**An Industry Oriented Mini Project Report**

**on**



**CAR PRICE ESTIMATION USING LINEAR REGRESSION**

*Submitted to the*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD**

*In partial fulfillment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**BY**

**MEHUL KATTELA**

**(17WJ1A05J4)**

**Under the Esteemed Guidance Of**

**DR. J. RAJESHWAR**

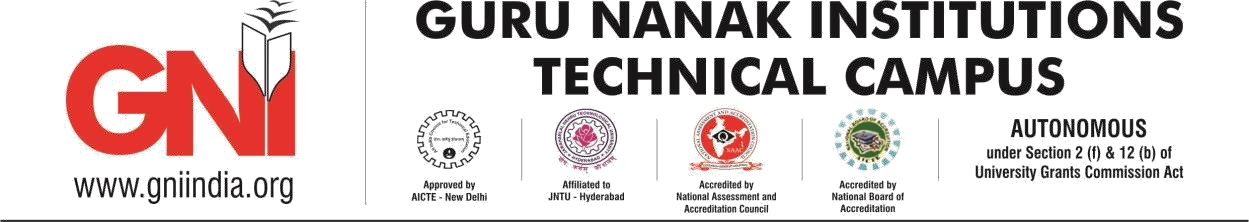
**Professor & HOD, CSE Department**

**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)**

**School of Engineering and Technology**

**Ibrahimpatnam R.R District 501506**

**2020-2021**

****

**CERTIFICATE**

This is to certify that An Industry Oriented Mini Project Report entitled **“CAR PRICE ESTIMATION USING LINEAR REGRESSION”** by **MEHUL KATTELA(17WJ1A05J4)** submittedin partial fulfillment for the award of the Degree of **Bachelor of Technology** in **Computer Science & Engineering** of the **Jawaharlal Nehru Technological University Hyderabad** during the academic year 2020-2021, is a bonafide record of work carried out under our guidance and supervision at **Guru Nanak Institutions Technical Campus**.

**INTERNAL GUIDE PROJECT CO-ORDINATOR HOD CSE**

**Dr.J.Rajeshwar Mr.A.Ravi Dr.J.Rajeshwar**

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

I wish to express my sincere thanks to **Dr. Rishi Sayal,** Associate Director, GNITC for providing me the conductive environment for carrying through our academic schedules and Project with ease.

I have been truly blessed to have a wonderful adviser **Dr. J. Rajeshwar,** HOD of CSE, GNITCfor guiding me to explore the ramification of my work and I express our sincere gratitude towards him for leading me thought the completion of Project.

I would like to say my sincere thanks to **Mr. A.RAVI**, Associate Professor, Department of CSE, Mini Project Coordinator, for providing seamless support and right suggestions are given in the development of the project.

I specially thank our internal guide **Dr. J. Rajeshwar,** Professor, Department of CSE, for his suggestions and constant guidance in every stage of the project.

MEHUL KATTELA (17WJ1A05J4)

TABLE OF CONTENTS

[ABSTRACT i](#_Toc65496277)

[LIST OF FIGURES ii](#_Toc65496278)

[LIST OF SYSMBOLS iv](#_Toc65496279)

[CHAPTER 1 1](#_Toc65496280)

[INTRODUCTION 1](#_Toc65496281)

[1.1 OVERVIEW 1](#_Toc65496282)

[1.2 PROBLEM STATEMENT 2](#_Toc65496283)

[1.3 PURPOSE 3](#_Toc65496284)

[1.4 SCOPE 4](#_Toc65496285)

[CHAPTER 2 5](#_Toc65496286)

[LITERATURE SURVEY 5](#_Toc65496287)

[2.1 REVIEW OF RELATED RESEARCH PAPERS 5](#_Toc65496288)

[CHAPTER 3 10](#_Toc65496289)

[SYSTEM REQUIREMENTS SPECIFICATIONS 10](#_Toc65496290)

[3.1 EXISTING SYSTEM 10](#_Toc65496291)

[3.2 PROPOSED SYSTEM 10](#_Toc65496292)

[3.3 REQUIREMENTS 11](#_Toc65496293)

[3.3.1 HARDWARE REQUIREMENTS 11](#_Toc65496294)

[3.3.2 SOFTWARE REQUIREMENTS 12](#_Toc65496295)

[3.4 FEASIBILITY ANALYSIS 12](#_Toc65496296)

[3.4.1 TECHNICAL FEASIBILITY 12](#_Toc65496297)

[3.4.2 OPERATION FEASIBILITY 13](#_Toc65496298)

[3.4.3 ECONOMIC FEASIBILITY 13](#_Toc65496299)

[3.5 FUNCTIONAL REQUIREMENTS 13](#_Toc65496300)

[3.6 NON-FUNCTIONAL REQUIREMENTS 13](#_Toc65496301)

[CHAPTER 4 14](#_Toc65496302)

[SYSTEM ANALYSIS AND DESIGN 14](#_Toc65496303)

[4.1 DESIGNING PART 14](#_Toc65496304)

[4.2 MODULES AND THEIR DESCRIPTION 16](#_Toc65496305)

[4.2.1 DATA COLLECTION 17](#_Toc65496306)

[4.2.2 DATA PRE-PROCESSING 17](#_Toc65496307)

[4.2.3 MODEL BUILDING 17](#_Toc65496308)

[4.2.4 EVALUATION 17](#_Toc65496309)

[4.3 SYSTEM ARCHITECTURE: 18](#_Toc65496310)

[4.4 DATA FLOW DIAGRAM: 19](#_Toc65496311)

[4.5 UML DIAGRAMS: 19](#_Toc65496312)

[4.5.1 GOALS: 20](#_Toc65496313)

[4.5.2 USE CASE DIAGRAM: 21](#_Toc65496314)

[4.5.3 CLASS DIAGRAM: 22](#_Toc65496315)

[4.5.4 SEQUENCE DIAGRAM: 23](#_Toc65496316)

[CHAPTER 5 24](#_Toc65496317)

[IMPLEMENTATION 24](#_Toc65496318)

[5.1 TECHNOLOGY USED 24](#_Toc65496319)

[5.1.1 PYTHON 24](#_Toc65496320)

[5.2 SAMPLE CODE 27](#_Toc65496321)

[5.3 SCREENSHOT 52](#_Toc65496322)

[5.3.1 Analyzing symbols and count of cars with the help of bar graph. 52](#_Toc65496323)

[5.3.2 Analyzing symbols and count of cars with the help of pie chart. 52](#_Toc65496324)

[5.3.3 Bar graph with symbol on x-axis and avg price(dollars) on y-axis. 53](#_Toc65496325)

[5.3.4 Bar graph with no of doors on x-axis and avg price(dollars) on y-axis. 53](#_Toc65496326)

[5.3.5 Bar graph with brand on x-axis and count of cars on y-axis 54](#_Toc65496328)

[5.3.6 Bar graph with car body on x-axis and avg price(dollars) on y-axis. 54](#_Toc65496329)

[5.3.7 Bar graph with fuel type on x-axis and avg price(dollars) on y-axis 55](#_Toc65496330)

[5.3.8 Bar graph with drive wheel type on x-axis and avg price(dollars) on y-axis 55](#_Toc65496331)

[5.3.9 Plotting length and width of car(inches) on x-axis, and price(dollars) on y-axis 56](#_Toc65496333)

[5.3.10 height of car(inches) & weight of car(pounds) on x-axis with price on y-axis. 56](#_Toc65496334)

[5.3.11 plotting mileage on x-axis and price of car(dollars) on y-axis 56](#_Toc65496335)

[5.3.12 Plotting mileage of three brand categories on x-axis and price of car on y-axis. 57](#_Toc65496336)

[5.3.13 Accuracy of the model. 58](#_Toc65496337)

[CHAPTER 6 59](#_Toc65496338)

[SOFTWARE TESTING 59](#_Toc65496339)

[6.1 DEFINATION 59](#_Toc65496340)

[6.2 SOFTWARE TESTING 59](#_Toc65496341)

[6.3 BLACK BOX TESTING 59](#_Toc65496342)

[6.4 WHITE BOX TESTING 59](#_Toc65496343)

[6.5 SOFTWARE TESTING STRATEGIES 60](#_Toc65496344)

[6.6 TYPES OF TESTING 60](#_Toc65496345)

[6.6.1 UNIT TESTING 60](#_Toc65496346)

[6.6.2 FUNCTIONAL TESTING 60](#_Toc65496347)

[6.6.3 SYSTEM TESTING 61](#_Toc65496348)

[6.6.4 PERFORMANCE TESTING 61](#_Toc65496349)

[6.6.5 INTEGRATION TESTING 61](#_Toc65496350)

[6.6.6 ACCEPTANCE TESTING 61](#_Toc65496351)

[6.7 BUILD THE TEST PLAN 61](#_Toc65496352)

[CHAPTER 7 62](#_Toc65496353)

[CONCLUSION AND FUTURE ENHANCEMENT 62](#_Toc65496354)

[7.1 CONCLUSION 62](#_Toc65496355)

[7.2 FUTURE ENHANCEMENT 63](#_Toc65496356)

[REFERENCES 64](#_Toc65496357)

# ABSTRACT

A car price prediction has been a high interest research area, as it requires noticeable effort and knowledge of the field expert. Considerable number of distinct attributes are examined for the reliable and accurate prediction. To build a model for predicting the price of used cars in Bosnia and Herzegovina, we applied three machine learning techniques (Artificial Neural Network, Support Vector Machine and Random Forest). However, the mentioned techniques were applied to work as an ensemble. The data used for the prediction was collected from the web portal autopijaca.ba using web scraper that was written in PHP programming language. Respective performances of different algorithms were then compared to find one that best suits the available data set. The final prediction model was integrated into Java application. Furthermore, the model was evaluated using test data and the accuracy of 87.38% was obtained.

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.2.1 | Use Case Diagram | 17 |
| 4.2.2 | Class Diagram | 18 |
| 4.2.3 | Object Diagram | 19 |
| 4.2.4 | State Diagram | 20 |
| 4.2.5 | Activity Diagram | 21 |
| 4.2.6 | Sequence diagram | 22 |
| 4.2.7 | Collaboration diagram | 23 |
| 4.2.8 | Component Diagram | 24 |
| 4.2.9 | Data Flow Diagram | 25-26 |
| 4.2.10 | Deployment Diagram | 27 |
| 4.2.11 | System Architecture | 28 |
| 7.1 | Analyzing symbols and count of cars with the help of bar graph | 57 |
| 7.2 | Analyzing symbols and count of cars with the help of pie chart | 57 |
| 7.3 | Bar graph with symbol on x-axis and avg price(dollars) on y-axis | 58 |
| 7.4 | Bar graph with no of doors on x-axis and avg price(dollars) on y-axis. | 58 |
| 7.5 | Bar graph with brand on x-axis and count of cars on y-axis. | 59 |
| 7.6 | Bar graph with car body on x-axis and avg price(dollars) on y-axis. | 59 |
| 7.7 | Bar graph with fuel type on x-axis and avg price(dollars) on y-axis. | 60 |
| 7.8 | Bar graph with drive wheel type on x-axis and avg price(dollars) on y-axis. | 60 |
| 7.9 | Plotting length and width of car(inches) on x-axis with price of car(dollars) on y-axis. | 61 |
| 7.10 | plotting height of car(inches) and weight of car(pounds) on x-axis with price of car(dollars) on y-axis. | 61 |
| 7.11 | Plotting Mileage on x-axis and Price of Car(Dollars) on y-axis. | 62 |
| 7.12 | Plotting mileage of three types of brand categories on x-axis and price of car(dollars) on y-axis. | 62 |
| 7.13 | Accuracy of the model. | 63 |

# LIST OF SYSMBOLS

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processes. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW

Car price estimation is somehow interesting and popular problem. As per information that was gotten from the Agency for Statistics of BiH, 921.456 vehicles were registered in 2014 from which 84% of them are cars for personal usage. This number is increased by 2.7% since 2013 and it is likely that this trend will continue, and the number of cars will increase in future. This adds additional significance to the problem of the car price prediction.

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

## 1.2 PROBLEM STATEMENT

Vehicle value forecast is by one way or another fascinating and famous issue. According to data that was gotten from the Organization for Measurements of BiH, 921.456 vehicles were enlisted in 2014 from which 84% of them are vehicles for individual use. This number is expanded by 2.7% since 2013 and all things considered, this pattern will proceed, and the quantity of vehicles will increment in future. This adds extra importance to the issue of the vehicle value expectation.

## 1.3 PURPOSE

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

## 1.4 SCOPE

Albeit, this framework has accomplished bewildering execution in vehicle value forecast issue our focus on the future examination is to test this framework to work effectively with different informational collections. We will expand our test information with eBay and OLX utilized vehicles informational indexes and approve the proposed approach.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 REVIEW OF RELATED RESEARCH PAPERS

**Title:** Data Mining with Open-Source Machine Learning Software in Java.

**Year: 2018**

**Author:** waikato.ac.

**Description:** here are two versions of Weka: Weka 3.8 is the latest stable version and Weka 3.9 is the development version. New releases of these two versions are normally made once or twice a year. For the bleeding edge, it is also possible to download nightly snapshots of these two versions.

The stable version receives only bug fixes and feature upgrades that do not break compatibility with its earlier releases, while the development version may receive new features that break compatibility with its earlier releases.

Weka 3.8 and 3.9 feature a package management system that makes it easy for the Weka community to add new functionality to Weka. The package management system requires an internet connection in order to download and install packages.

**Title:** Predicting the price of used cars using machine learning techniques.

**Author:** Pudaruth

**Year:2017**

**Description:** According to a World Economic Forum’s report, AI-enabled automation will generate 133 million new jobs globally by 2022. And in India itself, the demand for AI talent pool is expected to skyrocket with the government’s steps towards digitization, and multiple organizations accelerating their digital transformation initiatives. Are you ready to ride the wave? The BITS Pilani 11-month online PG Programme in AI & ML is designed to help working professionals like you to develop an understanding of AI & ML and its various building blocks.

**Title:** An expert system of price forecasting for used cars using adaptive neurofuzzy inference.

**Year: 2009**

**Author:** Wu, J. D., Hsu, C. C., & Chen, H. C.

**Description:** An expert system for used cars price forecasting using adaptive neuro-fuzzy inference system (ANFIS) is presented in this paper. The proposed system consists of three parts: [data acquisition system](https://www.sciencedirect.com/topics/engineering/data-acquisition-system), price forecasting algorithm and performance analysis. The effective factors in the present system for price forecasting are simply assumed as the mark of the car, manufacturing year and engine style. Further, the equipment of the car is considered to raise the performance of price forecasting. In price forecasting, to verify the effect of the proposed ANFIS, a conventional [artificial neural network](https://www.sciencedirect.com/topics/computer-science/artificial-neural-network) (ANN) with [back-propagation](https://www.sciencedirect.com/topics/engineering/backpropagation) (BP) network is compared with proposed ANFIS for [price forecast](https://www.sciencedirect.com/topics/engineering/price-forecast) because of its adaptive learning capability. The ANFIS includes both fuzzy logic qualitative approximation and the adaptive neural network capability. The experimental result pointed out that the proposed expert system using ANFIS has more possibilities in used car price forecasting.

**Title:** New Model for Residual Value Prediction of the Used Car Based on BP Neural Network and Nonlinear Curve Fit. In Measuring Technology and Mechatronics Automation

**Year: 2011**

**Author:** Gongqi, S., Yansong, W., & Qiang, Z.

**Description:** A new model for predicting the residual value of the private used car with various conditions, such as manufacturer, mileage, time of life, etc., was developed in this paper. A comprehensive method combined by the BP neural network and nonlinear curve fit was introduced for optimizing the model due to its flexible nonlinearity. Firstly, some distribution curves of residual value of the used cars were analyzed in time domain. Then, the BP neural network (NN) was established and used to extract the feature of the distribution curves in various conditions. A set of schemed data was used to train the NN and reached the training goal. Finally, the schemed data as inputs and the NN outputs were organized for nonlinear curve fit. Conclusion was drawn that the newly proposed model is feasible and accurate for residual value prediction of the used cars with various conditions.

**Title:** PIN Optimal Distribution of Auction Vehicles System: Applying Price Forecasting, Elasticity Estimation, and Genetic Algorithms to Used-Vehicle Distribution. Marketing Science

**Year:2009**

**Author:** Du, J., Xie, L., & Schroeder

**Description:** In addition to retailing new vehicles, automotive manufacturers in the United States sell millions of vehicles through leasing and to fleet customers every year. The majority of these vehicles are returned to the automotive manufacturers at the end of the contracted term and must be “remarketed.” In 2007, about 10 million used vehicles were sold at more than 400 auctions in the United States. Large consigners face decisions every day about when, where, and at what price to offer these vehicles, which has significant financial implications for their profitability. To address the challenges of the distribution process, (), a division of J.D. Power and Associates, developed the PIN Optimal Distribution of Auction Vehicles System (ODAV), an automated decision optimization system that helps remarketers maximize profits through the most advantageous distribution of their auction vehicles. At the core of the system is a combination of three models that determine the distribution of the vehicles on a daily basis: a nearest neighbor linear regression model for short-term auction price forecasting; an autoregressive integrated moving average time-series analysis model for volume-price elasticity; and a genetic algorithm optimizer for vehicle distribution. Since its launch in 2003, PIN has been providing ODAV services on a daily basis, and to date, more than two million vehicles have been distributed through this system. In this paper, we will describe the PIN ODAV System, its implementation, and the business impact by using as an example the experience with our largest client, Chrysler Group LLC.

# CHAPTER 3

# SYSTEM REQUIREMENTS SPECIFICATIONS

## 3.1 EXISTING SYSTEM

In existing system, they are implementing car price prediction system using k-means algorithm.

Existing System Disadvantages:

* Less accuracy
* Difficult to predict
* With global clusters, it doesn’t work well

## 3.2 PROPOSED SYSTEM

In this system we are implementing effective car price prediction system using linear regression. We can give the input as in CSV file or manual entry to the system. After taking input the algorithms applied. After accessing data set the operation is performed and effective results are obtained

Proposed System Advantages:

* Easy to predict
* High accuracy rate

## 3.3 REQUIREMENTS

**GENERAL**

These are the requirements for doing the project. Without using these tools and software’s we can’t do the project. So, we have two requirements to do the project. They are

1. Hardware Requirements.

2. Software Requirements.

### 3.3.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

### 3.3.2 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* OPERATING SYSTEM : WINDOWS 10
* PLATFORM : JUPYTER NOTEBOOK
* PROGRAMMING LANGUAGE : PYTHON

## 3.4 FEASIBILITY ANALYSIS

Feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system to be carried out. This is to ensure that the purpose of the system is not burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential

### 3.4.1 TECHNICAL FEASIBILITY

This study is carried out to check the feasibility that is the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the clients. The developed system must have modest requirements as only minimal or null ranges are required for implementing this system.

### 3.4.2 OPERATION FEASIBILITY

The operation staff in the organization feasibility. The employees of the concerned organization are supportive enough to implement the proposed system. Hence, it is operationally feasible. Therefore, the proposed system is feasible in all aspects. Hence, it is encouraging to undertaking a detailed system analysis.

### 3.4.3 ECONOMIC FEASIBILITY

The study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is omitted. The expenditure must be justified. Thus, the development system as well as within the budget and this was achieved because most of the technologies used are freely available. Only customized product to be purchased.

## 3.5 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs. Our system requires minimum three systems to achieve this concept.

## 3.6 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviour. They are contrasted with functional requirements that define specific behaviour or functions.

# CHAPTER 4

# SYSTEM ANALYSIS AND DESIGN

## 4.1 DESIGNING PART

The act, process, or profession of studying an activity (such as a procedure, a business, or a physiological function) typically by mathematical means in order to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently.

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product. The concept of design as a way of making sense of things has been the subject of many studies as has the design process itself. Since “design” can be used to express intention as opposed to the actual materials, forms, processes and markets, it is often used to describe the driving force of the creative thought itself. When attempting to characterize the major movements which operate within operate within the world of design today, three in particular seem to each be characterized by specific discourses and values and to be practiced by large numbers of designers and other professionals. Technology driven design, sustainable design and human created design are major movements which usually lead to distinguishably different results despite operating within the same legal, regulatory, contextual and economic constraints. Human centred design has its roots in semi-scientific fields such as ergonomics, computer science and artificial intelligence. The toolbox of human centred design techniques grows continuously, sometimes by borrowing from fields such as psychology or sociology and sometimes by defining new approaches which emerge from design practice. The most basic form of tool consists of facts about people such as anthropometric, biomechanical, cognitive, emotional, psychophysical, psychological and sociological data and model. This design is also well aligned with the corporate branding frameworks which many businesses use to prevent themselves to the world and to position themselves with respect to their competitors.

**• Design Process:**

Planning an item is a difficult cycle when the object is to make a joy and a persuading experience for the client. An all around planned and a reasonable interaction is an answer for all the cases and it will eliminate all disarray and questions. Recording the plan interaction would be useful in giving an expected conveyance season of the item and the necessary exertion for the undertaking. Client Experience Configuration is the way toward upgrading client fulfillment with an item by improving the convenience, availability and delight gave in the cooperation the item. In the process we experience various stages consistently. Each stage includes pertinent partners in the association that partake during the time spent making items profoundly effective and usable.

The design process involves the following 6 stages:

1. **Understand:**

Before beginning the design work, let the design team understand the requirements clearly. Outcomes of this stage are User Personas, User Stories, Use Cases, user Flows.

1. **Research:**

Design team does their research work to explore how the outer world is working on the particular feature. Outcomes of this stage are a bunch of ideas and material on which we can build the actual design work.

1. **Sketch:**

This stage involves UI definition of required feature. Designing is not something that we just create and start using it. Draw and draft and redraw and redraft, thus creating an unmatched experience. Outcomes of this stage are sketches, wireframes, Mockups, User Flows.

1. **Design:**

Turn the initial mock ups and wireframes to great-looking images with theme and styles applied to them. Outcomes of this stage are Design images, detailed design specs like colours, theme, styles, guidelines, Icons.

1. **Implement:**

Development team builds backend functionality first and connects it with UI when they get design artifacts. While implementing, it is possible to raise the need of minor changes in design. Outcome of this stage is developed UI with complete functionality and experience following the designed theme and style.

1. **Evaluate:**

When products features are implemented, the end product are evaluated based on few factors like whether the system is usable or not, is it easy to use for end user. The outcomes of this stage are User feedback, UI audit reports, areas marked where improvement is required. After this stage, the process will iterate itself and depending on the required changes, you may go to stage 2, 3 or 4. The process goes on until the desired experience and customer satisfaction is achieved.

## 4.2 MODULES AND THEIR DESCRIPTION

**MODULE:**

* **Data Collection**
* **Data Pre-processing**
* **Model Building**
* **Evaluation**

**MODULE DESCRIPTION**

### 4.2.1 DATA COLLECTION

To build and develop Machine Learning models, you must first acquire the relevant dataset. This dataset will be comprised of data gathered from multiple and disparate sources which are then combined in a proper format to form a dataset. Dataset formats differ according to use cases. For instance, a business dataset will be entirely different from a medical dataset. While a business dataset will contain relevant industry and business data, a medical dataset will include healthcare-related data.

### 4.2.2 DATA PRE-PROCESSING

In data pre-processing, it is pivotal to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data. Needless to say, this will hamper your ML project.

### 4.2.3 MODEL BUILDING

The model building process involves setting up ways of collecting data, understanding and paying attention to what is important in the data to answer the questions you are asking, finding a statistical, mathematical or a simulation model to gain understanding and make predictions, In these we use 80% of data for training and remaining 20 % of data for testing

### 4.2.4 EVALUATION

The algorithm was applied on the whole dataset, to test how accurately the classifier can categorize samples into cheap, moderate and expensive car classes. It is a meta estimator that fits a number of decisions on various subsamples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

## 

## 4.3 SYSTEM ARCHITECTURE:

Data

collection

Data

preprocessing

Categorization

with

linear

regression

cheap

moderate

expensive

Prediction

**System Architecture**

## 4.4 DATA FLOW DIAGRAM:

* The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out this data, and the output data is generated by this system.
* The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
* DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow the transformations that are applied as data moves from input to output.
* DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

## 4.5 UML DIAGRAMS:

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects

### 4.5.1 GOALS:

### 

The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Encourage the growth of OO tools market.
* Provide a formal basis for understanding the modelling language.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

### 4.5.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted



### 4.5.3 CLASS DIAGRAM:

****

### 4.5.4 SEQUENCE DIAGRAM:



# CHAPTER 5

# IMPLEMENTATION

## 5.1 TECHNOLOGY USED

### 5.1.1 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Importance of Python**

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**Features of Python**

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Libraries used in python:**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure: NumPy, Pandas, Matplotlib, Scikit-learn

## 

## 5.2 SAMPLE CODE

#!/usr/bin/env python

# coding: utf-8

# In[1]:

import warnings

warnings.filterwarnings('ignore')

# In[2]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

get\_ipython().run\_line\_magic('matplotlib', 'inline')

from matplotlib.pyplot import xticks

# In[3]:

df = pd.DataFrame(pd.read\_csv("CarPrice\_Assignment.csv"))

# In[4]:

df.head()

# In[5]:

df.shape

# In[6]:

df.describe()

# In[7]:

df.info()

# In[8]:

df.columns

# # Data Cleaning

# In[9]:

sum(df.duplicated(subset = 'car\_ID')) == 0

# In[10]:

df.isnull().sum()\*100/df.shape[0]

# In[11]:

df.price.describe()

# In[12]:

sns.distplot(df['price'])

# In[13]:

plt1 = sns.countplot(df['symboling'])

plt1.set(xlabel = 'Symbol', ylabel= 'Count of Cars')

plt.show()

plt.tight\_layout()

# In[14]:

df\_sym = pd.DataFrame(df['symboling'].value\_counts())

df\_sym.plot.pie(subplots=True,labels = df\_sym.index.values, autopct='%1.1f%%', figsize = (15,7.5))

# Unsquish the pie.

plt.gca().set\_aspect('equal')

plt.show()

plt.tight\_layout()

# In[15]:

plt1 = df[['symboling','price']].groupby("symboling").mean().plot(kind='bar',legend = False,)

plt1.set\_xlabel("Symbol")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# In[16]:

df.CarName.values[0:10]

# In[17]:

df['brand'] = df.CarName.str.split(' ').str.get(0).str.upper()

# In[18]:

len(set(df.brand.values))

# In[19]:

fig, ax = plt.subplots(figsize = (15,5))

plt1 = sns.countplot(df['brand'], order=pd.value\_counts(df['brand']).index,)

plt1.set(xlabel = 'Brand', ylabel= 'Count of Cars')

xticks(rotation = 90)

plt.show()

plt.tight\_layout()

# In[20]:

df['brand'] = df['brand'].replace(['VW', 'VOKSWAGEN'], 'VOLKSWAGEN')

df['brand'] = df['brand'].replace(['MAXDA'], 'MAZDA')

df['brand'] = df['brand'].replace(['PORCSHCE'], 'PORSCHE')

df['brand'] = df['brand'].replace(['TOYOUTA'], 'TOYOTA')

# In[21]:

fig, ax = plt.subplots(figsize = (15,5))

plt1 = sns.countplot(df['brand'], order=pd.value\_counts(df['brand']).index,)

plt1.set(xlabel = 'Brand', ylabel= 'Count of Cars')

xticks(rotation = 90)

plt.show()

plt.tight\_layout()

# In[22]:

df.brand.describe()

# In[23]:

df\_comp\_avg\_price = df[['brand','price']].groupby("brand", as\_index = False).mean().rename(columns={'price':'brand\_avg\_price'})

plt1 = df\_comp\_avg\_price.plot(x = 'brand', kind='bar',legend = False, sort\_columns = True, figsize = (15,3))

plt1.set\_xlabel("Brand")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 90)

plt.show()

# In[24]:

df = df.merge(df\_comp\_avg\_price, on = 'brand')

# In[25]:

df['brand\_category'] = df['brand\_avg\_price'].apply(lambda x : "Budget" if x < 10000

else ("Mid\_Range" if 10000 <= x < 20000

else "Luxury"))

# ### Fuel Type

# In[26]:

df\_fuel\_avg\_price = df[['fueltype','price']].groupby("fueltype", as\_index = False).mean().rename(columns={'price':'fuel\_avg\_price'})

plt1 = df\_fuel\_avg\_price.plot(x = 'fueltype', kind='bar',legend = False, sort\_columns = True)

plt1.set\_xlabel("Fuel Type")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# #### Aspiration

# In[27]:

df\_aspir\_avg\_price = df[['aspiration','price']].groupby("aspiration", as\_index = False).mean().rename(columns={'price':'aspir\_avg\_price'})

plt1 = df\_aspir\_avg\_price.plot(x = 'aspiration', kind='bar',legend = False, sort\_columns = True)

plt1.set\_xlabel("Aspiration")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# #### Door Numbers

# In[28]:

df\_door\_avg\_price = df[['doornumber','price']].groupby("doornumber", as\_index = False).mean().rename(columns={'price':'door\_avg\_price'})

plt1 = df\_door\_avg\_price.plot(x = 'doornumber', kind='bar',legend = False, sort\_columns = True)

plt1.set\_xlabel("No of Doors")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# #### Car Body

# In[29]:

df\_body\_avg\_price = df[['carbody','price']].groupby("carbody", as\_index = False).mean().rename(columns={'price':'carbody\_avg\_price'})

plt1 = df\_body\_avg\_price.plot(x = 'carbody', kind='bar',legend = False, sort\_columns = True)

plt1.set\_xlabel("Car Body")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# #### Drivewheel

# In[30]:

df\_drivewheel\_avg\_price = df[['drivewheel','price']].groupby("drivewheel", as\_index = False).mean().rename(columns={'price':'drivewheel\_avg\_price'})

plt1 = df\_drivewheel\_avg\_price.plot(x = 'drivewheel', kind='bar', sort\_columns = True,legend = False,)

plt1.set\_xlabel("Drive Wheel Type")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# #### Wheel base

# In[31]:

plt1 = sns.scatterplot(x = 'wheelbase', y = 'price', data = df)

plt1.set\_xlabel('Wheelbase (Inches)')

plt1.set\_ylabel('Price of Car (Dollars)')

plt.show()

# ## Car Dimensions

# In[32]:

fig, axs = plt.subplots(2,2,figsize=(15,10))

plt1 = sns.scatterplot(x = 'carlength', y = 'price', data = df, ax = axs[0,0])

plt1.set\_xlabel('Length of Car (Inches)')

plt1.set\_ylabel('Price of Car (Dollars)')

plt2 = sns.scatterplot(x = 'carwidth', y = 'price', data = df, ax = axs[0,1])

plt2.set\_xlabel('Width of Car (Inches)')

plt2.set\_ylabel('Price of Car (Dollars)')

plt3 = sns.scatterplot(x = 'carheight', y = 'price', data = df, ax = axs[1,0])

plt3.set\_xlabel('Height of Car (Inches)')

plt3.set\_ylabel('Price of Car (Dollars)')

plt3 = sns.scatterplot(x = 'curbweight', y = 'price', data = df, ax = axs[1,1])

plt3.set\_xlabel('Weight of Car (Pounds)')

plt3.set\_ylabel('Price of Car (Dollars)')

plt.tight\_layout()

# ## Engine Specifications

# #### Engine Type, Cylinder, Fuel System

# In[33]:

fig, axs = plt.subplots(1,3,figsize=(20,5))

#

df\_engine\_avg\_price = df[['enginetype','price']].groupby("enginetype", as\_index = False).mean().rename(columns={'price':'engine\_avg\_price'})

plt1 = df\_engine\_avg\_price.plot(x = 'enginetype', kind='bar', sort\_columns = True, legend = False, ax = axs[0])

plt1.set\_xlabel("Engine Type")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

#

df\_cylindernumber\_avg\_price = df[['cylindernumber','price']].groupby("cylindernumber", as\_index = False).mean().rename(columns={'price':'cylindernumber\_avg\_price'})

plt1 = df\_cylindernumber\_avg\_price.plot(x = 'cylindernumber', kind='bar', sort\_columns = True,legend = False, ax = axs[1])

plt1.set\_xlabel("Cylinder Number")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

#

df\_fuelsystem\_avg\_price = df[['fuelsystem','price']].groupby("fuelsystem", as\_index = False).mean().rename(columns={'price':'fuelsystem\_avg\_price'})

plt1 = df\_fuelsystem\_avg\_price.plot(x = 'fuelsystem', kind='bar', sort\_columns = True,legend = False, ax = axs[2])

plt1.set\_xlabel("Fuel System")

plt1.set\_ylabel("Avg Price (Dollars)")

xticks(rotation = 0)

plt.show()

# ### Engine Size, Bore Ratio, Stroke, Horsepower & Compression Ratio

# In[34]:

fig, axs = plt.subplots(3,2,figsize=(20,20))

#

plt1 = sns.scatterplot(x = 'enginesize', y = 'price', data = df, ax = axs[0,0])

plt1.set\_xlabel('Size of Engine (Cubic Inches)')

plt1.set\_ylabel('Price of Car (Dollars)')

#

plt2 = sns.scatterplot(x = 'boreratio', y = 'price', data = df, ax = axs[0,1])

plt2.set\_xlabel('Bore Ratio')

plt2.set\_ylabel('Price of Car (Dollars)')

#

plt3 = sns.scatterplot(x = 'stroke', y = 'price', data = df, ax = axs[1,0])

plt3.set\_xlabel('Stroke')

plt3.set\_ylabel('Price of Car (Dollars)')

#

plt4 = sns.scatterplot(x = 'compressionratio', y = 'price', data = df, ax = axs[1,1])

plt4.set\_xlabel('Compression Ratio')

plt4.set\_ylabel('Price of Car (Dollars)')

#

plt5 = sns.scatterplot(x = 'horsepower', y = 'price', data = df, ax = axs[2,0])

plt5.set\_xlabel('Horsepower')

plt5.set\_ylabel('Price of Car (Dollars)')

#

plt5 = sns.scatterplot(x = 'peakrpm', y = 'price', data = df, ax = axs[2,1])

plt5.set\_xlabel('Peak RPM')

plt5.set\_ylabel('Price of Car (Dollars)')

plt.tight\_layout()

plt.show()

# ## City Mileage & Highway Mileage

# In[35]:

df['mileage'] = df['citympg']\*0.55 + df['highwaympg']\*0.45

# In[36]:

plt1 = sns.scatterplot(x = 'mileage', y = 'price', data = df)

plt1.set\_xlabel('Mileage')

plt1.set\_ylabel('Price of Car (Dollars)')

plt.show()

# ## Bivariate Analysis

# #### Brand Category – Mileage

# In[37]:

plt1 = sns.scatterplot(x = 'mileage', y = 'price', hue = 'brand\_category', data = df)

plt1.set\_xlabel('Mileage')

plt1.set\_ylabel('Price of Car (Dollars)')

plt.show()

# #### Brand Category – Horsepower

# In[38]:

plt1 = sns.scatterplot(x = 'horsepower', y = 'price', hue = 'brand\_category', data = df)

plt1.set\_xlabel('Horsepower')

plt1.set\_ylabel('Price of Car (Dollars)')

plt.show()

# #### Mileage - Fuel Type

# In[39]:

plt1 = sns.scatterplot(x = 'mileage', y = 'price', hue = 'fueltype', data = df)

plt1.set\_xlabel('Mileage')

plt1.set\_ylabel('Price of Car (Dollars)')

plt.show()

# #### Horsepower - Fuel Type

# In[40]:

plt1 = sns.scatterplot(x = 'horsepower', y = 'price', hue = 'fueltype', data = df)

plt1.set\_xlabel('Horsepower')

plt1.set\_ylabel('Price of Car (Dollars)')

plt.show()

# # Linear Regression Model

# In[41]:

auto = df[['fueltype', 'aspiration', 'carbody', 'drivewheel', 'wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginetype',

'cylindernumber', 'enginesize', 'boreratio', 'horsepower', 'price', 'brand\_category', 'mileage']]

# In[42]:

auto.head()

# ## Visualising the Data

# In[43]:

plt.figure(figsize=(15, 15))

sns.pairplot(auto)

plt.show()

# In[44]:

plt.figure(figsize=(10, 20))

plt.subplot(4,2,1)

sns.boxplot(x = 'fueltype', y = 'price', data = auto)

plt.subplot(4,2,2)

sns.boxplot(x = 'aspiration', y = 'price', data = auto)

plt.subplot(4,2,3)

sns.boxplot(x = 'carbody', y = 'price', data = auto)

plt.subplot(4,2,4)

sns.boxplot(x = 'drivewheel', y = 'price', data = auto)

plt.subplot(4,2,5)

sns.boxplot(x = 'enginetype', y = 'price', data = auto)

plt.subplot(4,2,6)

sns.boxplot(x = 'brand\_category', y = 'price', data = auto)

plt.subplot(4,2,7)

sns.boxplot(x = 'cylindernumber', y = 'price', data = auto)

plt.tight\_layout()

plt.show()

# ### Data Preparation

# In[45]:

cyl\_no = pd.get\_dummies(auto['cylindernumber'], drop\_first = True)

# In[46]:

auto = pd.concat([auto, cyl\_no], axis = 1)

# In[47]:

brand\_cat = pd.get\_dummies(auto['brand\_category'], drop\_first = True)

# In[48]:

auto = pd.concat([auto, brand\_cat], axis = 1)

# In[49]:

eng\_typ = pd.get\_dummies(auto['enginetype'], drop\_first = True)

# In[50]:

auto = pd.concat([auto, eng\_typ], axis = 1)

# In[51]:

drwh = pd.get\_dummies(auto['drivewheel'], drop\_first = True)

# In[52]:

auto = pd.concat([auto, drwh], axis = 1)

# In[53]:

carb = pd.get\_dummies(auto['carbody'], drop\_first = True)

# In[54]:

auto = pd.concat([auto, carb], axis = 1)

asp = pd.get\_dummies(auto['aspiration'], drop\_first = True)

auto = pd.concat([auto, asp], axis = 1)

fuelt = pd.get\_dummies(auto['fueltype'], drop\_first = True)

auto = pd.concat([auto, fuelt], axis = 1)

auto.drop(['fueltype', 'aspiration', 'carbody', 'drivewheel', 'enginetype', 'cylindernumber','brand\_category'], axis = 1, inplace = True)

# ### Model Building

# #### Splitting the Data into Training and Testing sets

# In[55]:

from sklearn.model\_selection import train\_test\_split

np.random.seed(0)

df\_train, df\_test = train\_test\_split(auto, train\_size = 0.7, test\_size = 0.3, random\_state = 100)

# In[56]:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

num\_vars = ['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize','boreratio', 'horsepower', 'price','mileage']

df\_train[num\_vars] = scaler.fit\_transform(df\_train[num\_vars])

# In[57]:

df\_train.head()

# In[58]:

df\_train.describe()

# In[59]:

plt.figure(figsize = (16, 10))

sns.heatmap(df\_train.corr(), annot = True, cmap="YlGnBu")

plt.show()

# In[60]:

y\_train = df\_train.pop('price')

X\_train = df\_train

# In[61]:

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LinearRegression

# In[62]:

lm = LinearRegression()

lm.fit(X\_train, y\_train)

rfe = RFE(lm, 10) # running RFE

rfe = rfe.fit(X\_train, y\_train)

# In[63]:

list(zip(X\_train.columns,rfe.support\_,rfe.ranking\_))

# In[64]:

col = X\_train.columns[rfe.support\_]

col

# In[65]:

X\_train\_rfe = X\_train[col]

# In[66]:

import statsmodels.api as sm

X\_train\_rfe = sm.add\_constant(X\_train\_rfe)

# In[67]:

lm = sm.OLS(y\_train,X\_train\_rfe).fit() # Running the linear model

# In[68]:

print(lm.summary())

# In[69]:

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

X = X\_train\_rfe

vif['Features'] = X.columns

vif['VIF'] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

# In[70]:

X\_train\_new1 = X\_train\_rfe.drop(["twelve"], axis = 1)

# Adding a constant variable

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_new1)

lm = sm.OLS(y\_train,X\_train\_lm).fit() # Running the linear model

#Let's see the summary of our linear model

print(lm.summary())

# In[71]:

X\_train\_new2 = X\_train\_new1.drop(["mileage"], axis = 1)

# Adding a constant variable

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_new2)

lm = sm.OLS(y\_train,X\_train\_lm).fit() # Running the linear model

#Let's see the summary of our linear model

print(lm.summary())

# In[72]:

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

X = X\_train\_new2

vif['Features'] = X.columns

vif['VIF'] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

# In[73]:

X\_train\_new3 = X\_train\_new2.drop(["curbweight"], axis = 1)

# Adding a constant variable

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_new3)

lm = sm.OLS(y\_train,X\_train\_lm).fit() # Running the linear model

#Let's see the summary of our linear model

print(lm.summary())

# In[74]:

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

X = X\_train\_new3

vif['Features'] = X.columns

vif['VIF'] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

# In[75]:

X\_train\_new4 = X\_train\_new3.drop(["sedan"], axis = 1)

# Adding a constant variable

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_new4)

lm = sm.OLS(y\_train,X\_train\_lm).fit() # Running the linear model

#Let's see the summary of our linear model

print(lm.summary())

# In[76]:

X\_train\_new5 = X\_train\_new4.drop(["wagon"], axis = 1)

# Adding a constant variable

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_new5)

lm = sm.OLS(y\_train,X\_train\_lm).fit() # Running the linear model

#Let's see the summary of our linear model

print(lm.summary())

# In[77]:

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

X = X\_train\_new5

vif['Features'] = X.columns

vif['VIF'] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

# In[78]:

X\_train\_new6 = X\_train\_new5.drop(["dohcv"], axis = 1)

# Adding a constant variable

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_new6)

lm = sm.OLS(y\_train,X\_train\_lm).fit() # Running the linear model

#Let's see the summary of our linear model

print(lm.summary())

# In[79]:

y\_train\_price = lm.predict(X\_train\_lm)

# In[80]:

fig = plt.figure()

sns.distplot((y\_train - y\_train\_price), bins = 20)

fig.suptitle('Error Terms', fontsize = 20) # Plot heading

plt.xlabel('Errors', fontsize = 18) # X-label

# ### Making Predictions

# In[81]:

num\_vars = ['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize','boreratio', 'horsepower', 'price','mileage']

df\_test[num\_vars] = scaler.transform(df\_test[num\_vars])

# In[82]:

y\_test = df\_test.pop('price')

X\_test = df\_test

# In[83]:

X\_test\_new = X\_test[['carwidth', 'horsepower', 'Luxury', 'hatchback']]

# Adding a constant variable

X\_test\_new = sm.add\_constant(X\_test\_new)

# In[84]:

y\_pred = lm.predict(X\_test\_new)

# In[85]:

from sklearn.metrics import r2\_score

r2\_score(y\_test, y\_pred)

# In[86]:

fig = plt.figure()

plt.scatter(y\_test,y\_pred)

fig.suptitle('y\_test vs y\_pred', fontsize=20) # Plot heading

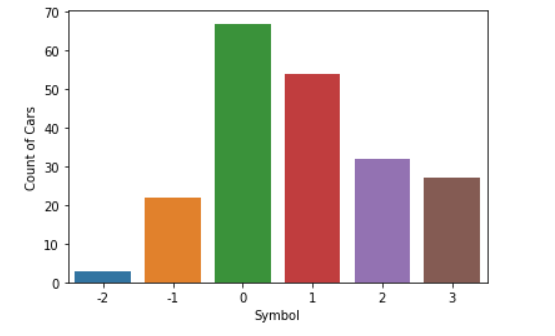
plt.xlabel('y\_test', fontsize=18) # X-label

plt.ylabel('y\_pred', fontsize=16) # Y-label

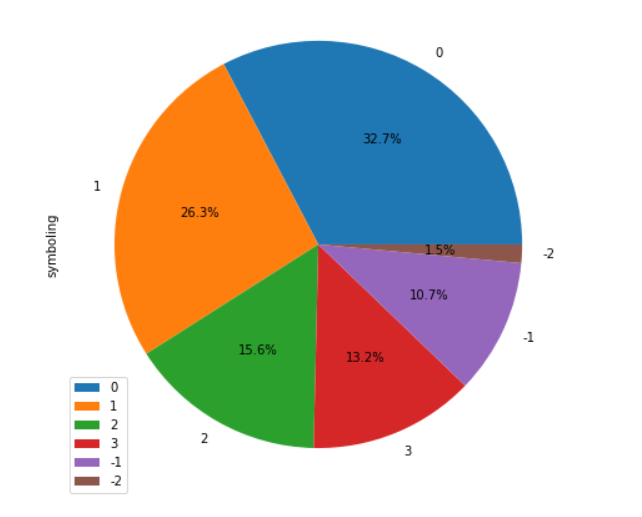
## 5.3 SCREENSHOT

# 

### 5.3.1 Analyzing symbols and count of cars with the help of bar graph.

****

### 5.3.2 Analyzing symbols and count of cars with the help of pie chart.

****

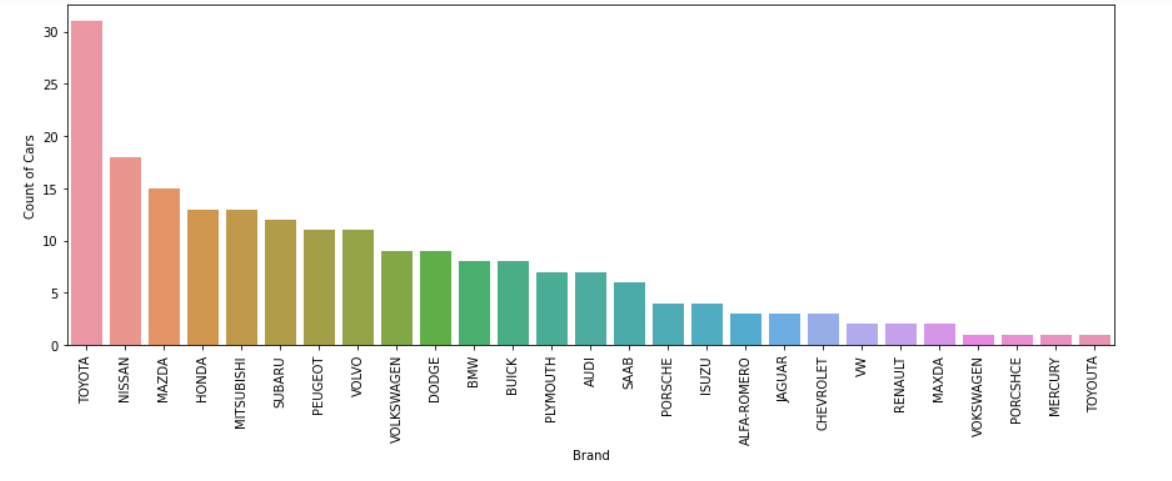
### 5.3.3 Bar graph with symbol on x-axis and avg price(dollars) on y-axis.

# 

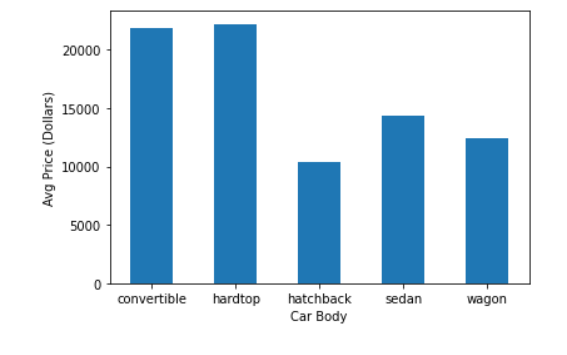
### 5.3.4 Bar graph with no of doors on x-axis and avg price(dollars) on y-axis.

# 

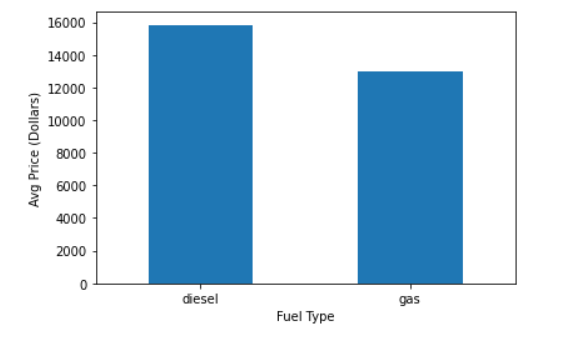
# 

5.3.5 Bar graph with brand on x-axis and count of cars on y-axis**.**

### 5.3.6 Bar graph with car body on x-axis and avg price(dollars) on y-axis**.**

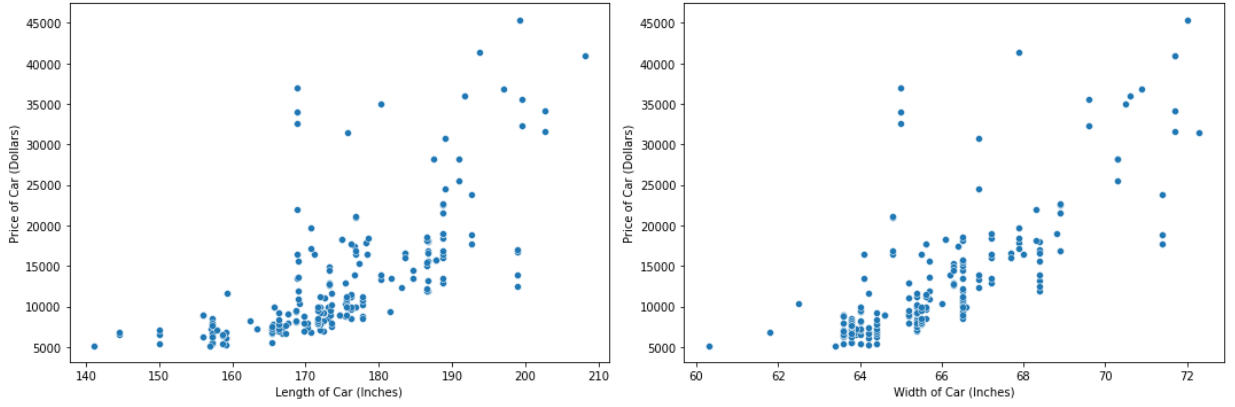


5.3.7 Bar graph with fuel type on x-axis and avg price(dollars) on y-axis**.**

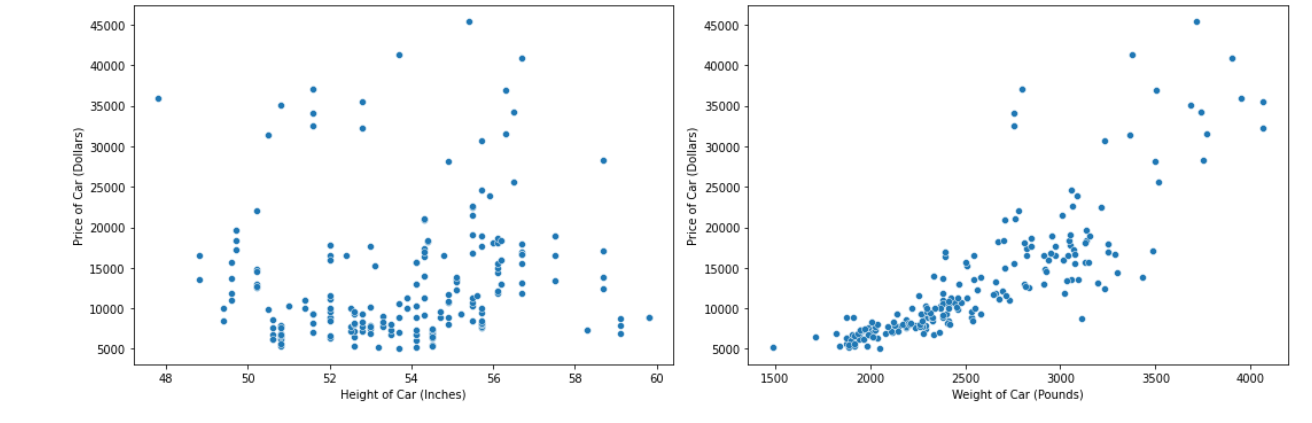
****

5.3.8 Bar graph with drive wheel type on x-axis and avg price(dollars) on y-axis**.**

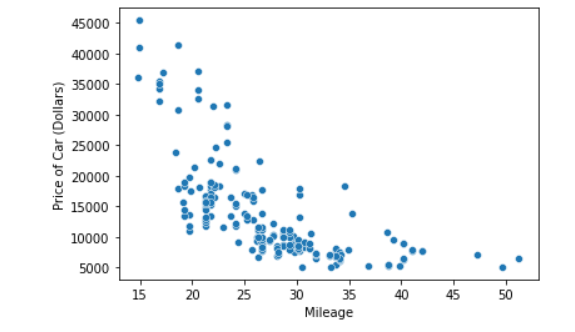
# 

5.3.9 Plotting length and width of car(inches) on x-axis, and price(dollars) on y-axis**.**

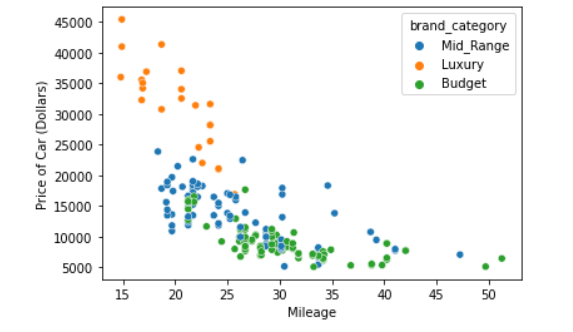
### 5.3.10 height of car(inches) & weight of car(pounds) on x-axis with price on y-axis.

****

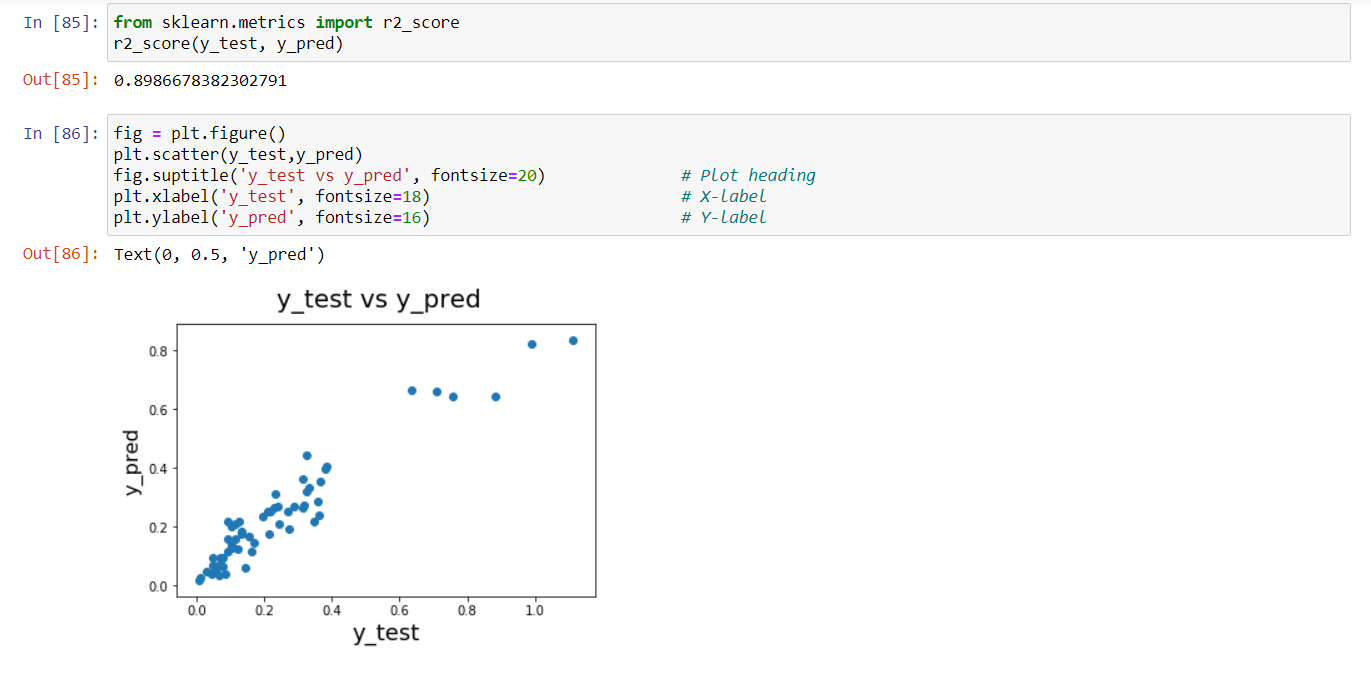
### 5.3.11 plotting mileage on x-axis and price of car(dollars) on y-axis

****

### 5.3.12 Plotting mileage of three brand categories on x-axis and price of car on y-axis.



### 5.3.13 Accuracy of the model.



# 

# CHAPTER 6

# SOFTWARE TESTING

## 6.1 DEFINATION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 6.2 SOFTWARE TESTING

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

## 6.3 BLACK BOX TESTING

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance.

## 6.4 WHITE BOX TESTING

White-box testing is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

## 6.5 SOFTWARE TESTING STRATEGIES

A strategy for software testing will begin in the following order:

* Unit testing
* Integration testing
* System testing
* Validation testing

## 6.6 TYPES OF TESTING

### 6.6.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### 6.6.2 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

### 6.6.3 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### 6.6.4 PERFORMANCE TESTING

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

### 6.6.5 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

### 6.6.6 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

## 6.7 BUILD THE TEST PLAN

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

# 

# CHAPTER 7

# CONCLUSION AND FUTURE ENHANCEMENT

## 7.1 CONCLUSION

Car price prediction can be a challenging task due to the high number of attributes that should be considered for the accurate prediction. The major step in the prediction process is collection and preprocessing of the data.

Data cleaning is one of the processes that increases prediction performance, yet insufficient for the cases of complex data sets as the one in this research. Applying k-means clustering algorithm on the data set, accuracy was less than 50%. Therefore, the linear regression algorithm has been proposed and this algorithm gains accuracy of 89.86%. This is significant improvement compared to k-means clustering approach. However, the drawback of the proposed system is that linear regression is susceptible to overfitting.

## 7.2 FUTURE ENHANCEMENT

Although, this system has achieved astonishing performance in car price prediction problem our aim for the future research is to test this system to work successfully with various data sets. We will extend our test data with eBay and OLX used cars data sets and validate the proposed approach.

# REFERENCES

[1] Agencija za statistiku BiH. (n.d.), retrieved from: http://www.bhas.ba . [accessed July 18, 2018.]

[2] Listiani, M. (2009). Support vector regression analysis for price prediction in a car leasing application (Doctoral dissertation, Master thesis, TU Hamburg-Harburg).

[3] Richardson, M. S. (2009). Determinants of used car resale value. Retrieved from: https://digitalcc.coloradocollege.edu/islandora/object /coccc%3A1346 [accessed: August 1, 2018.]

[4] Wu, J. D., Hsu, C. C., & Chen, H. C. (2009). An expert system of price forecasting for used cars using adaptive neuro-fuzzy inference. Expert Systems with Applications, 36(4), 7809-7817.

[5] Du, J., Xie, L., & Schroeder, S. (2009). Practice Prize Paper—PIN Optimal Distribution of Auction Vehicles System: Applying Price Forecasting, Elasticity Estimation, and Genetic Algorithms to Used-Vehicle Distribution. Marketing Science, 28(4), 637-644.

[6] Gongqi, S., Yansong, W., & Qiang, Z. (2011, January). New Model for Residual Value Prediction of the Used Car Based on BP Neural Network and Nonlinear Curve Fit. In Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on (Vol. 2, pp. 682-685). IEEE.

[7] Pudaruth, S. (2014). Predicting the price of used cars using machine learning techniques. Int. J. Inf. Comput. Technol, 4(7), 753-764. Noor, K., & Jan, S. (2017).

[8] Vehicle Price Prediction System using Machine Learning Techniques. International Journal of Computer Applications, 167(9), 27-31.

[9] Auto pijaca BiH. (n.d.), Retrieved from: https://www.autopijaca.ba. [accessed August 10, 2018].

[10] Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (n.d.), Retrieved from: https://www.cs.waikato.ac.nz/ml/weka/.[August 04, 2018].

[11] Ho, T. K. (1995, August). Random decision forests. In Document analysis and recognition, 1995., proceedings of the third international conference on (Vol. 1, pp. 278-282). IEEE.

[12] Russell, S. (2015). Artificial Intelligence: A Modern Approach (3rd edition). PE.

[13] Ben-Hur, A., Horn, D., Siegelmann, H. T., & Vapnik, V. (2001). Support vector clustering. Journal of machine learning research, 2(Dec), 125-137.

[14] Aizerman, M. A. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control, 25, 821-837.

[15] 3.2.4.3.1. sklearn.ensemble. RandomForestClassifier— scikit-learn 0.19.2 documentation. (n.d.). Retrieved from: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble .RandomForestClassifier.html [accessed: August 30, 2018] Used cars database. (n.d.) Retrieved from: https://www.kaggle.com/orgesleka/used-cars-database. [accessed: June 04, 2018].