for data analysis, manipulation, cleaning, etc.

Figure-3 PANDAS

2. NUMPY

NumPy is one of the most widely used open-source Python libraries, focusing on scientific computation. It features built-in mathematical functions for quick computation and supports big matrices and multidimensional data. "Numerical Python" is defined by the term "NumPy."

Figure-1.4 NUMPY

3. MATPLOTLIB

Matplotlib is a cross-platform, data visualization and graphical plotting library (histograms, scatter plots, bar charts, etc) for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

Figure-1.5
MATPLOTLIB

4. SCI-KIT LEARN

Figure-1.6
SCI-KIT LEARN

Scikit Learn is an open-source library for machine learning algorithms that runs on the Python environment. It can be used with both supervised and unsupervised learning algorithms. The library includes popular algorithms as well as the NumPy, Matplotlib, and SciPy packages. Scikit learns most well-known use is for music suggestions in Spotify.

5. SEABORN

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on top matplotlib library and is also closely integrated with the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data.

Figure-1.7
SEABORN

1.1.2. HTML

The **Hypertext Markup Language** is the standard markup language for documents designed to be displayed in a web browser. It defines the meaning and structure of web content. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Figure-1.8
HTML LOGO

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for its appearance.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting

structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <**img**> and <**input**> directly introduce content into the page. Other tags such as <**p**> and </**p**> surround and provide information about document text and may include sub-element tags. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the <**canvas**> element, together with JavaScript.

1.1.2.1. HISTORY OF HTML

The first version of HTML was written by Tim Berners-Lee in 1993. Since then, there have been many different versions of HTML. The most widely used version throughout the 2000's was **HTML 4.01**, which became an official standard in December 1999. Another version, **XHTML**, was a rewrite of HTML as an XML language. XML is a standard markup language that is used to create other markup languages. Hundreds of XML languages are in use today, including GML (Geography Markup Language), MathML, MusicML, and RSS (Really Simple Syndication). Since each of these languages was written in a common language (XML), their content can easily be shared across applications. This makes XML potentially very powerful, and it's no surprise that the W3C would create an XML version of HTML (again, called XHTML). XHTML became an official standard in 2000, and was updated in 2002. XHTML is very similar to HTML, but has stricter rules. Strict rules are necessary for all XML languages, because without it, interoperability between applications would be impossible. You'll learn more about the differences between HTML and XHTML.

Figure-1.9 TIM BERNERS-LEE

Most pages on the Web today were built using either HTML 4.01 or XHTML 1.0. However, in recent years, the W3C (in collaboration with another organization, the WHATWG), has been working on a brand-new version of HTML, **HTML5**. In 2011, HTML5 is still a draft

specification, and is not yet an official standard. However, it is already widely supported by browsers and other web-enabled devices, and is the way of the future.

1.1.2.2. FEATURES OF HTML

1.) It is easy to learn and easy to use.

2.) It is platform-independent.

3.) Images, videos, and audio can be added to a web page.

4.) Hypertext can be added to the text.

5.) It is a markup language.

1.1.3. FLASK

Flask is a micro web framework written in Python. It is classified as a micro web framework because it doesn't require tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Applications that use the Flask micro web framework include **Pinterest** and **LinkedIn.**

Figure-1.10
FLASK LOGO

1.1.3.1. HISTORY OF FLASK

Flask was created by **Armin Ronacher** of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

Figure-1.11
ARMIN RONACHER

When Ronacher and Georg Brandl created a bulletin board system written in Python in 2004, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallet's project. Since 2018, Flask-related data and objects can be rendered with Bootstrap.

Flask has become popular among Python enthusiasts. As of October 2020, it has the second-most number of stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018, 2019, 2020 and 2021.

1.1.3.2. FEATURES OF FLASK

- ❖ FLASK PROVIDES A DEVELOPMENT SERVER AND A DEBUGGER.
- ❖ IT USES JINJA2 TEMPLATES.
- ❖ IT IS COMPLIANT WITH WSGI 1.0.
- ❖ IT PROVIDES INTEGRATED SUPPORT FOR UNIT TESTING.
- ❖ MANY EXTENSIONS ARE AVAILABLE FOR FLASK, WHICH CAN BE USED TO ENHANCE ITS FUNCTIONALITIES.

1.1.4. CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Figure-1.12 CSS LOGO

CSS is designed to enable the separation of content and presentation, including layout, colours, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .CSS file, which reduces complexity and repetition in the structural content; and enable the .CSS file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/CSS is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

1.1.4.1. HISTORY OF CSS

CSS was first proposed by Håkon Wium Lie on 10 October 1994. At the time, Lie was working with Tim Berners-Lee at CERN. Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside World Wide Web Consortium resulted in the first W3C CSS Recommendation (CSS1) being released in 1996. In particular, a proposal by Bert Bos was influential; he became co-author of CSS1, and is regarded as co-creator of CSS.

Figure-1.13 HAKON VIUM LIE

Style sheets have existed in one form or another since the beginnings of Standard Generalized Markup Language (SGML) in the 1980s, and CSS was developed to provide style sheets for the web. One requirement for a web style sheet language was for style sheets to come from different sources on the web. Therefore, existing style sheet languages like DSSSL and FOSI were not suitable. CSS, on the other hand, let a document's style be influenced by multiple style sheets by way of "cascading" styles.

As HTML grew, it came to encompass a wider variety of stylistic capabilities to meet the demands of web developers. This evolution gave the designer more control over site appearance, at the cost of more complex HTML. Variations in web browser implementations, such as Viola WWW and Worldwide Web, made consistent site appearance difficult, and users had less control over how web content was displayed. The browser/editor developed by Tim Berners-Lee had style sheets that were hard-coded into the program. The style sheets could therefore not be linked to documents on the web. Robert Cailliau, also of CERN, wanted to separate the structure from the presentation so that different style sheets could describe different presentation for printing, screen-based presentations, and editors.

1.1.4.2. FEATURES OF CSS

- Opportunity in Web designing
- Website Design
- Web Control

1.2. MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

IBM has a rich history with machine learning. One of its own, Arthur Samuel, is credited for coining the term, “machine learning” with his research (PDF, 481 KB) (link resides outside IBM) around the game of checkers. Robert Nealey, the self-proclaimed checkers master, played the game on an IBM 7094 computer in 1962, and he lost to the computer. Compared to what can be done today, this feat seems trivial, but it’s considered a major milestone in the field of artificial intelligence.

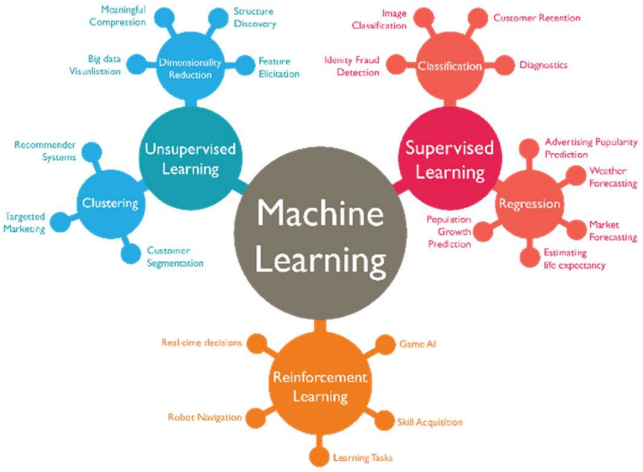


Figure-1.14
MACHINE LEARNING

Over the last couple of decades, the technological advances in storage and processing power have enabled some innovative products based on machine learning, such as Netflix’s recommendation engine and self-driving cars.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

Machine learning algorithms are typically created using frameworks that accelerate solution development, such as TensorFlow and PyTorch.

1.2.1. HISTORY OF MACHINE LEARNING

The term machine learning was coined in 1959 by **Arthur Samuel**, an IBM employee and pioneer in the field of computer gaming and artificial intelligence. The synonym self-teaching computers was also used in this time period.

Figure-1.15 ARTHUR SAMUEL

By the early 1960s an experimental "learning machine" with punched tape memory, called Cybertron, had been developed by Raytheon Company to analyze sonar signals, electrocardiograms, and speech patterns using rudimentary reinforcement learning. It was repetitively "trained" by a human operator/teacher to recognize patterns and equipped with a "goof" button to cause it to re-evaluate incorrect decisions. A representative book on research into machine learning during the 1960s was Nilsson's book on Learning Machines, dealing mostly with machine learning for pattern classification. Interest related to pattern recognition continued into the 1970s, as described by Duda and Hart in 1973. In 1981 a report was given on using teaching strategies so that a neural network learns to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal.

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?"

1.2.2. MACHINE LEARNING MODELS

Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

There are various types of machine learning models available based on different business goals and data sets.

→ CLASSIFICATION OF MACHINE LEARNING MODELS

Based on different business goals and data sets, there are three learning models for algorithms. Each machine learning algorithm settles into one of the three models:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

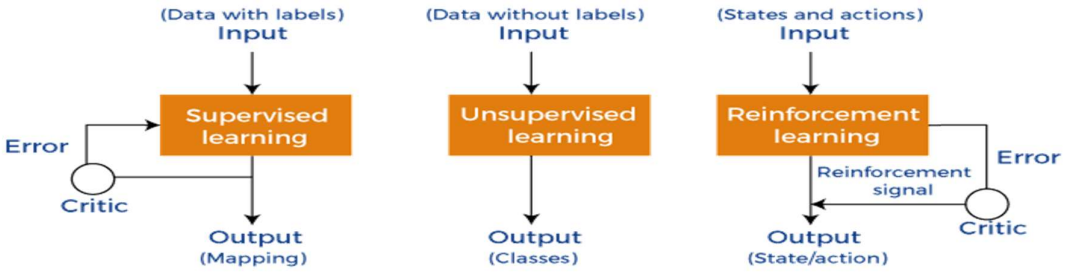


Figure-1.16
MACHINE LEARNING MODEL

- **Supervised Machine Learning Models**

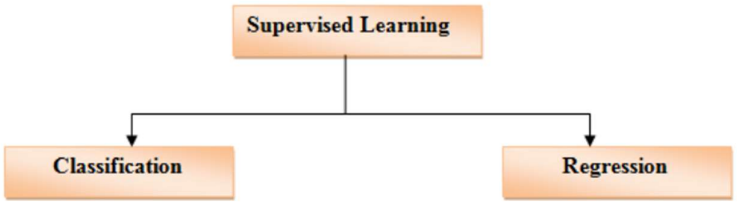


Figure-1.17
SUPERVISED LEARNING

Supervised Learning is the simplest machine learning model to understand in which input data is called training data and has a known label or result as an output. So, it works on the principle of input-output pairs. It requires creating a function that can be trained using a training data set, and then it is applied to unknown data and makes some predictive performance. Supervised learning is task-based and tested on labelled data sets.

We can implement a supervised learning model on simple real-life problems. For example, we have a dataset consisting of age and height; then, we can build a supervised learning model to predict the person's height based on their age.

Supervised Learning models are further classified into two categories:

❖ REGRESSION

In regression problems, the output is a continuous variable. Some commonly used Regression models are as follows:

a) Linear Regression

Linear regression is the simplest machine learning model in which we try to predict one output variable using one or more input variables. The representation of linear regression is a linear equation, which combines a set of input values(x) and predicted output(y) for the set of those input values. It is represented in the form of a line:

$Y = bx + c.$

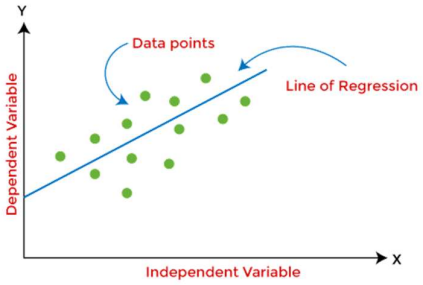


Figure-1.18
LINEAR REGRESSION

The main aim of the linear regression model is to find the best fit line that best fits the data points.

Linear regression is extended to multiple linear regression (find a plane of best fit) and polynomial regression (find the best fit curve).

b) Decision Tree

Decision trees are the popular machine learning models that can be used for both regression and classification problems.

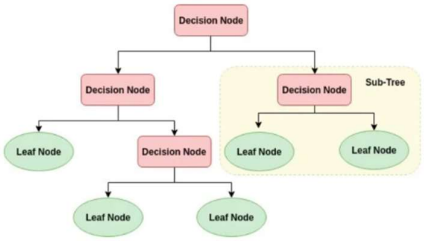


Figure-1.19
DECISION TREE

A decision tree uses a tree-like structure of decisions along with their possible consequences and outcomes. In this, each internal node is used to represent a test on an attribute; each branch is used to represent the outcome of the test. The more nodes a decision tree has, the more accurate the result will be.

The advantage of decision trees is that they are intuitive and easy to implement, but they lack accuracy.

Decision trees are widely used in **operations research, specifically in decision analysis, strategic planning**, and mainly in machine learning.

c) Random Forest

Random Forest is the ensemble learning method, which consists of a large number of decision trees. Each decision tree in a random forest predicts an outcome, and the prediction with the majority of votes is considered as the outcome. A random forest model can be used for both regression and classification problems.

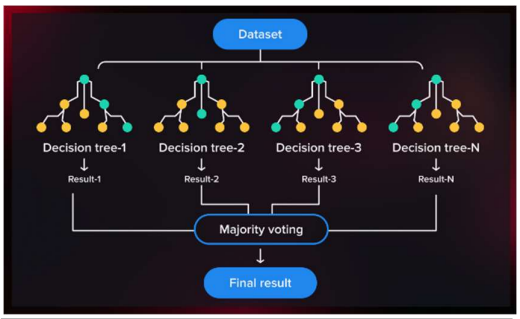


Figure-1.20
RANDOM FOREST

For the classification task, the outcome of the random forest is taken from the majority of votes. Whereas in the regression task, the outcome is taken from the mean or average of the predictions generated by each tree.

d) Neural Networks

Neural networks are the subset of machine learning and are also known as artificial neural networks. Neural networks are made up of artificial neurons and designed in a way that resembles the human brain structure and working. Each artificial neuron connects with many other neurons in a neural network, and such millions of connected neurons create a sophisticated cognitive structure.

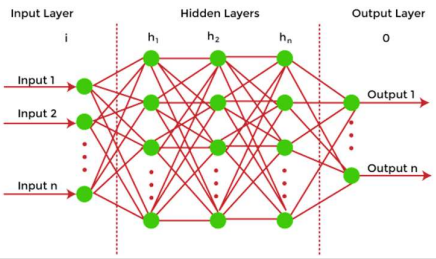


Figure-1.21
NEURAL NETWORKS

Neural network consists of a multilayer structure, containing one input layer, one or more hidden layers, and one output layer. As each neuron is connected with another neuron, it transfers data from one layer to the other neuron of the next layers. Finally, data reaches the last layer or output layer of the neural network and generates output.

Neural networks depend on training data to learn and improve their accuracy. However, a perfectly trained & accurate neural network can cluster data quickly and become a powerful machine learning and AI tool. One of the best-known neural networks is **Google's search algorithm.**

❖ CLASSIFICATION

Classification models are the second type of Supervised Learning techniques, which are used to generate conclusions from observed values in the categorical form. For example, the classification model can identify if the email is spam or not; a buyer will purchase the product or not, etc. Classification algorithms are used to predict two classes and categorize the output into different groups.

In classification, a classifier model is designed that classifies the dataset into different categories, and each category is assigned a label.

There are two types of classifications in machine learning:

- **Binary classification**: If the problem has only two possible classes, called a binary classifier. For example, cat or dog, Yes or No,
- **Multi-class classification**: If the problem has more than two possible classes, it is a multi-class classifier.

Some popular classification algorithms are as below:

a) Logistic Regression

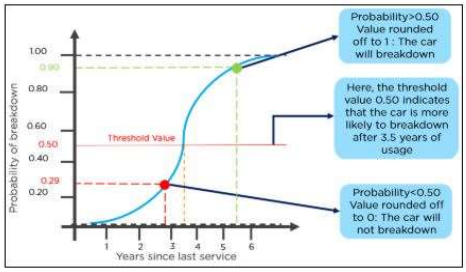


Figure-1.22
LOGISTIC REGRESSION

Logistic Regression is used to solve the classification problems in machine learning. They are similar to linear regression but used to predict categorical variables. It can predict the output in either Yes or No, 0 or 1, True or False, etc. However, rather than giving the exact values, it provides the probabilistic values between 0 & 1.

b) Support Vector Machine

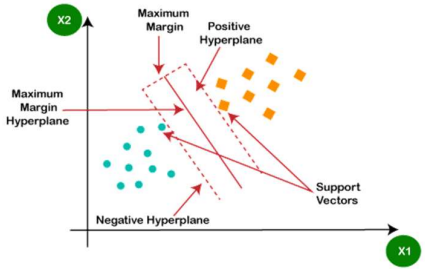


Figure-1.23
SUPPORT VECTOR MACHINE

Support vector machine or SVM is the popular machine learning algorithm, which is widely used for classification and regression tasks. However, specifically, it is used to solve classification problems. The main aim of SVM is to find the best decision boundaries in an N-dimensional space, which can segregate data points into classes, and the best decision boundary is known as Hyperplane. SVM selects the extreme vector to find the hyperplane, and these vectors are known as support vectors.

c) Naïve Bayes

Naïve Bayes is another popular classification algorithm used in machine learning. It is called so as it is based on Bayes theorem and follows the naïve(independent) assumption between the features which is given as:

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)}$$

Figure-1.24
NAIVE BAYES

Each naïve Bayes classifier assumes that the value of a specific variable is independent of any other variable/feature. For example, if a fruit needs to be classified based on colour, shape,

and taste. So yellow, oval, and sweet will be recognized as mango. Here each feature is independent of other features.

→ **Unsupervised Machine learning models**

Unsupervised Machine learning models implement the learning process opposite to supervised learning, which means it enables the model to learn from the unlabelled training dataset. Based on the unlabelled dataset, the model predicts the output. Using unsupervised learning, the model learns hidden patterns from the dataset by itself without any supervision.

Unsupervised learning models are mainly used to perform three tasks, which are as follows:

a. Clustering

Clustering is an unsupervised learning technique that involves clustering or groping the data points into different clusters based on similarities and differences. The objects with the most similarities remain in the same group, and they have no similarities from other groups.

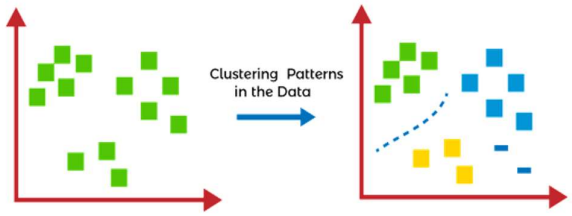


Figure-1.25
CLUSTERING

Some commonly used Clustering algorithms are K-means Clustering, hierarchal Clustering & DBSCAN.

b. Reinforcement Learning

In reinforcement learning, the algorithm learns actions for given set of states that lead goal state. It is a feedback-based learning model that takes feedback signals after each state or action by interacting with the environment. This feedback works as a reward and the agent's goal is to maximize the positive rewards to improve their performance.

The behaviour of the model in reinforcement learning is similar to human learning, as humans learn things by experiences as feedback and interact with the environment.

Below are some popular algorithms that come under reinforcement learning:

- **Q-learning:** Q-learning is one of the popular model-free algorithms of reinforcement learning, which is based on the Bellman equation. It aims to learn the policy that can help the AI agent to take the best action for maximizing the reward under a specific circumstance. It incorporates Q values for each state-action pair that indicate the reward to following a given state path, and it tries to maximize the Q-value.

- **State-Action-Reward-State-Action (SARSA):** SARSA is an On-policy algorithm based on the Markov decision process. It uses the action performed by the current policy to learn the Q-value. The SARSA algorithm stands **for State Action Reward State Action, which symbolizes the tuple (s, a, r, s', a').**
- **Deep Q Network:** DQN **or Deep Q Neural network is Q-learning** within the neural network. It is basically employed in a big state space environment where defining a Q-table would be a complex task. So, in such a case, rather than using Q-table, the neural network uses Q-values for each action based on the state.

1.2.3. FEATURES OF MACHINE LEARNING

1. Machine learning uses data to detect various patterns in a given dataset.
2. It can learn from past data and improve automatically.
3. It is a data-driven technology.
4. Machine learning is much similar to data mining as it also deals with the huge amount of the data.

1.3. ABOUT THE PROJECT

In this project, we aim to build a machine learning model that can predict the price of cars based on various features and attributes. The ability to accurately predict car prices can be valuable for both buyers and sellers in the automotive market. Buyers can use the predictions to make informed decisions, ensuring they get a fair deal, while sellers can use it to set competitive prices for their vehicles.

1.3.1. DATASET

For this project, we'll be using a dataset that contains information about different cars, including their make, model, year of manufacturing, mileage, engine specifications, fuel type, transmission type, and several other relevant features. The dataset may also include historical car prices.

1.3.2. OBJECTIVE

The primary objective of this project is to develop a robust predictive model that can estimate car prices based on the input features. We will follow a data-driven approach, leveraging machine learning techniques to achieve this goal. The dataset will be split into a training set and a test set to evaluate the model's performance accurately.

Steps:

1. Data Exploration and Preprocessing: We'll start by exploring the dataset, understanding the features, and checking for missing or irrelevant data. If required, we'll perform data cleaning and preprocessing to ensure the data is in a suitable format for training the model.
2. Feature Selection and Engineering: Some features in the dataset may not directly contribute to the car price prediction or might be redundant. We'll analyze the features and select the most relevant ones. Additionally, we might engineer new features if we find that they can improve the model's predictive power.
3. Model Selection: We'll experiment with various machine learning algorithms, such as linear regression, decision trees, random forests, support vector machines, or gradient boosting, to identify the best model for the car price prediction task. Hyperparameter tuning and cross-validation will be used to optimize the model's performance.
4. Model Training and Evaluation: After selecting the best model, we'll train it on the training dataset and evaluate its performance using the test dataset. We'll use metrics like mean squared error (MSE), mean absolute error (MAE), and R-squared to assess the model's accuracy and robustness.
5. Deployment and Testing: Once we have a satisfactory model, we'll deploy it into a production environment, allowing users to input car features and obtain predicted prices. We'll conduct additional tests to ensure the model performs well with real-world data and handles various scenarios effectively.

Figure-1.26
MODEL DEPLOYMENT

CHAPTER-2

PROJECT WORK

PROJECT NAME: CAR PRICE PREDICTION USING MACHINE LEARNING

2.1. ABOUT THE DATA

2.1.1. THE DATASET

- The data is collected from **Kaggle**. Below is the link for data.

(https://github.com/arsh0220/DATASETS/blob/main/CAR%20DETAILS%20(SECOND%20HAND%20CARS).csv)

- In the dataset, we have the data of different car brands which will help a user to buy a car at the best price.
- Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors.
- Basically, each feature of the dataset equally contributes to the change in selling price.
- The csv file of dataset is downloaded for further processing.

```
cars.describe().T
```

	count	mean	std	min	25%	50%	75%	max
year	4340.0	2013.090783	4.215344	1992.0	2011.00	2014.0	2016.0	2020.0
selling_price	4340.0	504127.311751	578548.736139	20000.0	208749.75	350000.0	600000.0	8900000.0
km_driven	4340.0	66215.777419	46644.102194	1.0	35000.00	60000.0	90000.0	806599.0
fuel	4340.0	0.531336	0.549621	0.0	0.00	1.0	1.0	4.0
seller_type	4340.0	0.794470	0.458629	0.0	1.00	1.0	1.0	2.0
transmission	4340.0	0.103226	0.304289	0.0	0.00	0.0	0.0	1.0
owner	4340.0	0.466590	0.740330	0.0	0.00	0.0	1.0	4.0
brand	4340.0	4.867512	5.637731	0.0	0.00	3.0	10.0	28.0
model	4340.0	58.927419	51.901595	1.0	12.00	41.0	106.0	187.0

Code-2.1
DESCRIPTION OF THE DATASET

2.1.2. LIBRARIES USED

- Pandas

- Matplotlib
- Seaborn
- Sci-Kit Learn

2.1.3. LOADING THE DATASET

- The downloaded csv file is used to load the data to process using the **read_csv** method in panda library.

```
import pandas as pd
url='https://raw.githubusercontent.com/arsh0220/DATASETS/main/CAR%20DETAILS%20(SECOND%20HAND%20CARS).csv'
cars=pd.read_csv(url)
cars
```

Code-2.2
DATA LOADING

- The training data consists of 8 features and 4340 car brands.

2.1.4. FEATURES OF THE DATASET

2.1.4.1. Name of Car:

- This feature represents the name or model of the car being sold.
- It can include various details, such as the brand, model series, and specific variant of the car.
- Significance: The name of the car, including the brand, model, and variant, can have a substantial impact on the car's price. High-end or luxury brands/models may generally command higher prices compared to regular or budget-friendly models. Rare or limited-edition variants might also fetch higher prices due to their exclusivity.

2.1.4.2. Year of Car:

- This feature indicates the manufacturing year of the car.
- It provides valuable information about the car's age, which can affect its market value.
- Significance: The manufacturing year of the car is a crucial feature as it directly affects the car's depreciation and overall condition. Newer cars tend to have higher prices compared to older ones, assuming all other factors remain constant. Older cars may have lower prices due to wear and tear and outdated features.

2.1.4.3. Selling Price:

- The selling price is the target variable in the dataset, and it represents the price at which the car is being sold.
- This is the variable that the machine learning model will be trained to predict.
- Significance: The selling price is the most critical variable in the prediction task since it is what we want to predict. It depends on all the other features in the dataset, and the model's goal is to learn how these features influence the final selling price.

2.1.4.4. Fuel Type:

- This feature specifies the type of fuel used by the car.
- Common values include “Petrol”, “Diesel”, “CNG” (Compressed Natural Gas), “LPG”, “Electric” etc.
- Significance: Fuel type can impact the car's price due to differences in fuel efficiency, availability, and environmental concerns. For example, electric or hybrid cars may have higher initial prices due to their eco-friendliness and lower running costs.

2.1.4.5. Seller Type:

- Seller type indicates the category or type of the seller selling the car.
- For example, it might include values like “Individual”, “Dealer” or “Trustmark Dealer”.
- Significance: The type of seller can influence the car's price. Dealers or showrooms may mark up prices compared to individual sellers because they offer additional services, warranties, or refurbishments. Individual sellers, on the other hand, might offer more negotiable prices.

2.1.4.6. Transmission:

- This feature represents the type of transmission in the car.
- Common values include “Manual” and “Automatic.”
- Significance: The type of transmission affects the driving experience and convenience. Automatic transmission cars generally have higher prices compared to manual transmission cars. Automatic transmissions are considered more user-friendly, especially in urban traffic, and may appeal to a broader market.

2.1.4.7. Owner:

- The owner feature shows the number of previous owners the car has had.
- It can take values like “First Owner”, “Second Owner”, “Third Owner”, etc.
- Significance: The number of previous owners can impact the car's price. First-owner cars might be perceived as having a better history and are often priced higher than multiple-owner cars. Cars with more owners may be priced lower due to concerns about their maintenance and condition.

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

Figure-2.1
FEATURES OF THE DATA

2.1.5. DATA PREPROCESSING & QUALITY HANDLING

- The main objective of preprocessing and handling data is to check whether the data used in project is reliable, accurate and suitable for analysis.
- Dataset is checked for any null values.

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   name            4340 non-null   object
 1   year            4340 non-null   int64 
 2   selling_price   4340 non-null   int64 
 3   km_driven       4340 non-null   int64 
 4   fuel            4340 non-null   object
 5   seller_type     4340 non-null   object
 6   transmission    4340 non-null   object
 7   owner           4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

Code-2.3
TO CHECK FOR NULL VALUES

- Our dataset does not contain any null values, which means are data is clean so we don’t need to drop any row.

2.1.5.1. ANALYSING THE NAME COLUMN:

- The NAME column consists of various different cars brand and models, the price of the car highly depends on the NAME.
- The NAME column has large number of entries which makes it difficult for us to replace every name with a numeric value and apply in model.

```
cars['name'].unique().shape
```

```
(1491,)
```

Code-2.4
CHECK FOR UNIQUE ENTRIES IN "NAME"

- To overcome this problem, we splited the names of the car using the [.split( )] method.

```
c="Maruti 800 AC"
c.split()
```

```
['Maruti', '800', 'AC']
```

Code-2.5
SPLITTING THE NAME

- Using for loop and various codes the whole NAME column is splited and the first word and second word are chosen as brand and model respectively.

```
for i in cars['name']:
  print(i)
```

```
Maruti 800 AC
Maruti Wagon R LXI Minor
Hyundai Verna 1.6 SX
Datsun RediGO T Option
Honda Amaze VX i-DTEC
Maruti Alto LX BSIII
Hyundai Xcent 1.2 Kappa S
Tata Indigo Grand Petrol
Hyundai Creta 1.6 VTVT S
Maruti Celerio Green VXI
Chevrolet Sail 1.2 Base
Tata Indigo Grand Petrol
Toyota Corolla Altis 1.8 VL CVT
Maruti 800 AC
Maruti Wagon R LXI Minor
Hyundai Verna 1.6 SX
Datsun RediGO T Option
Honda Amaze VX i-DTEC
Maruti Alto LX BSIII
Hyundai Xcent 1.2 Kappa S
```

Code-2.6
PRINTING THE NAMES OF CAR

```
mod=[]
for i in cars['name']:
  mod.append(i.split(' '))
mod
```

```
[['Maruti', '800', 'AC'],
 ['Maruti', 'Wagon', 'R', 'LXI', 'Minor'],
 ['Hyundai', 'Verna', '1.6', 'SX'],
 ['Datsun', 'RediGO', 'T', 'Option'],
 ['Honda', 'Amaze', 'VX', 'i-DTEC'],
 ['Maruti', 'Alto', 'LX', 'BSIII'],
 ['Hyundai', 'Xcent', '1.2', 'Kappa', 'S'],
 ['Tata', 'Indigo', 'Grand', 'Petrol'],
 ['Hyundai', 'Creta', '1.6', 'VTVT', 'S'],
 ['Maruti', 'Celerio', 'Green', 'VXI'],
 ['Chevrolet', 'Sail', '1.2', 'Base'],
 ['Tata', 'Indigo', 'Grand', 'Petrol'],
 ['Toyota', 'Corolla', 'Altis', '1.8', 'VL', 'CVT'],
 ['Maruti', '800', 'AC'],
 ['Maruti', 'Wagon', 'R', 'LXI', 'Minor'],
 ['Hyundai', 'Verna', '1.6', 'SX'],
 ['Datsun', 'RediGO', 'T', 'Option'],
 ['Honda', 'Amaze', 'VX', 'i-DTEC'],
 ['Maruti', 'Alto', 'LX', 'BSIII'],
```

Code-2.7
SPLITTING THE NAMES

```
brand=[]
m=[]
for i in cars['name']:
  mo=i.split(' ')
  first_word=mo[0]
  second_word=mo[1]
  brand.append(first_word)
  m.append(second_word)
brand
```

```
['Maruti',
 'Maruti',
 'Hyundai',
 'Datsun',
 'Honda',
 'Maruti',
 'Hyundai',
 'Tata',
 'Hyundai',
 'Maruti',
 'Chevrolet',
 'Tata',
 'Toyota',
 'Maruti',
 'Maruti',
 'Hyundai',
 'Datsun',
 'Honda',
 'Maruti',
 'Hyundai',
```

Code-2.8
PRINTING FIRST WORD (BRAND)

```
m
```

```
['800',
 'Wagon',
 'Verna',
 'RediGO',
 'Amaze',
 'Alto',
 'Xcent',
 'Indigo',
 'Creta',
 'Celerio',
 'Sail',
 'Indigo',
 'Corolla',
 '800',
 'Wagon',
 'Verna',
 'RediGO',
 'Amaze',
 'Alto',
 'Xcent',
 'Indigo',
 'Creta',
 'Celerio',
 'Sail',
 'Indigo',
 'Corolla',
 'Ciaz',
 'Venue',
 'Enjoy',
 'XF',
 'New',
```

Code-2.9
PRINTING SECOND WORD (MODEL)

```
cars["brand"].unique()
```

```
array(['Maruti', 'Hyundai', 'Datsun', 'Honda', 'Tata', 'Chevrolet',
       'Toyota', 'Jaguar', 'Mercedes-Benz', 'Audi', 'Skoda', 'Jeep',
       'BMW', 'Mahindra', 'Ford', 'Nissan', 'Renault', 'Fiat',
       'Volkswagen', 'Volvo', 'Mitsubishi', 'Land Rover', 'Daewoo', 'MG',
       'Force', 'Isuzu', 'OpelCorsa', 'Ambassador', 'Kia'], dtype=object)
```

```
len(cars['brand'].unique())
```

```
29
```

Code-2.10
PRINTING UNIQUE BRANDS

```
cars['model'].replace({'Wagon':'Wagon R','City':'Honda City','Rover':'Land Rover','CLASSIC':'Classic','redi-GO':'RediGO'},inplace=True)
cars['model'].unique()
```

```
array(['800', 'Wagon R', 'Verna', 'RediGO', 'Amaze', 'Alto', 'Xcent',
       'Indigo', 'Creta', 'Celerio', 'Sail', 'Corolla', 'Ciaz', 'Venue',
       'Enjoy', 'XF', 'New', 'Vitara', 'Q5', 'Honda City', 'Tigor', 'A6',
       'Superb', 'Innova', 'Compass', 'E-Class', 'i10', '3', 'Q7',
       'Elantra', 'Scorpio', 'Santro', 'Grand', 'Swift', 'Eeco', 'i20',
       'Omni', 'Jeep', 'Indica', 'EON', 'Etios', 'Tavera', 'EcoSport',
       'Civic', 'Rapid', 'Getz', 'Terrano', 'Elite', 'Brio', 'S-Class',
       'XUV500', 'Duster', 'Bolero', 'Avventura', 'A8', 'Jetta', 'A4',
       'X1', 'V40', 'SX4', '7', 'Sonata', 'Micra', 'Xylo', 'KWID',
       'Ertiga', 'Beat', 'Zen', 'Baleno', 'Nano', 'Cruze', 'Figo',
       'Spark', 'Bolt', 'Quanto', 'XJ', 'Nexon', 'Vento', 'Esteem', '5',
       'Linea', 'Scala', 'XUV300', 'S-Cross', 'Ameo', 'Renault', 'Optra',
       'Mobilio', 'Zest', 'Fabia', 'GO', 'Fiesta', 'Sumo', 'Jazz',
       'Tiago', 'Marazzo', 'A-Star', 'Yeti', 'Aspire', 'Outlander',
       'Endeavour', 'Polo', 'Ritz', 'Estilo', 'Manza', 'Safari',
       'Fortuner', 'Ecosport', 'Supro', 'KUV', 'Accord', 'TUV', 'BR-V',
       'Ikon', 'Punto', 'WR-V', 'Laura', 'Tucson', 'GL-Class', 'Pulse',
       'C-Class', 'X5', 'Gypsy', 'Verito', 'Hexa', 'Venture', 'Captiva',
       'Aveo', 'Thar', 'Octavia', 'Accent', 'Land Rover', 'Lodgy',
       'Grande', 'Palio', 'M-Class', 'Ignis', 'Sunny', 'Q3', 'A5',
       'Camry', 'Pajero', 'Xenon', 'Captur', 'Passat', 'Freestyle',
       'Alturas', 'Aria', 'CR-V', 'Matiz', 'Hector', 'Santa', 'One',
       'NuvoSport', 'S-Presso', 'Yaris', 'Fusion', 'Fluence', 'Evalia',
       'Koleos', 'Harrier', 'Altroz', 'D-Max', 'Kicks', '1.6Gls', 'BRV',
       'B', '500', 'Triber', 'Montero', 'X-Trail', 'XC60', 'Classic',
       'Seltos', '1.4', 'Ingenio', 'CrossPolo', 'Spacio', 'Winger', 'RS7',
       'GLS', 'XC', 'Qualis'], dtype=object)
```

```
cars['model'].unique().shape
```

```
(183,)
```

Code-2.11
PRINTING UNIQUE MODELS

- After separating Brand and Model from the NAME we will check for any duplicate values and will resolve the duplicacy to avoid errors.

```
def find_duplicates_using_set(lst):
    unique_items = set(lst)
    if len(unique_items) == len(lst):
        return []  # No duplicates found
    else:
        duplicates = [item for item in unique_items if lst.count(item) > 1]
        return duplicates
print(find_duplicates_using_set(old_values))
```

```
['New', 'Grand', 'Classic']
```

Code-2.12
CHECKING FOR DUPLICACY

```
cars.loc[(cars['brand'] == 'Maruti') & (cars['model'] == 'Grand'), 'model'] = 'Grand Vitara'
```

```
cars[(cars['brand'] == 'Maruti') & (cars['model'] == 'Grand Vitara')]
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	brand	model
3397	Maruti Grand Vitara MT	2007	479000	145000	Petrol	Individual	Manual	Third Owner	Maruti	Grand Vitara

```
cars.loc[(cars['brand'] == 'Hyundai') & (cars['model'] == 'Grand')]['name'].unique()
```

```
array(['Hyundai Grand i10 1.2 Kappa Asta',
       'Hyundai Grand i10 CRDi Sportz', 'Hyundai Grand i10 Magna',
       'Hyundai Grand i10 1.2 CRDi Asta', 'Hyundai Grand i10 CRDi Magna',
       'Hyundai Grand i10 1.2 Kappa Sportz Option',
       'Hyundai Grand i10 Sportz', 'Hyundai Grand i10 1.2 Kappa Magna AT',
       'Hyundai Grand i10 Magna AT', 'Hyundai Grand i10 Nios Magna CRDi',
       'Hyundai Grand i10 1.2 Kappa Sportz Dual Tone',
       'Hyundai Grand i10 AT Asta',
       'Hyundai Grand i10 1.2 Kappa Sportz BSIV',
       'Hyundai Grand i10 Asta', 'Hyundai Grand i10 Asta Option',
       'Hyundai Grand i10 1.2 Kappa Magna BSIV',
       'Hyundai Grand i10 1.2 CRDi Magna',
```

```
cars.loc[(cars['brand'] == 'Maruti') & (cars['model'] == 'Grand'), 'model'] = 'Grand Vitara'
```

```
cars[(cars['brand'] == 'Maruti') & (cars['model'] == 'Grand Vitara')]
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	brand	model
3397	Maruti Grand Vitara MT	2007	479000	145000	Petrol	Individual	Manual	Third Owner	Maruti	Grand Vitara

```
cars.loc[(cars['brand'] == 'Hyundai') & (cars['model'] == 'Grand')]['name'].unique()
```

```
array(['Hyundai Grand i10 1.2 Kappa Asta',
       'Hyundai Grand i10 CRDi Sportz', 'Hyundai Grand i10 Magna',
       'Hyundai Grand i10 1.2 CRDi Asta', 'Hyundai Grand i10 CRDi Magna',
       'Hyundai Grand i10 1.2 Kappa Sportz Option',
       'Hyundai Grand i10 Sportz', 'Hyundai Grand i10 1.2 Kappa Magna AT',
       'Hyundai Grand i10 Magna AT', 'Hyundai Grand i10 Nios Magna CRDi',
       'Hyundai Grand i10 1.2 Kappa Sportz Dual Tone',
       'Hyundai Grand i10 AT Asta',
       'Hyundai Grand i10 1.2 Kappa Sportz BSIV',
       'Hyundai Grand i10 Asta', 'Hyundai Grand i10 Asta Option',
       'Hyundai Grand i10 1.2 Kappa Magna BSIV',
       'Hyundai Grand i10 1.2 CRDi Magna',
         'Hyundai Grand i10 1.2 Kappa Era', 'Hyundai Grand i10 CRDi Asta',
         'Hyundai Grand i10 1.2 Kappa Sportz AT',
         'Hyundai Grand i10 Asta Option AT'], dtype=object)
```

```
cars.loc[(cars['brand'] == 'Hyundai') & (cars['model'] == 'Grand'), 'model'] = 'Grand i10'
```

```
cars.loc[(cars['brand'] == 'Tata') & (cars['model'] == 'New')]['name'].unique()
```

```
array(['Tata New Safari DICOR 2.2 EX 4x2',
       'Tata New Safari DICOR 2.2 GX 4x2',
       'Tata New Safari DICOR 2.2 EX 4x4',
       'Tata New Safari DICOR 2.2 GX 4x2 BS IV', 'Tata New Safari 4X4',
       'Tata New Safari Dicor EX 4X2 BS IV', 'Tata New Safari 4X4 EX',
       'Tata New Safari DICOR 2.2 VX 4x4', 'Tata New Safari 4X2',
       'Tata New Safari 3L Dicor LX 4x2',
       'Tata New Safari DICOR 2.2 VX 4x2'], dtype=object)
```

```
cars.loc[(cars['brand'] == 'Tata') & (cars['model'] == 'New'), 'model'] = 'New Safari'
```

```
cars.loc[(cars['brand'] == 'Mercedes-Benz') & (cars['model'] == 'New')]['name'].unique()
```

```
array(['Mercedes-Benz New C-Class 220 CDI AT',
       'Mercedes-Benz New C-Class C 220 CDI Avantgarde',
       'Mercedes-Benz New C-Class C 220 CDI BE Avantgare',
       'Mercedes-Benz New C-Class C 220 CDI Grand Edition',
       'Mercedes-Benz New C-Class 200 CDI Classic'], dtype=object)
```

Code-2.13
RESOLVING DUPLICACY

2.1.5.2. REPLACING ALL THE CHARACTER DATA INTO INTEGER DATA

- Now as we have analysed the column and no resolved the duplicacy, we will replace character data to integer data.

```
old_values=['800', 'Wagon R', 'Alto', 'Celerio', 'Ciaz', 'Vitara', 'Swift',
       'Eeco', 'Omni', 'SX4', 'Ertiga', 'Zen', 'Baleno', 'Esteem',
       'S-Cross', 'A-Star', 'Ritz', 'Estilo', 'Gypsy', 'Ignis',
       'S-Presso', 'Grand Vitara','Verna', 'Xcent', 'Creta', 'Venue', 'i10', 'Elantra', 'Santro',
       'Grand i10', 'i20', 'EON', 'Getz', 'Elite', 'Sonata', 'Tucson',
       'Accent', 'Santa','RediGO', 'GO','Amaze', 'Honda City', 'Civic', 'Brio', 'Mobilio', 'Jazz', 'Accord',
       'BR-V', 'WR-V', 'CR-V', 'BRV','Indigo', 'Tigor', 'Indica', 'Nano', 'Bolt', 'Nexon', 'Zest',
       'Sumo', 'Tiago', 'Manza', 'Safari', 'New Safari', 'Hexa', 'Venture',
       'Xenon', 'Aria', 'Harrier', 'Altroz', 'Spacio', 'Winger','Sail', 'Enjoy', 'Tavera', 'Beat', 'Cruze', 'Spark', 'Optra',
       'Captiva', 'Aveo','Corolla', 'Innova', 'Etios', 'Fortuner', 'Camry', 'Yaris',
       'Qualis','XF', 'XJ','New C-Class', 'E-Class', 'S-Class', 'GL-Class', 'C-Class', 'M-Class', 'B',
       'GLS','Q5', 'A6', 'Q7', 'A8', 'A4', 'Q3', 'A5', 'RS7','Superb', 'Rapid', 'Fabia', 'Yeti', 'Laura', 'Octavia','Compass','3', 'X1', '7',
       'Renault', 'Marazzo', 'Supro', 'KUV', 'TUV', 'Verito', 'Thar',
       'Alturas', 'NuvoSport', 'Ingenio','EcoSport', 'Figo', 'Fiesta', 'Aspire', 'Endeavour', 'Ecosport',
       'Ikon', 'Freestyle', 'Fusion', 'Ford Classic','Terrano', 'Micra', 'Sunny', 'Evalia', 'Kicks', 'X-Trail','Duster', 'KWID', 'Scala', 'Pul
       'Koleos', 'Triber','Avventura', 'Linea', 'Punto', 'Grande', 'Palio', '500','Jetta', 'Vento', 'Ameo', 'Polo', 'Passat', 'CrossPolo','V40

new_values=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,

cars["model"].replace(old_values,new_values,inplace=True)

cars.replace({'fuel':{'Petrol':0, 'Diesel':1, 'CNG':2, 'LPG':3, 'Electric':4}},inplace=True)
cars.replace({'seller_type':{'Dealer':0, 'Individual':1,'Trustmark Dealer':2},},inplace=True)
cars.replace({'transmission':{'Manual':0, 'Automatic':1}},inplace=True)
cars.replace({'owner':{'First Owner':0, 'Second Owner':1,'Third Owner':2,'Fourth & Above Owner':3,'Test Drive Car':4 }},inplace=True)
cars.replace({'brand':{'Maruti':0,'Hyundai':1,'Datsun':2,'Honda':3,'Tata':4,'Chevrolet':5,'Toyota':6,'Jaguar':7,'Mercedes-Benz':8,'Audi':9,'Sko
cars
```

Code-2.14
REPLACING CHARACTER DATA WITH INTEGER DATA

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	brand	model
0	Maruti 800 AC	2007	60000	70000	0	1	0	0	0	1
1	Maruti Wagon R LXI Minor	2007	135000	50000	0	1	0	0	0	2
2	Hyundai Verna 1.6 SX	2012	600000	100000	1	1	0	0	1	23
3	Datsun RediGO T Option	2017	250000	46000	0	1	0	0	2	39
4	Honda Amaze VX i-DTEC	2014	450000	141000	1	1	0	1	3	41
...	...	...	...	...	...	...	...	...	...	...
4335	Hyundai i20 Magna 1.4 CRDi (Diesel)	2014	409999	80000	1	1	0	1	1	31
4336	Hyundai i20 Magna 1.4 CRDi	2014	409999	80000	1	1	0	1	1	31
4337	Maruti 800 AC BSIII	2009	110000	83000	0	1	0	1	0	1
4338	Hyundai Creta 1.6 CRDi SX Option	2016	865000	90000	1	1	0	0	1	25

Figure-2.2
REPLACED DATASET

- THE FINAL DATASET IS ALSO AVALIABLE ON THE BELOW LINK:

(https://github.com/arsh0220/PROJECT-DATASETS/blob/main/finalcars%20(2).csv

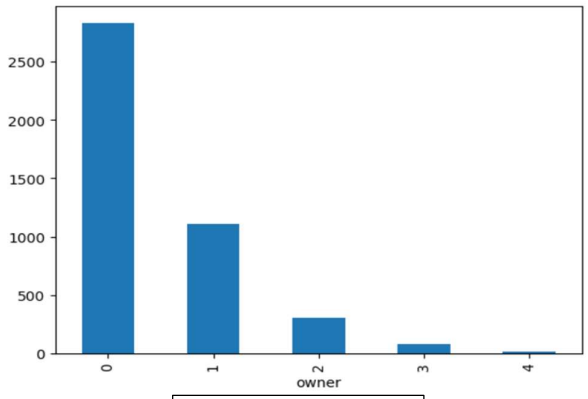
2.2. VISUALIZATION

2.2.1. MATPLOTLIB

- These codes groups the data by the 'owner', ‘transmission’, ‘name’, ‘selling price’ category and then counts the occurrences of each category and plots a histogram, line and bar graph.

```
import matplotlib.pyplot as plt
cars.groupby('owner')['owner'].count().plot(kind='bar')
```

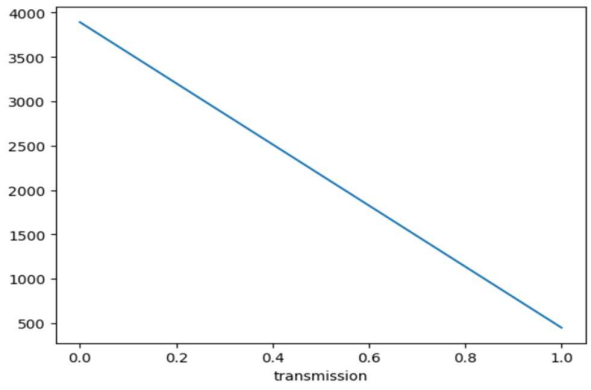
Code-2.15
TO PLOT GRAPH BASED ON OWNER COLUMN



Graph-2.1
BASED ON OWNER

```
cars.groupby('transmission')['transmission'].count().plot(kind='line')
```

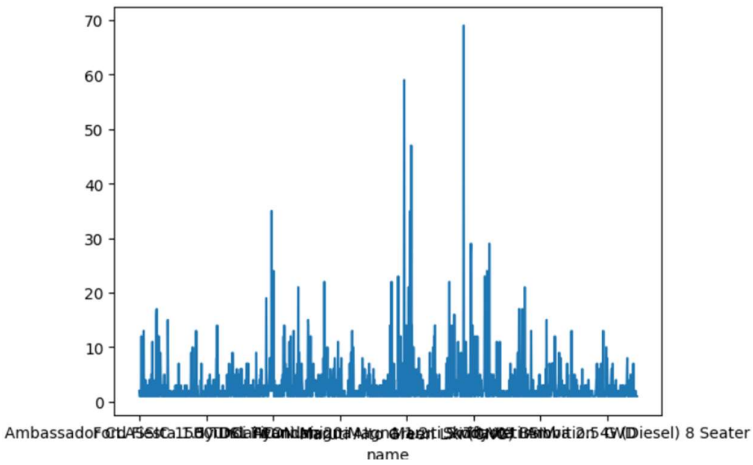
Code-2.16
TO PLOT GRAPH BASED ON TRANSMISSION COLUMN



Graph-2.2
BASED ON TRANSMISSION

```
cars.groupby('name')['name'].count().plot(kind='line')
```

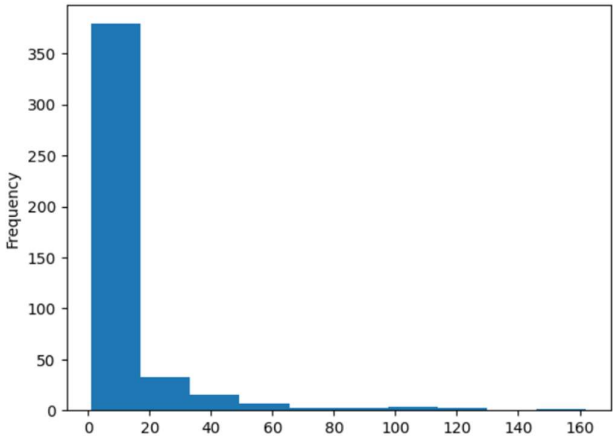
Code-2.17
TO PLOT GRAPH BASED ON NAME COLUMN



Graph-2.3
BASED ON NAME

```
cars.groupby('selling_price')['selling_price'].count().plot(kind='hist')
```

Code-2.18
TO PLOT GRAPH BASED ON SELLING PRICE



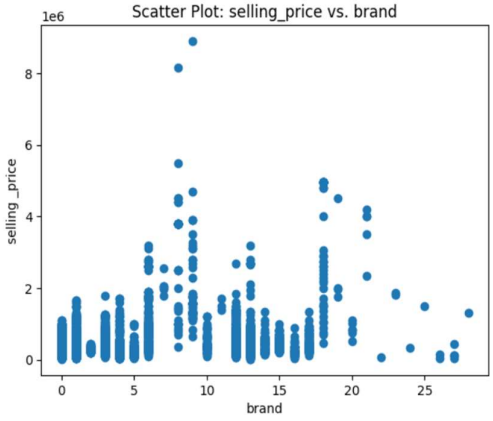
Graph-2.4
BASED ON SELLING PRICE

- Visualizing the relationship between selling price and the features.

```
plt.scatter(cars['name'], y)
plt.xlabel('name')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. name')
plt.show()
```

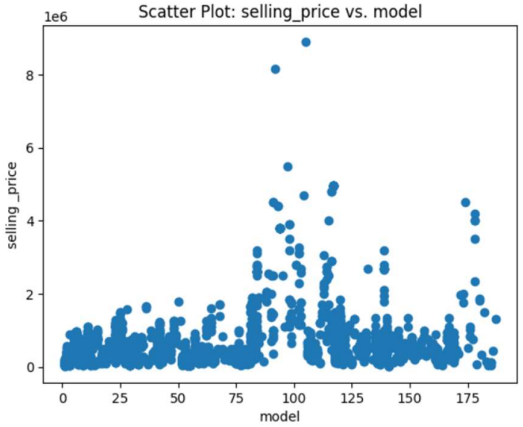
Graph-2.5
RELATIONSHIP BETWEEN "SELLING PRICE AND NAME"

```
plt.scatter(cars['brand'], y)
plt.xlabel('brand')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. brand')
plt.show()
```



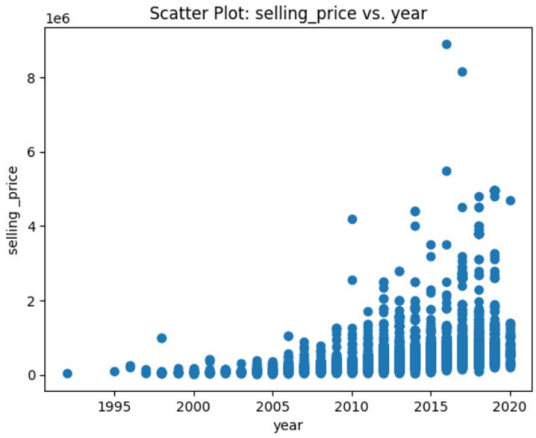
Graph-2.6
RELATIONSHIP BETWEEN "SELLING PRICE AND BRAND"

```
plt.scatter(cars['model'], y)
plt.xlabel('model')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. model')
plt.show()
```



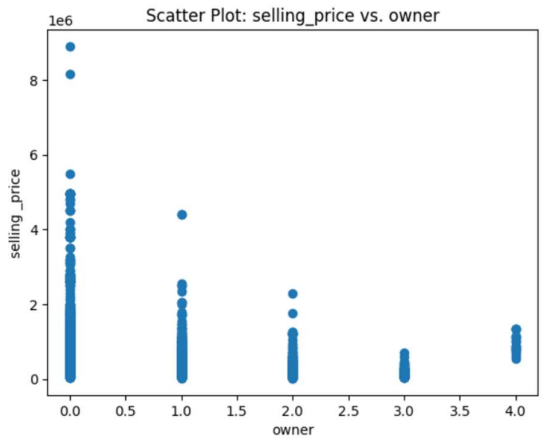
Graph-2.7
RELATIONSHIP BETWEEN "SELLING PRICE AND MODEL"

```
plt.scatter(cars['year'], y)
plt.xlabel('year')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. year')
plt.show()
```



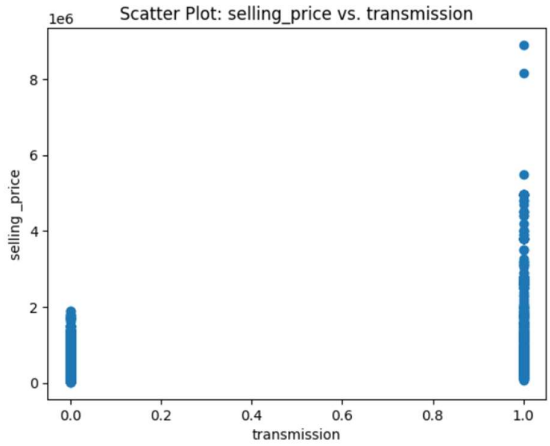
Graph-2.8
RELATIONSHIP BETWEEN "SELLING PRICE AND YEAR"

```
plt.scatter(cars['owner'], y)
plt.xlabel('owner')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. owner')
plt.show()
```



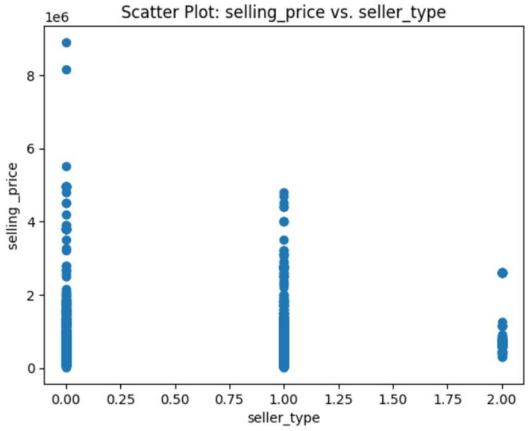
Graph-2.9
RELATIONSHIP BETWEEN "SELLING PRICE AND OWNER"

```
plt.scatter(cars['transmission'], y)
plt.xlabel('transmission')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. transmission')
plt.show()
```



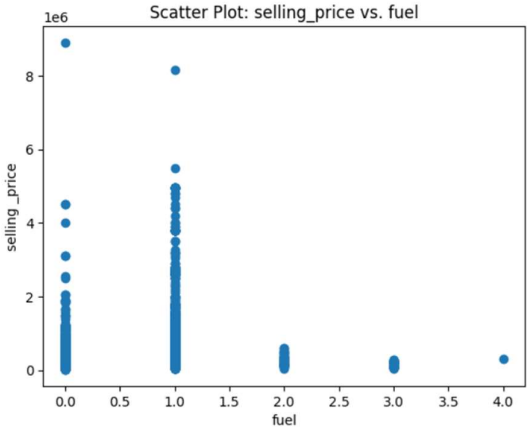
Graph-2.10
RELATIONSHIP BETWEEN "SELLING PRICE AND TRANSMISSION"

```
plt.scatter(cars['seller_type'], y)
plt.xlabel('seller_type')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. seller_type')
plt.show()
```



Graph-2.11
RELATIONSHIP BETWEEN "SELLING PRICE AND SELLER TYPE"

```
plt.scatter(cars['fuel'], y)
plt.xlabel('fuel')
plt.ylabel('selling _price')
plt.title('Scatter Plot: selling_price vs. fuel')
plt.show()
```

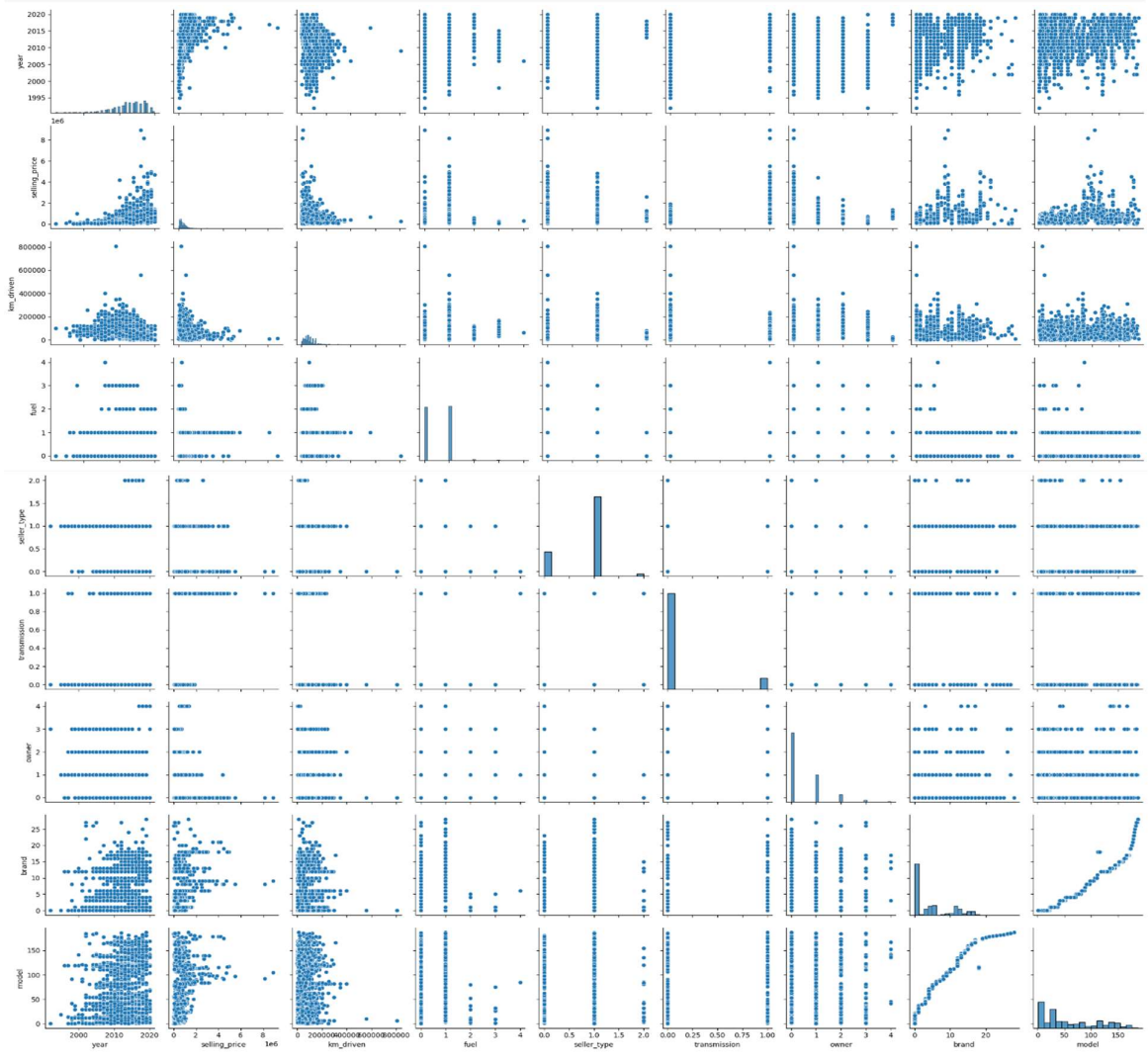


Graph-2.12
RELATIONSHIP BETWEEN "SELLING PRICE AND FUEL"

2.2.2. **SEABORN**

- The code **sns.pairplot(cars)** creates a grid of scatter plots and histograms known as a "Pair Plot" or "Scatterplot Matrix." This type of graph is a powerful visualization tool used to explore the relationships between multiple numerical variables in a dataset.

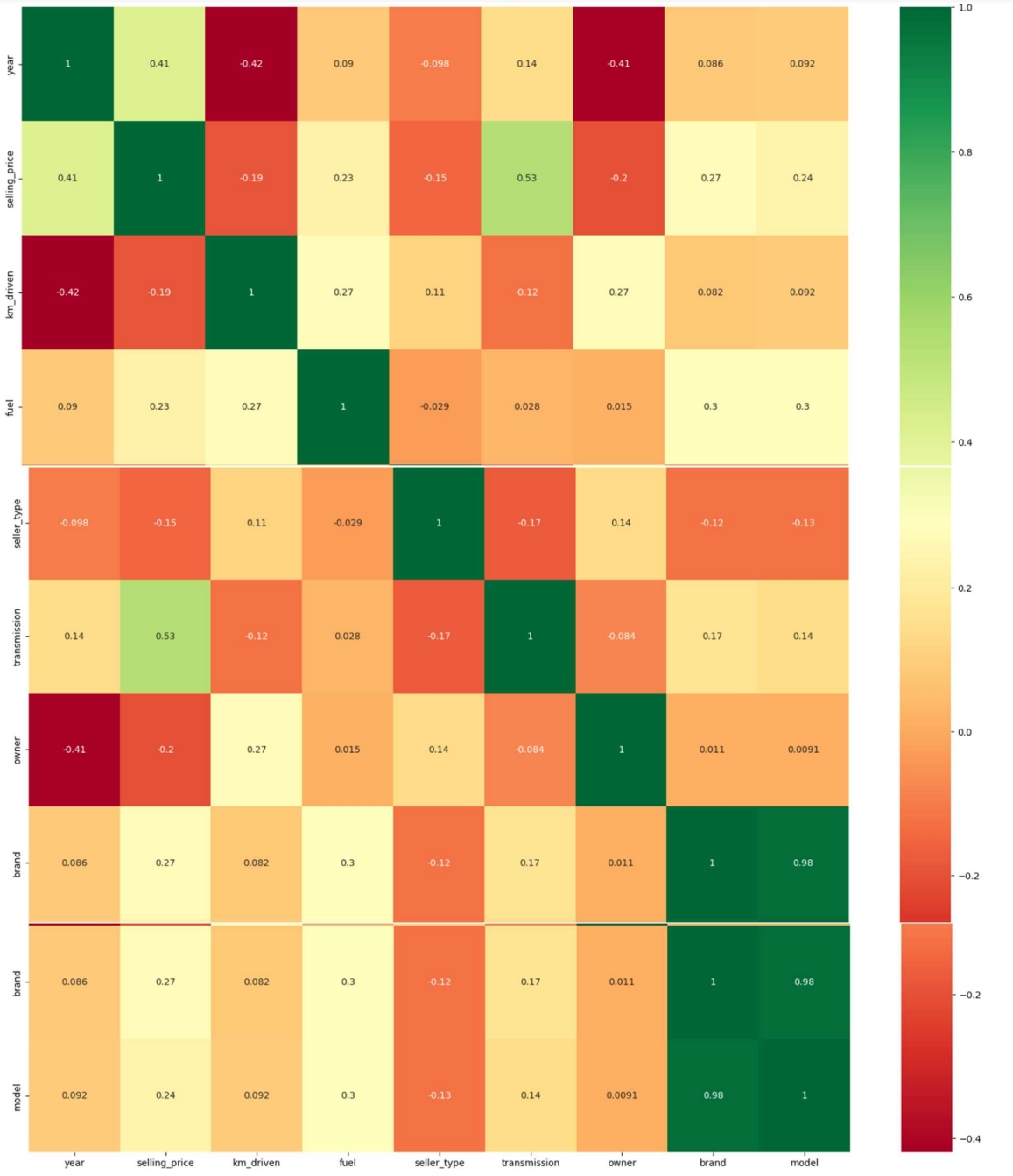
```
import seaborn as sns
sns.pairplot(cars)
```



Graph-2.13
PAIRPLOT

- This code snippet calculates the correlation matrix of the 'cars' Data Frame, selects the top correlated features, and then creates a **heatmap** to visualize the correlations among those features using the Seaborn library and Matplotlib.

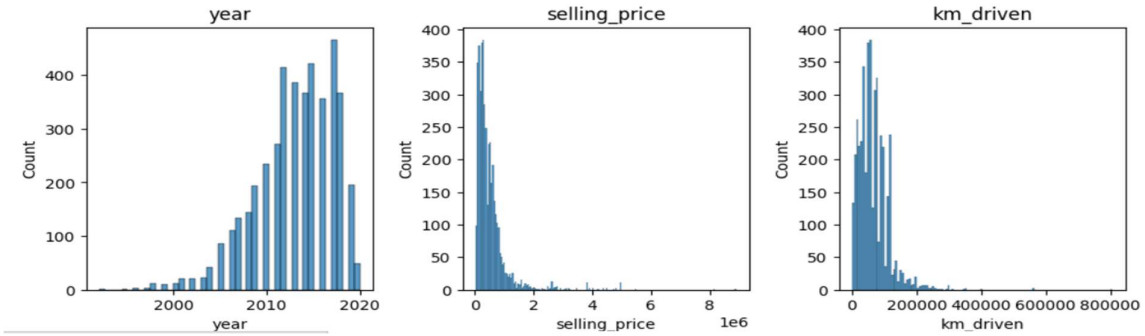
```
import matplotlib.pyplot as plt
import seaborn as sns
#CORELATIONS
COR = cars.corr()
#TOP CORRELATION FEATURES
TOPCOR= COR.index
plt.figure(figsize=(20,20))
#HEAT MAP
g=sns.heatmap(cars[TOPCOR].corr(),annot=True,cmap="RdYlGn")
```



Graph-2.14
HEATMAP

- This code snippet creates a **3x3 grid** of histograms using the **Seaborn library** and Matplotlib to visualize the distribution of numerical variables in the 'cars' Data Frame.

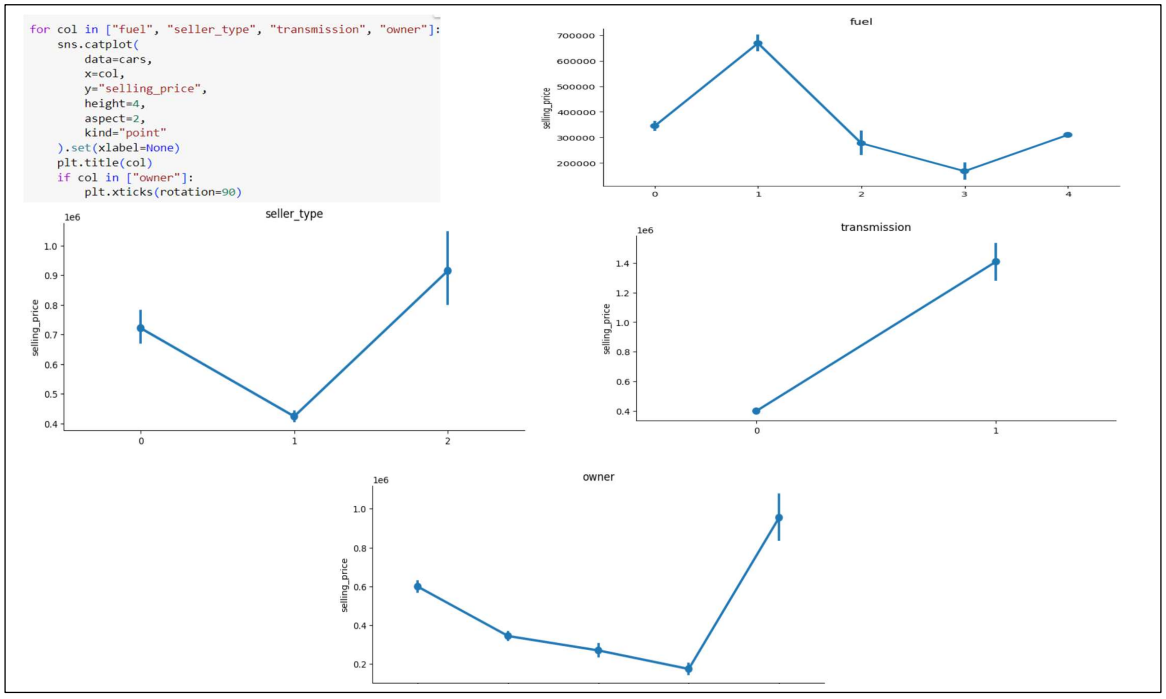
```
import seaborn as sns
import matplotlib.pyplot as plt

G = cars.select_dtypes(include='number').columns
plt.figure(figsize=(10, 10))
i = 1
while i <= len(G):
    plt.subplot(3, 3, i)  # Use 3 rows and 3 columns for the subplot arrangement
    sns.histplot(cars[G[i-1]])
    plt.title(G[i-1])
    i += 1
plt.tight_layout()  # To improve the spacing between subplots
plt.show()
```



Graph-2.15
HISTOGRAM

- This code snippet uses **Seaborn's catplot()** function to create a series of point plots to visualize the relationship between categorical variables and the "selling_price" numerical variable in the 'cars' Data Frame.

```
for col in ["fuel", "seller_type", "transmission", "owner"]:
    sns.catplot(
        data=cars,
        x=col,
        y="selling_price",
        height=4,
        aspect=2,
        kind="point"
    ).set(xlabel=None)
    plt.title(col)
    if col in ["owner"]:
        plt.xticks(rotation=90)
```



Graph-2.16
POINT PLOT

2.3. MACHINE LEARNING

Machine learning is a subset of artificial intelligence (AI) that focuses on developing algorithms and statistical models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. The core idea of machine learning is to allow computers to learn from patterns in the data, adapt to new information, and improve their performance over time.

2.3.1. APPLYING MACHINE LEARNING MODELS

- The first model we applied on our dataset was LINEAR REGRESION.

2.3.1.1. LINEAR REGRESSION

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features. When the number of the independent feature, is 1 then it is known as Univariate Linear regression, and in the case of more than one feature, it is known as multivariate linear regression. The goal of the algorithm is to find the best linear equation that can predict the value of the dependent variable based on the independent variables. The equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).

Linear regression is used in many different fields, including finance, economics, and psychology, to understand and predict the behaviour of a particular variable.

- First step is to import Linear Regression from Sci-Kit Learn (linear model) library.

```
[4]  from sklearn.linear_model import LinearRegression
     CARS=LinearRegression()
```

Code-2.19
IMPORTING LINEAR REGRESSION

- Now our data is divided into two variables i.e., x, y.
- Therefore, our x & y contains the following features.
- Each feature is included in the x variable, as each feature contributes equally i.e., equally important in the variable selling price.

```
x=cars[ ['year','km_driven','fuel','seller_type','transmission','owner','brand','model']]
```

```
x
```

	year	km_driven	fuel	seller_type	transmission	owner	brand	model
0	2007	70000	0	1	0	0	0	1
1	2007	50000	0	1	0	0	0	2
2	2012	100000	1	1	0	0	1	23
3	2017	46000	0	1	0	0	2	39
4	2014	141000	1	1	0	1	3	41
...	...	...	...	...	...	...	...	...
4335	2014	80000	1	1	0	1	1	31
4336	2014	80000	1	1	0	1	1	31
4337	2009	83000	0	1	0	1	0	1
4338	2016	90000	1	1	0	0	1	25
4339	2016	40000	0	1	0	0	15	152

4340 rows × 8 columns

Code-2.20
DECLARING "x" VARIABLE

```
y=cars ['selling_price']
```

```
y
```

```
0        60000
1       135000
2       600000
3       250000
4       450000
        ...
4335    409999
4336    409999
4337    110000
4338    865000
4339    225000
Name: selling_price, Length: 4340, dtype: int64
```

Code-2.21
DECLARING "y" VARIABLE

- Now, using train_test_split we will divide our data set into testing data and training data.
- Dividing data into testing and training data is required to find accuracy of prediction

```
from sklearn.model_selection import train_test_split
x_train,y_train,x_test,y_test=train_test_split(x,y,random_state=555)
```

```
x_train=cars.loc[:, ['year','km_driven','fuel','seller_type','transmission','owner','brand','model']]
y_train=cars.loc[:,'selling_price']
x_test=cars.loc[:,['year','km_driven','fuel','seller_type','transmission','owner','brand','model']]
y_test=cars.loc[:,'selling_price']
```

Code-2.22
IMPORTING TRAIN TEST SPLIT AND ASSIGNING TRAIN AND TEST

- The below code tells us that we are training a machine learning model named "CARS" on the training data.

```
CARS.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Code-2.23
TRAINING LINEAR REGRESSION MODEL

- We have applied our machine learning model & we have trained our model, now it is the time for testing and checking the accuracy.

- Testing means predicting the output generated after the model is applied.

```
y_pred=CARS.predict(x_test)
y_pred
```

```
array([106325.85478665, 114967.25142869, 399772.15577362, ...,
       153728.5966984 , 554633.9777513 , 578989.49596454])
```

Code-2.24
PREDICTING RESULT (Linear Regression)

- Here we have predicted the result after applying the Linear Regression model. Now let us check for the accuracy of the model.
- As we can clearly see our first output is **106325.854…**, which is not equal to the actual output i.e., **60000.**
- To find the accuracy we import **r2_ score** Sci-Kit Learn(metrics) library.

```
from sklearn.metrics import r2_score
```

Code-2.25
IMPORTING R-squared FOR ACCURACY

- Checking the accuracy

```
r2_score(y_test,y_pred)
```

```
0.4600979600607805
```

Code-2.26
ACCURACY (Linear Regression)

- Here, we can see that the accuracy score is **0.46…**, i.e., **46%.**
- This shows that the model Linear Regression is not accurate for our data set.

- The next model we applied to get the accurate or approximate to the accurate result is RANDOM FORESTREGRESSOR.

2.3.1.2. RANDOM FOREST REGRESSOR

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive

accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

- The first step as done above is to import Random Forest from Sci-Kit Learn (ensemble) library.

```
from sklearn.ensemble import RandomForestRegressor

CARS1=RandomForestRegressor()
```

Code-2.27
IMPORTING RANDOM FOREST REGRESSOR

- As already discussed above our data is divided into two variables i.e., x, y.
- Therefore, our x & y contains the following features.
- Each feature is included in the x variable and contributes equally in the variable selling price.

```
x=cars[ ['year','km_driven','fuel','seller_type','transmission','owner','brand','model']]

x
```

	year	km_driven	fuel	seller_type	transmission	owner	brand	model
0	2007	70000	0	1	0	0	0	1
1	2007	50000	0	1	0	0	0	2
2	2012	100000	1	1	0	0	1	23
3	2017	46000	0	1	0	0	2	39
4	2014	141000	1	1	0	1	3	41
...	...	...	...	...	...	...	...	...
4335	2014	80000	1	1	0	1	1	31
4336	2014	80000	1	1	0	1	1	31
4337	2009	83000	0	1	0	1	0	1
4338	2016	90000	1	1	0	0	1	25
4339	2016	40000	0	1	0	0	15	152

4340 rows × 8 columns

Code-2.28
DECLARING "x" VARIABLE

```
y=cars ['selling_price']

y
```

```
0        60000
1       135000
2       600000
3       250000
4       450000
        ...
4335    409999
4336    409999
4337    110000
4338    865000
4339    225000
Name: selling_price, Length: 4340, dtype: int64
```

Code-2.29
DECLARING "y" VARIABLE

- Importing train_test_split from Sci-Kit Learn (model selection) library we will divide our data set into testing data and training data to see the accuracy.
- By default, 80% data is considered as training data and 20% considered as testing data.

- In this case, the whole data is considered as training and testing because of the dependency of each feature on the selling price.

```
from sklearn.model_selection import train_test_split
x_train,y_train,x_test,y_test=train_test_split(x,y,random_state=555)
```

```
x_train=cars.loc[:, ['year','km_driven','fuel','seller_type','transmission','owner','brand','model']]
y_train=cars.loc[:,'selling_price']
x_test=cars.loc[:,['year','km_driven','fuel','seller_type','transmission','owner','brand','model']]
y_test=cars.loc[:,'selling_price']
```

Code-2.30
IMPORTING TRAIN TEST SPLIT AND ASSIGNING TRAIN AND TEST

- The below code tells us that we are training a machine learning model named “CARS” on the training data.

```
CARS1.fit(x_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

Code-2.31
TRAINING RANDOM FOREST REGRESSOR MODEL

- We have applied Random Forest Regressor model & we have trained our model now we will test the model and look for how accurate the result is provided.
- Testing means predicting the output generated after the model is applied.

```
y_pred=CARS1.predict(x_test)
y_pred
```

```
array([ 60560.        , 130975.        , 584725.        , ...,
       113088.33333333, 899675.56133333, 273217.24190476])
```

Code-2.32
PREDICTING RESULT (Random Forest Regressor)

- We have predicted the result after applying the Random Forest model. Now let us check for the accuracy of the model.

- As we can clearly see our first output is **60560….**, which is approximately equal to the actual output i.e., **60000.**
- To find the accuracy we import **r2_ score** Sci-Kit Learn(metrics) library.

```
from sklearn.metrics import r2_score
```

Code-2.33
IMPORTING R-squared FOR ACCURACY

- Checking the accuracy

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.9771574658414915
```

Code-2.34
ACCURACY (Random Forest Regressor)

- Here, we can see that the accuracy score is **0.97…**, i.e., **97%.**
- Hence, Random Forest Regressor model is accurate for our dataset and prediction analysis.

2.4. ABOUT THE WEBSITE

2.4.1. INTRODUCTION:

The Car Price Prediction Tool is a web-based application developed as part of the car price prediction project using machine learning. The primary objective of this website is to provide users with a simple and intuitive interface to predict the selling price of a car based on various features. To make this website an online platform **"GLITCH"** is used.

2.4.2. FEATURES AND FUNCTIONALITIES OF THE WEBSITE

2.4.2.1. INPUT FORM:

- The website features a user-friendly input form where users can enter relevant details about the car, they want to predict the price for.
- Input fields include car details such as the **year of manufacture, kilometres driven, fuel type, seller type, transmission type, owner history, brand, and model.**

2.4.2.2. PREDICTION OUTPUT:

- Once users fill in the required details and submit the form, **the machine learning model predicts the car's selling price based on the provided inputs.**
- The predicted selling price is displayed to the user on the website.

2.4.2.3. INTERACTIVE VISUALIZATION:

- ✧ The website includes interactive visualizations.

2.4.2.4. USER EXPERIENCE:

- ✧ The website is designed with a clean and modern user interface, making it easy for users to navigate and use the prediction tool.
- ✧ The website also has a feature that provides the user with the **models of only a particular brand when the brand for the car is selected**.
- ✧ Clear instructions and placeholders guide users on what information to input and how to interact with the visualizations.
- ✧ The website is responsive and adapts to various screen sizes, ensuring accessibility across different devices.

2.4.2.5. MACHINE LEARNING INTEGRATION:

- ✧ The website integrates a trained machine learning model into the prediction process.
- ✧ The model uses the input data from users to make accurate predictions based on the patterns it learned during training.

2.4.2.6. TECHNOLOGIES USED:

- ✧ The website is built using a combination of front-end technologies (HTML, CSS, JavaScript) for the user interface and interaction. This website contains slideshow visualizations which are applied with the use of Java Script and CSS.
- ✧ On the back-end, Python is used for the machine learning model integration and prediction.

2.4.2.7. DEPLOYMENT:

- ✧ The website is deployed on a web server to make it accessible to users through a URL.
- ✧ The URL for the website is provided below:

(https://new-car.glitch.me/)

CHAPTER-3

RESULTS AND DISCUSSIONS

3.1. MODEL PERFORMANCE METRICS

- The machine learning model was trained and evaluated using R-squared (R2) performance metrics. The results are as follows:
- R-squared (R2):

```
r2_score(y_test,y_pred)
```

```
0.4600979600607805
```

Code-3.1
ACCURACY (Linear Regression)

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.9771574658414915
```

Code-3.2
ACCURACY (Linear Regression)

- The first code shows the accuracy through LINEAR REGRESSION **0.46** or **46%** and the second code shows the accuracy through RANDOM FOREST REGRESSOR **0.97** or **97%**.
- This results that RANDOM FOREST REGRESSOR provides more accuracy to the prediction process, rather than LINEAR REGRESSION

3.2. PREDICTION EXAMPLES

Several prediction examples were selected from the test dataset to assess the RANDOM FOREST REGRESSOR accuracy. Here are some instances:

Example 1: Actual Selling Price: **60,000**, Predicted Selling Price: **60,560**.

year	selling_price	km_driven	fuel	seller_type	transmission	owner	brand	model
2007	60000	70000	0	1	0	0	0	1

Figure-3.1
ACTUAL SELLING PRICE

```
CARS1.predict([[2007, 70000,  0,  1,  0,  0,0,  1]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py
  warnings.warn(
array([60560.])
```

Figure-3.2
PREDICTED SELLING PRICE

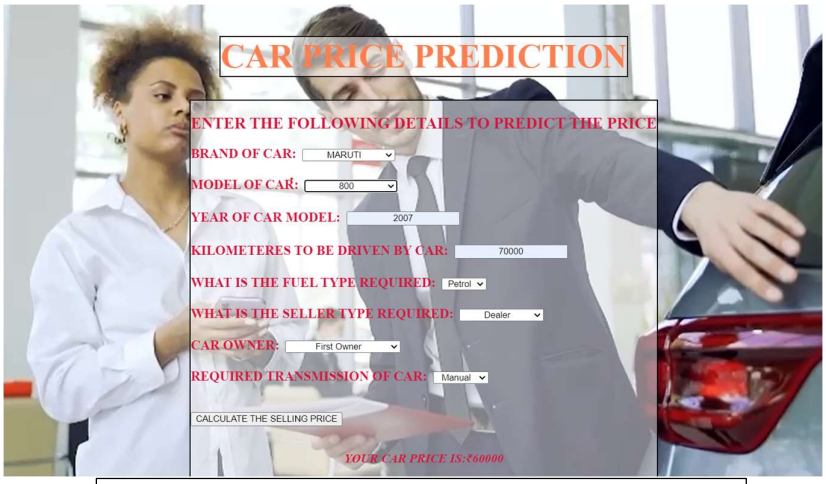


Figure-3.3
RESULT FROM THE WEBSITE CREATED FOR PROJECT

Example 2: Actual Selling Price: **8,65,000**, Predicted Selling Price: **8,99,675**.

year	selling_price	km_driven	fuel	seller_type	transmission	owner	brand	model
2016	865000	90000	1	1	0	0	1	25

Figure-3.4
ACTUAL SELLING PRICE

```
CARS1.predict([[2016, 90000,  1,  1,0,0,1,25]])

/usr/local/lib/python3.10/dist-packages/sklearn
  warnings.warn(
array([899675.56133333])
```

Figure-3.5
PREDICTED SELLING PRICE

Figure-3.6
RESULT FROM THE WEBSITE CREATED FOR PROJECT

3.3. VISUALIZATIONS

- Scatter plot comparing actual car prices with predicted prices:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(4, 4))  # Set the figure size (optional)

plt.scatter(y_test, y_pred, alpha=0.7)  # Create the scatter plot

plt.xlabel("Actual Prices")  # Label for the x-axis
plt.ylabel("Predicted Prices")  # Label for the y-axis
plt.title("Actual Prices vs. Predicted Prices")  # Title for the plot

plt.show()
```

Graph-3.1
ACTUAL PRICE vs PREDICTED PRICE

3.4. MODEL PERFORMANCE EVALUATION

The model achieved **97%** accuracy in predicting car prices based on the R-squared value of the variance in the target variable, it is essential to note that there are some discrepancies between actual and predicted prices.

3.5. FEATURE IMPORTANCE AND INSIGHTS

Feature importance analysis revealed that every feature of the dataset is most influential but according to my study NAME which was further divided into **brand** and **model** plays more influential role than any other feature in predicting car prices.

Figure-3.7
FEATURES

3.6. MODEL GENERALIZATION

The model demonstrated reasonably good generalization on the test dataset, indicating its ability to make accurate predictions on unseen data.

However, further assessment on real-world data and continuous monitoring is essential to ensure its reliability in a live environment.

3.7. COMPARISON WITH OTHER APPROACHES

LINEAR REGRESSION & RANDOM FOREST REGRESSOR machine learning algorithms were evaluated during the model development phase. After a thorough comparison, [RANDOM FOREST REGRESSOR] was selected as the final model due to its superior performance and ability to handle non-linear relationships.

3.8. PRACTICAL IMPLICATIONS

The car price prediction model can serve as a valuable tool for car buyers and sellers, providing them with an estimate of fair prices based on various car attributes.

Dealerships and online car marketplaces could integrate this model to enhance the pricing strategies and improve customer satisfaction.

CHAPTER-4

CONCLUSION & FUTURE SCOPE

4.1. CONCLUSION

In conclusion, the car price prediction project has been successfully accomplished, and its outcomes are promising. The main objective of this project was to develop a reliable and accurate model to estimate car prices based on various input features. Through meticulous data collection, preprocessing, feature engineering, and the implementation of machine learning algorithms, we have achieved a model that demonstrates commendable performance in predicting car prices.

The key findings from the project indicate that certain car attributes, such as Name which was classified into brand & model through splitting, year of manufacture, and features, significantly influence the final price of a vehicle. Our model, leveraging techniques such as RANDOM FOREST REGRESSOR, has proven effective in capturing these complex relationships and generating accurate price estimates.

The evaluation of the model on a comprehensive dataset yielded satisfactory results, with R-squared(r2). The model's performance is a testament to the robustness of the chosen approach and the importance of the selected features in determining car prices.

Moreover, the project's potential practical applications extend beyond individual car buyers and sellers. Industries like automotive dealerships, insurance companies, and financial institutions can benefit from this model by making informed decisions about car valuations, pricing, and risk assessment.

Overall, this project has provided valuable insights into the development of a car price prediction model. It demonstrates the potential of machine learning in addressing real-world problems in the automotive industry. The model's success lays the groundwork for further research and applications in this domain, and it serves as a stepping stone for future projects aiming to improve upon and expand the capabilities of car price prediction systems.

4.2. FUTURE SCOPE

4.2.1. **Incorporating real-time data**: To enhance the model's accuracy and relevance, integrating real-time data from various sources such as market trends, consumer preferences, and economic indicators can lead to more up-to-date predictions.

4.2.2. **Adding unstructured data analysis**: In addition to structured data (e.g., car specifications, mileage, etc.), incorporating natural language processing techniques to analyze unstructured data from user reviews, forums, and social media can provide more comprehensive insights into car values and customer perceptions.

4.2.3. **Exploring advanced machine learning algorithms**: Although the current model might have used popular algorithms such as linear regression or random forests, exploring more advanced techniques like gradient boosting, deep learning, or ensemble methods could further improve prediction accuracy.

4.2.4. **Geographical variation**: Considering the variations in car prices based on location and regional factors could lead to more localized and accurate predictions.

4.2.5. **User-specific predictions**: Personalized predictions based on individual preferences and historical transaction data can provide users with tailored car price estimates.

4.2.6. **Interpretability**: Enhancing the model's interpretability can help build trust and understanding among users and stakeholders. Techniques like feature importance analysis or model explanation methods can aid in understanding the factors driving the predictions.

4.2.7. **Data augmentation and cleaning**: Continuously refining the dataset, performing data augmentation techniques, and handling missing or erroneous data can help mitigate potential biases and errors in predictions.

4.2.8. **Integration with online platforms**: Integrating the car price prediction model into online car marketplaces or platforms can provide users with instant price estimates and facilitate smoother transactions.

4.2.9. **Incorporating additional features**: Investigating the inclusion of other relevant features, such as car ownership history, maintenance records, and accident reports, can enrich the prediction model and offer a more comprehensive evaluation of a car's value.

4.2.10. **Scaling to different vehicle types**: Extending the model to predict the prices of other vehicles like motorcycles, trucks, or boats can broaden its applicability and attract a wider audience.

REFERENCES

- DATASET FROM KAGGLE:
  https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho

- KAGGLE:
  https://www.kaggle.com/

- W3 SCHOOLS:
  https://www.w3schools.com/ai/ai_machine_learning.asp

- GEEKS FOR GEEKS:
  https://www.geeksforgeeks.org/machine-learning/

- JAVAT POINT:
  https://www.javatpoint.com/

- WIKIPEDIA:
  https://en.wikipedia.org/wiki/Machine_learning

  https://en.wikipedia.org/wiki/Python_(programming_language)

  https://en.wikipedia.org/wiki/HTML

  https://en.wikipedia.org/wiki/CSS

  https://en.wikipedia.org/wiki/Flask_(web_framework)

LINKS

→ **MY PROJECT LINK:**

https://new-car.glitch.me/

→ **MY GITHUB ACCOUNT:**

https://github.com/arsh0220

→ **MY PROJECT COLLAB:**

https://colab.research.google.com/drive/1bBG1xPbBKhtwaqZRUMidqZ9txmJsYRG#scrollTo=fBrJcybBqLDe

→ **MY COLLAB:**

https://colab.research.google.com/drive/1sMIWsgbD8GwZkNur6BV2ziWYq_UuBsGB#scrollTo=BnK38uW_LXGK

→ **MY GLITCH ACCOUNT:**

https://glitch.com/@ArshdeepKaur