University of Mumbai

2020-2021

M.H. SABOO SIDDIK COLLEGE OF ENGINEERING

8, Saboo Siddik Road, Byculla, Mumbai - 400 008

Department of Computer Engineering

Project Report

On

"CAR PRICE PREDICTION"

By

Nida Momin - 3219066

Mustansir Sabir - 3219068

Marzook Khatri - 3219071

Guided By

Prof. Saiqa Khan

M.H. SABOO SIDDIK COLLEGE OF ENGINEERING

8, Saboo Siddik Road, Byculla, Mumbai -400 008.

This is to certify that,

Roll No	Name
3219066	Nida Momin
3219068	Mustansir Sabir
3219071	Marzook Khatri

Of Final Year/Second year (B.E./S.E/T.E Semester VIII/IV/VI) degree course in Computer Engineering, have completed the specified project report on,

"**CAR PRCE PREDICTION**"

As a partial fulfillment of the project work in a satisfactory manner as per the rules of the curriculum laid by the University of Mumbai, during the Academic Year July 2020 — June 2021.

____

Internal Guide

__________________ __________________

Internal Examiner External Examiner

__________________ __________________

Head of Department Principal

This project report entitled **Car Price Prediction** by **Nida Momin, Mustansir Sabir and Marzook Khatri** is approved for the degree of Computer Engineering.

EXAMINERS

1.

2.

SUPERVISORS

1.

2.

Date:

Place:

ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to our parents for motivating and encouraging us throughout the career.

We wish to express our sincere thanks to our principal Dr. Ganesh Kame, M.H. Saboo Siddik College of Engineering for providing us all the facilities, support and wonderful environment to meet our project requirements.

We would also take the opportunity to express our humble gratitude to our Head of Department of Computer Engineering Dr. Zainab Pirani for supporting us in all aspects and for encouraging us with her valuable suggestions to make our project success.

We our highly thankful to our internal project guide Prof. Saiqa Khan whose valuable guidance helped us understand the project better, their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the project successfully.

We would also like to acknowledge with much appreciation the role of the staff of the Computer Department, especially the Laboratory staff, who gave the permission to use the labs when needed and the necessary material to complete the project.

We would like to express our gratitude and appreciate the guidance given by other supervisors and project guides, their comments and tips helped us in improving our presentation skills. Although there may be many who remain unacknowledged in this humble note of appreciation but there are none who remain unappreciated.

Nida Momin

Mustansir Sabir

Marzook Khatri

Table of Content

Sr.no	Contents	Page no.
	Abstract	1
1.	Introduction and Motivation	2
	1.1 Introduction	2
	1.2 Aim	3
	1.3 Objective	3
	1.4 Motivation	3
	1.5 Scope	4
	1.5.1 Local Scope	4
	1.5.2 Global Scope	4
2.	Problem Statement	5
3.	Requirement Analysis	6
	3.1 Review of Literature	6
	3.2 Existing System	7
	3.3 Drawbacks of Existing System	7
4.	Design Details	9
	4.1 Design of the System	9
	4.2 Flow Chart	12
	4.3 Block Diagram	13
	4.4 Data Flow Diagram	13
	4.5 Use Case Diagram	15
5.	Implementation Details	16
	5.1 Process Model	16
	5.2 Methodology	21
6.	Technology Used	22
	6.1 Hardware Requirement	22
	6.2 Software Requirement	22

	6.3 Programming Language Used	22
7.	Test Case Design 7.1 Unit Testing 7.2 Integration Testing 7.3 White Box Testing 7.4 Black Box Testing	25 25 25 25 26
8.	Results	27
9.	Project Timeline	28
10.	Task Distribution	29
11.	Conclusion and Future Scope 11.1 Conclusion 11.2 Future Scope	30 30 30
12.	References	31
13.	Source Code	32

List of Figures

Fig. No.	Figures	Page No.
1	Flowchart	12
2	Block Diagram	13
3.1	Level 0 DFD	13
3.2	Level 1 DFD	14
3.3	Level 2 DFD	14
4	Use Case	15
5	Regression Graph 1	27
6	Regression Graph 2	27
7	Project Timeline	28

Abstract

Nowadays cars are being sold more than ever. Developing countries adopt the lease culture instead of buying a new car due to affordability. Therefore, the rise of used cars sales is exponentially increasing. Car sellers sometimes take advantage of this scenario by listing unrealistic prices owing to the demand. Therefore, a need for a model arises that can assign a price for a vehicle by evaluating its features taking the prices of other cars into consideration.

This project presents a car price prediction system by using the supervised machine learning technique. It focuses on developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases.

We implement and evaluate Linear Regression Algorithm on a dataset which offered 83% prediction accuracy. Considerable number of distinct attributes are examined for the reliable and accurate prediction. We have used the dataset which is available on Kaggle, for this project as it clearly fulfills all the car attributes on which the price can be calculated.

The project proposes a system where price is dependent variable which is predicted, and this price is derived from factors like cars Company, engine size, horsepower, boreratio, wheelbase, car-width, car-length, fuel-type etc. The model can predict the price of cars accurately by choosing the most correlated features.

Chapter 1
Introduction and Motivation

1.1 Introduction

Car price prediction especially when the vehicle is used and not coming direct from the factory, is both a critical and important task. With increase in demand for used cars and upto 8 percent decrease in demand for the new cars, more and more vehicle buyers are finding alternatives of buying new cars outright. People prefer to buy cars through lease which is a legal contract between buyer and seller.

It is found through studies that finding fair estimated price of a used car is important as well as challenging. So, there is a need of accurate price prediction mechanism for the used cars. Prediction techniques of machine learning can be helpful in this regard. Machine learning uses two techniques, i.e. inductive and deductive. The deductive learning is based on the usage of existing facts and knowledge to deduce new knowledge and facts while in inductive machine learning new computer programs are created by finding patterns and rules in the new data sets which were never explored before.

We use deductive approach of linear regression since it creates new values based on existing values. In this technique, there is single dependent variable Y and there can be various independent variables X. The relationship among variables is direct or linear.

This project has the following goals:

i. Design: The project includes the design of a system explaining linear relationship between X and Y which are price and other factors like model and make of the car.

ii. Predict: The project predicts the price of the vehicle using linear regression model which identifies different patterns and projects and predicts the value of the vehicle.

iii. Confirm: The project finds out which variable associated with the vehicle is the best predictor of its price.

Due to rising fuel prices, fuel economy is of prime importance while determining the prices. Other factors such as the type of fuel it uses, aspiration, body-style, drive-wheels engine-location, wheel-base, length, width, heightcurb-weight, engine-type, num-of-Cylinders, engine-size, boreratio, Horsepower may also influence the price as well.

1.2 Aim

Our main aim is to predict the prices of the used cars given various attributes. This will assure the customer whether the prices are exact and correct and are not manipulated by the sellers. For this purpose we have used machine learning algorithm which makes it easier to analyze the data to track the prices. This is an excellent way to find the prices for people who prefer to buy cars through lease or second hand.

1.3 Objectives

Predicting the resale value of a car is not a simple task. It is trite knowledge that the value of used cars depends on a number of factors. The main objective is to predict the prices if used cars. This project is aimed at facilitating the prompt, accurate and easy management of prediction of car prices. It will be very helpful to the seller on one hand and to the customers as they would be able to get good and fast service.

For this following step have been considered:

- To build a supervised machine learning model for forecasting value of a vehicle based on multiple attributes.
- The system that is being built must be feature based i.e. feature wise prediction must be possible.
- Providing graphical comparisons to provide a better view.

1.4 Motivation

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including engine-size, car-length, fuel type, etc. can influence the actual worth of a car. The automotive industry is composed of a few top global multinational players and several retailers. Retail market features players who deal in both new and used vehicles.

From the perspective of a seller, it is also a dilemma to price a used car appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

1.5 Scope

1.5.1 Local Scope

The scope for the automotive sector in 2021 is currently stable, and changed from the negative view that we have had on the sector over the past three years. The used car market has demonstrated a significant growth in value contributing to the larger share of the overall market. The used car market in India accounts for nearly 3.4 million vehicles per year.

1.5.2 Global Scope

The car market has been increasing steadily by around 5% for the last ten years, showing the high demand for cars by the population worldwide. There are hundreds of car websites globally but none of them provide such a facility to predict the price of used cars based on their attributes. Our dataset of 200+ records was used with the cross-validation technique with ten folds. The car make, year manufactured, paint type, transmission type, engine capacity and mileage have been used to predict the price of second-hand cars using machine learning algorithms. Thus, we conclude that predicting the price of second-hand cars is a very risky enterprise but which is feasible. This system will be very useful to car dealers and car owners who need to assess the value of their cars. In the future, we intend to collect more data and more features and to use a larger

Chapter 2
Problem Statement

An automobile company **XYZ automobiles** aspires to enter the Indian market by setting up their manufacturing unit there and producing cars locally to give competition to their Indian and European counterparts.

They have contracted an automobile consulting company to understand the factors on which the pricing of cars depends. Specifically, they want to understand the factors affecting the pricing of cars in the American market, since those may be very different from the other market.

The company wants to know:

Which variables are significant in predicting the price of a car?

How well those variables describe the price of a car?

Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the Indian market?

Business Goal

We are required to model the price of cars with the available independent variables. It will be used by the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels. Further, the model will be a good way for management to understand the pricing dynamics of a new market.

Chapter 3
Requirement Analysis

3.1 Review of Literature

There are various applications and methods which inspired us to build our project. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of information like the technological stack, algorithms, and shortcomings of our project which led us to build a better project.

- CARS24

Cars24 is a web platform where seller can sell their used car. It is an Indian Start-up with a simplified user interface which asks seller parameters like car model, kilometers traveled, year of registration and vehicle type (petrol, diesel). These allow the web model to run certain algorithms on given parameters and predict the price.

- GET VEHICLE PRICE

Get Vehicle Price is an android app which works on similar parameters as of Cars24. This app predicts vehicle prices on various parameters like Fiscal power, horsepower, kilometers traveled. This app uses a machine learning approach to predict the price of a car, bike, electric vehicle and hybrid vehicle. This app can predict the price of any vehicle because of the smartly optimized algorithm.

- CARWALE

CarWale app is one of the top-rated car apps in India for new and used car research. It provides accurate on-road prices of cars, genuine user and expert reviews. It can also compare different cars with the car comparison tool. This app also helps you to connect with your nearest car dealers for the best offers available.

- CARTRADE

CarTrade is web and Android platform where user can research New Cars in India by exploring Car Prices, Car Specs, Images, Mileage, Reviews, and Car Comparisons. On this app one can Sell Used Car to genuine buyers with ease.

One can list their used car for sale along with the details like image, model, and year of purchase and kilometers so that it is displayed to lakhs of interested car buyers in their city. User can read user reviews and expert car reviews with images that help in finalizing a new car buying decision.

3.2 Existing System

Surprisingly, work on estimated the price of used cars is very recent but also very sparse. Predicting the price of second-hand cards has not received much attention from academia despite its huge importance for the society. Many researchers used artificial neural networks (ANN) to analyse the stock market and predict market behavior. They used four different supervised machine learning techniques namely kNN (k-Nearest Neighbor), Naïve Bayes, linear regression and decision trees to predict the price of second-hand cars. The best result was obtained using KNN.

Some also compared the use of neural networks with linear regression in order to predict the stock prices of companies. They also found that neural networks had superior performance both in terms of accuracy and speed compared to linear regression. Few used support vector machines (SVM) to predict the price of leased cars. They showed that SVM performed better than simple linear regression and multivariate regression. The improvement of SVM regression over simple regression was not expressed in simple measures like mean deviation or variance. Some models were optimized to handle nonlinear relationships which cannot be done with simple linear regression methods.

It was found that this model was reasonably accurate in predicting the residual value of used cars. Also the car manufacturers are more willing to produce vehicles which do not depreciate rapidly. The neural network was able to predict which customers were likely to renew their policy and which ones would terminate soon. Thus, we have seen that neural networks have been used successfully for predicting the price of various commodities.

3.3 Drawbacks of existing system

In the existing system, to predict the price of vehicles both two wheeler and four wheeler, a lot of data mining algorithms and machine learning algorithms were widely used. The major drawback of this existing system is they need more attributes in order to predict the vehicle price. More comparison techniques must be used to get the result more efficiently.

It is highly complicated to get sufficient data sets that were spread widely all over the world. The datasets can be collected only through online. But not on the offline mode. It is not possible for everyone to collect the data sets through online mode particularly in rural areas. The data sets will not have about the vehicles which were not used for long time and also the traditional model vehicles may or may not be included in the data sets.

The major drawbacks of existing system is the system is very slow due to most of the works about the keyword query just analyze individual points, and they are inappropriate to many applications that call for analysis of groups of different vehicle points. There are no fast query retrieval methods and is low due to lack of SVM under Constraints.

Chapter 4
Design Details

4.1 Design of the system

4.1.1 Data Collection

To accurately predict the prices of used cars, we used an open dataset to train our model. We used the 'Used Car Database' from Kaggle which is scraped from the German subsidiary of eBay, a publicly listed online classified portal. The dataset contains the prices and attributes of over 205 used cars sold with across 22 brands. Our dataset contains 25 unique attributes of a car being sold, out of which we removed a few irrelevant columns that have little to or no impact on a car's price from our analysis, such as some door number, aspiration, etc. Also we replaced misspelled car_company names such as Maxda into Mazda, Toyouta into Toyota.

4.1.2 Data Preparation and Training

To perform linear regression, the (numeric) target variable should be linearly related to at least one another numeric variable. We'll first subset the list of all (independent) numeric variables, and then make a **pair wise plot**.

Correlation of price with independent variables:

- Price is highly (positively) correlated with wheelbase, carlength, carwidth, curbweight, enginesize, horsepower (notice how all of these variables represent the size/weight/engine power of the car)
- Price is negatively correlated to citympg and highwaympg (-0.70 approximately). This suggest that cars having high mileage may fall in the 'economy' cars category, and are priced lower (think Maruti Alto/Swift type of cars, which are designed to be affordable by the middle class, who value mileage more than horsepower/size of car etc.)

Correlation among independent variables:

- Many independent variables are highly correlated (look at the top-left part of matrix): wheelbase, carlength, curbweight, enginesize etc. are all measures of 'size/weight', and are positively correlated

Thus, while building the model, we'll have to pay attention to multi-colinearity (especially linear models, such as linear and logistic regression, suffer more from multi-collinearity).
Additionally, we perform the following steps on the dataset helping us narrowing down the features –

- Split the attributes into X and Y where Y will include only price attribute and the rest included in X.
- Using dummy values
- Storing column names in cols, since column names are lost after
- Scaling the features so that the df is converted to a numpy array.
- Assigning values to train and test parameters.

After pre-processing, the final dataset contains 15 features for second hand cars – symboling, wheelbase, carlength, carwidth, carheight, curbweight, enginesize, boreratio, stroke, compressionratio, horsepower, peakrpm, citympg, highwaympg, and price.
Out of these features, the most important for our prediction model is price, enginesize, boreratio and horsepower.

After preparing the data, it is analyzed through visual exploration to gather insights about the model that can be applied to the data, understand the diversity in the data and the range of every field. We use a bar chart, box plot, distribution graph, etc. to explore each feature varies and its relation with other features including the target feature.

We split our input data into training, testing data and cross validation with a 70:20:10 split ratio. The splitting was done by picking at random which results in a balance between the training data and testing data amongst the whole dataset. This is done to avoid overfitting and enhance generalization.

4.1.3 Model selection

Linear regression is commonly used in forecasting and financial analysis—for a company to tell how a change in the GDP could affect sales, for example.
Linear regression is one of the most commonly used predictive modeling techniques. It is represented by an equation $Y = a + bX + e$, where a is the intercept, b is the slope of the line and

e is the error term. This equation can be used to predict the value of a target variable based on given predictor variable(s).
Simple linear regression is useful for finding relationship between two continuous variables. One is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship.
The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line.

Linear Regression is a very simple algorithm that can be implemented very easily to give satisfactory results. Furthermore, these models can be trained easily and efficiently even on systems with relatively low computational power when compared to other complex algorithms. Linear regression has a considerably lower time complexity when compared to some of the other machine learning algorithms.

The mathematical equations of Linear regression are also fairly easy to understand and interpret. Hence Linear regression is very easy to master. Linear regression fits linearly seperable datasets almost perfectly and is often used to find the nature of the relationship between variables.

Overfitting is a situation that arises when a machine learning model fits a dataset very closely and hence captures the noisy data as well. This negatively impacts the performance of model and reduces its accuracy on the test set. Regularization is a technique that can be easily implemented and is capable of effectively reducing the complexity of a function so as to reduce the risk of overfitting.
All these factors led us to choose Linear Regression model for our project.

4.2 Flow Chart

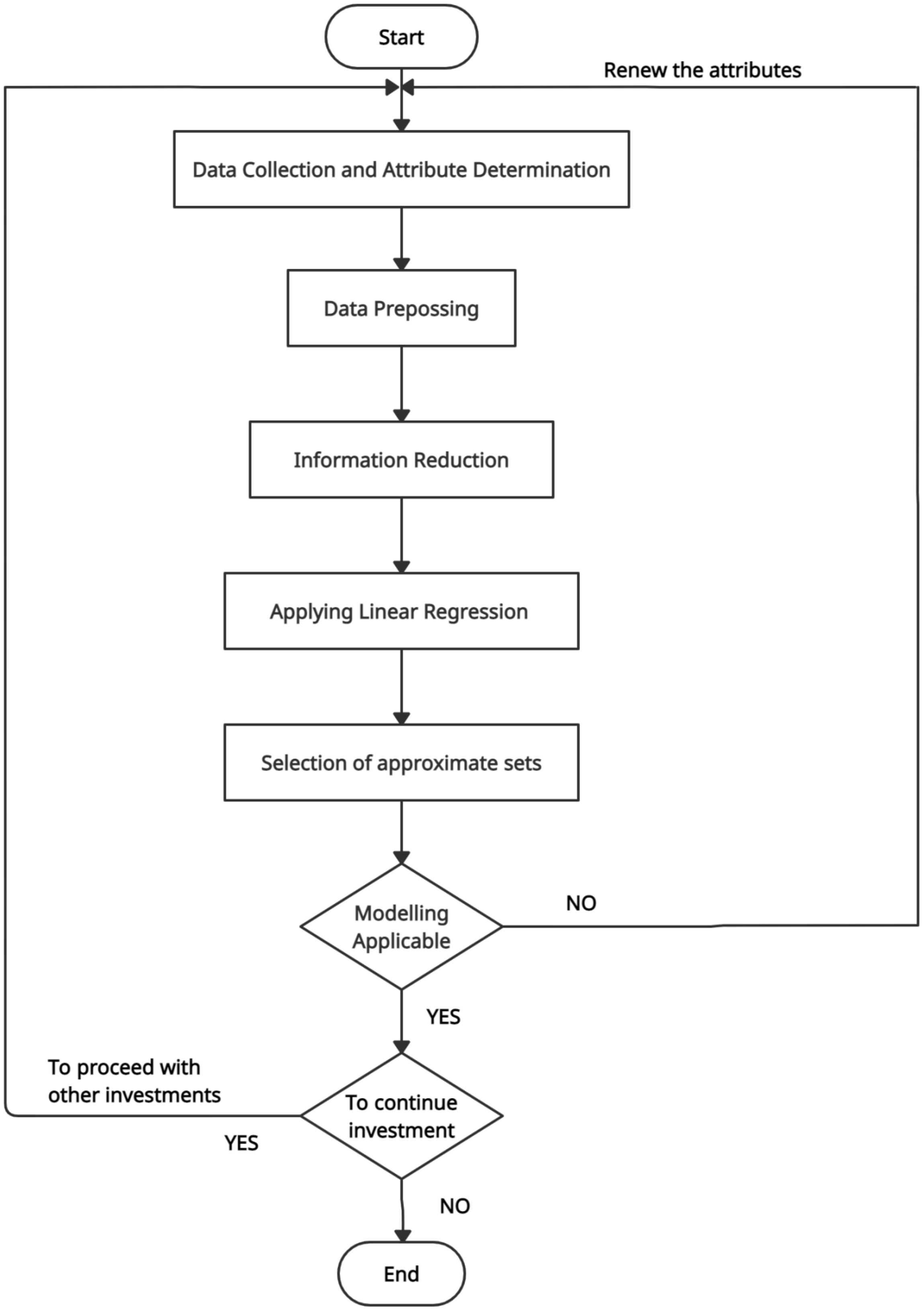

Start
Renew the attributes
Data Collection and Attribute Determination
Data Preposssing
Information Reduction
Applying Linear Regression
Selection of approximate sets
Modelling Applicable
NO
YES
To proceed with other investments
To continue investment
YES
NO
End

Fig.1: Flowchart for Car Price Prediction

4.3 Block Diagram

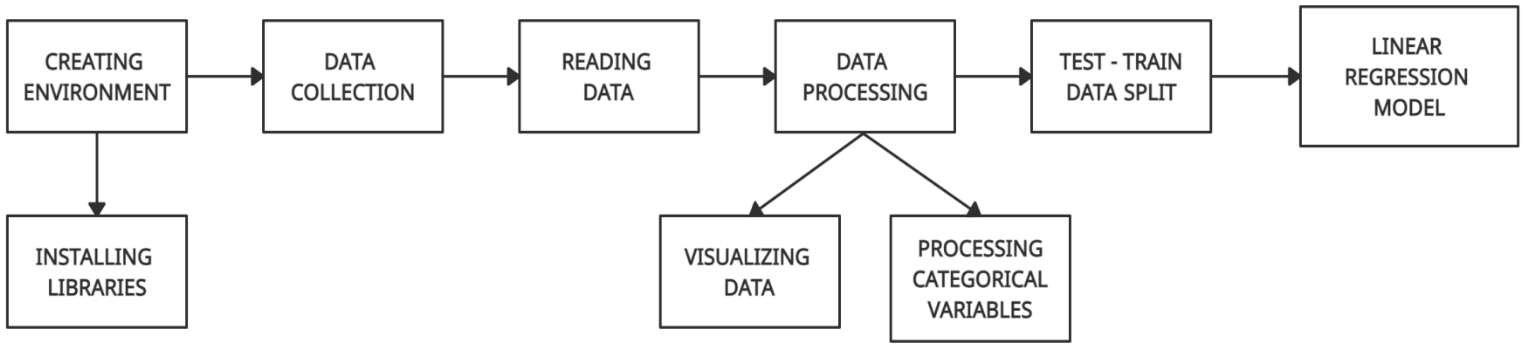


Fig.2: Block Diagram for Car price Prediction

4.4 Data Flow Diagram (DFD)

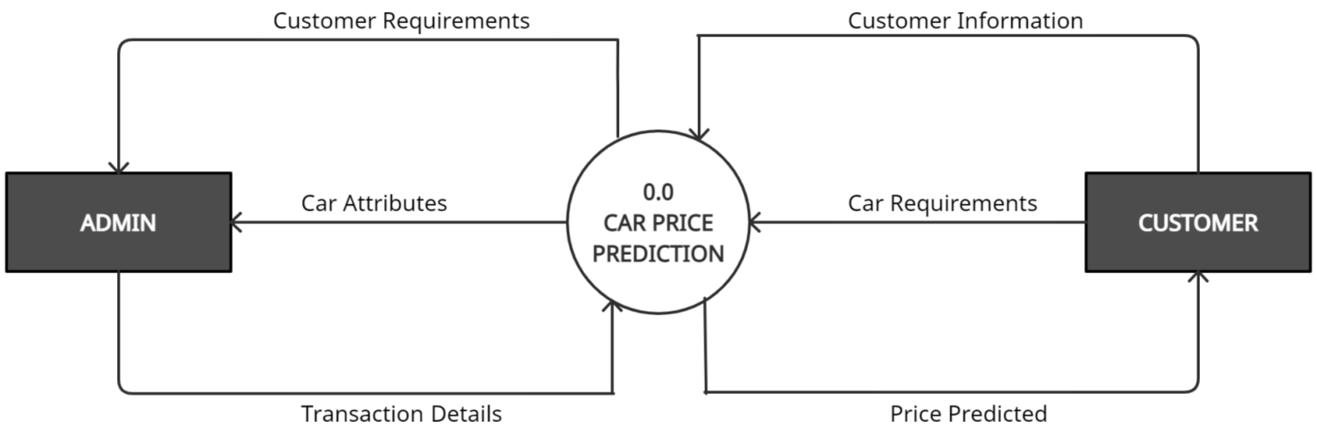


Fig.3.1: level 0 DFD

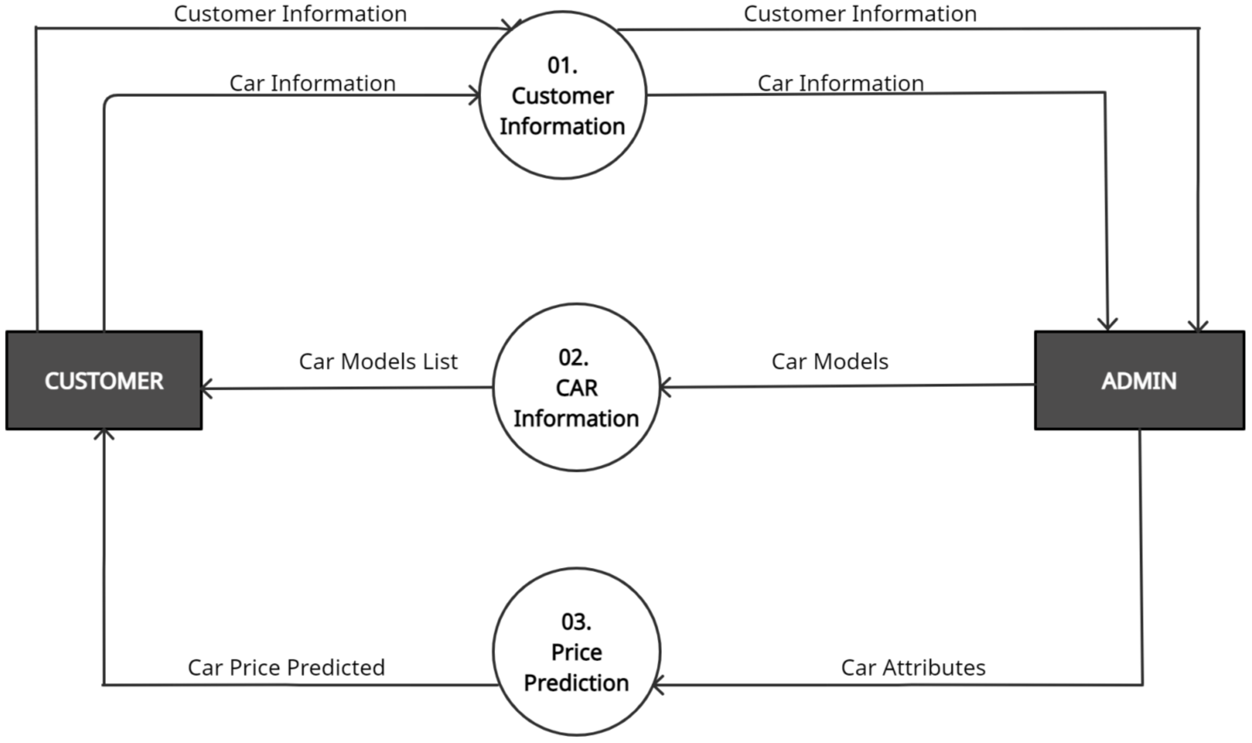


Fig. 3.2: Level 1 DFD

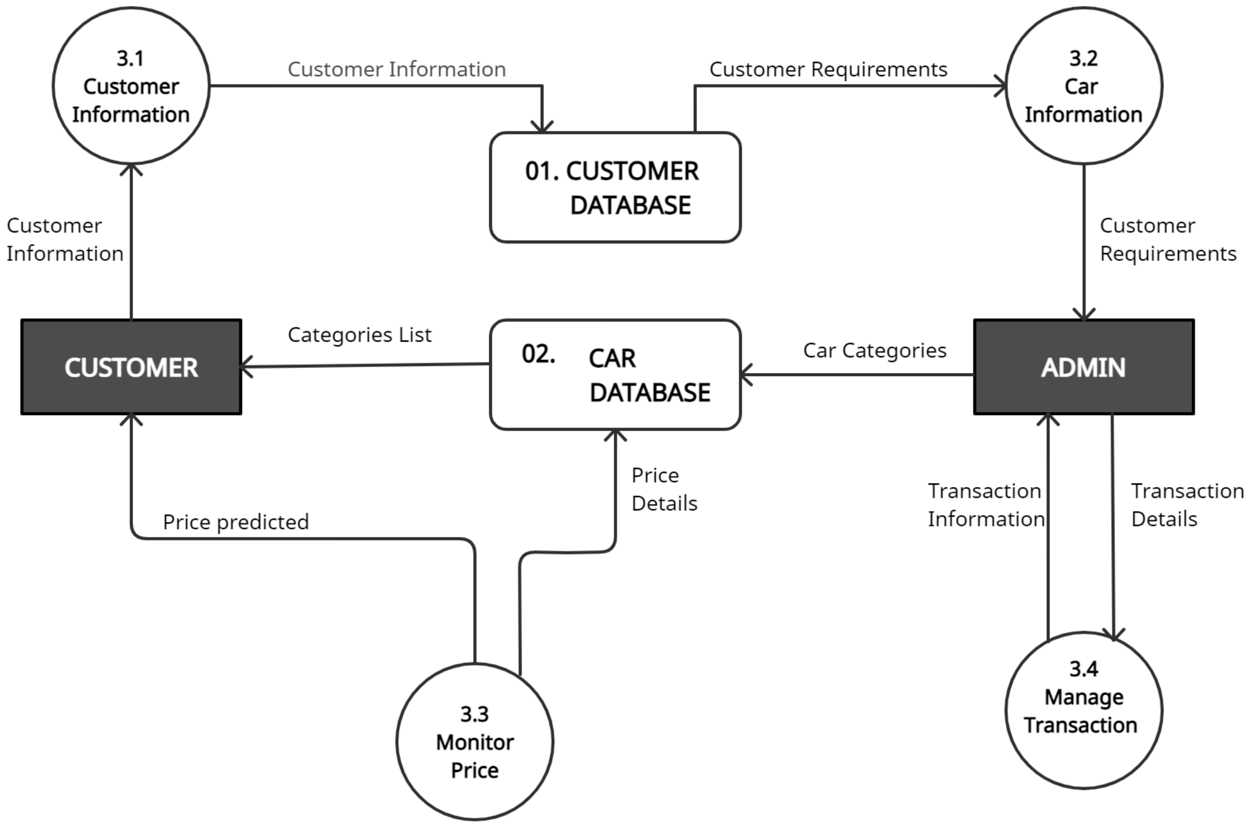


Fig. 3.3: Level 2 DFD

4.6 Use Case Diagram

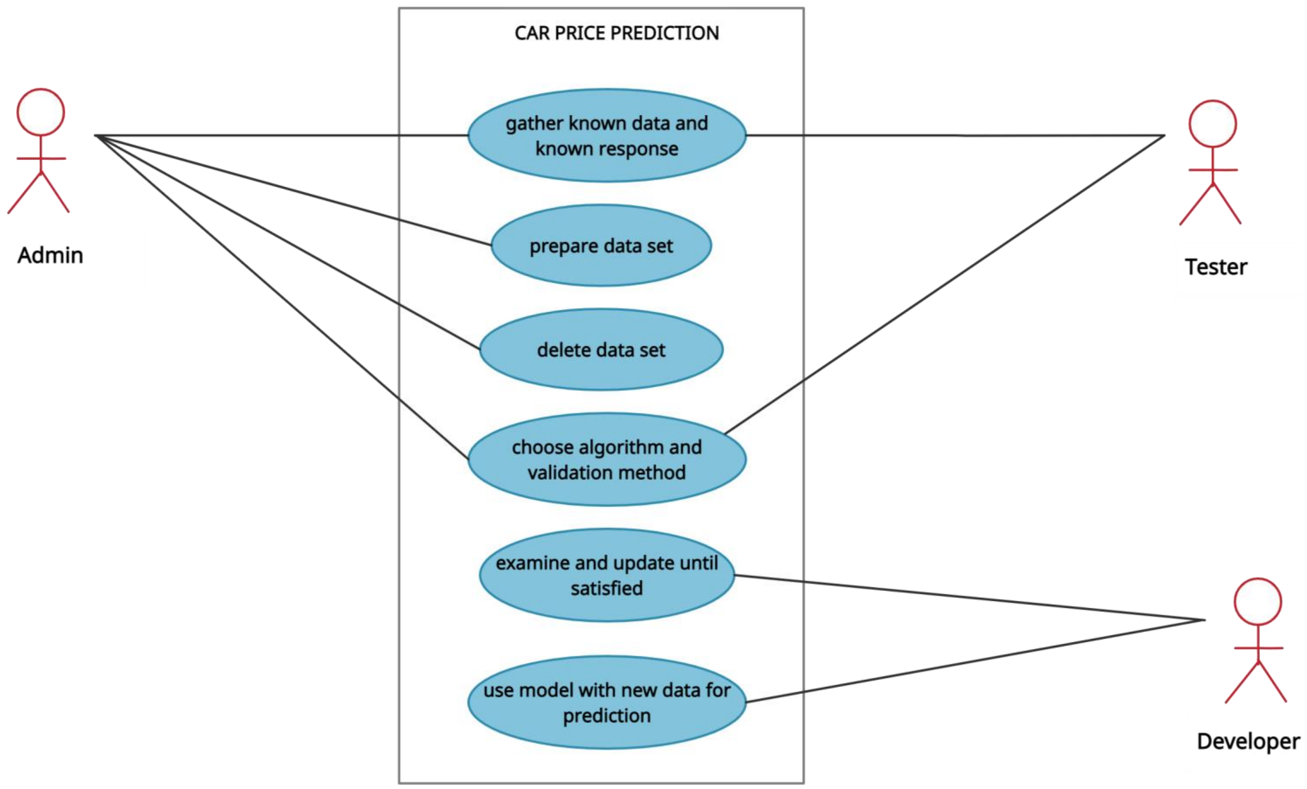

CAR PRICE PREDICTION
Admin
Tester
Developer
gather known data and known response
prepare data set
delete data set
choose algorithm and validation method
examine and update until satisfied
use model with new data for prediction

Fig. 4: Use Case diagram for Car Price Prediction

Chapter 5
Implementation Details

5.1 Process Model

A machine learning pipeline can be broken down into three major steps. Data collection, data modelling and deployment. All influence one another.

We may start a project by collecting data, model it, realize the data we collected was poor, go back to collecting data, model it again, find a good model, deploy it, find it doesn't work, make another model, deploy it, find it doesn't work again, go back to data collection. It's a cycle.

1. Problem definition—Rephrase your business problem as a machine learning problem

To help decide whether or not your business could use machine learning, the first step is to match the business problem you're trying to solve a machine learning problem.

The four major types of machine learning are supervised learning, unsupervised learning, transfer learning and reinforcement learning (there's semi-supervised as well but I've left it out for brevity). The three most used in business applications are supervised learning, unsupervised learning and transfer learning.

2. Data—If machine learning is getting insights out of data, what data do you have?

The data you have or need to collect will depend on the problem you want to solve.

If you already have data, it's likely it will be in one of two forms. Structured or unstructured. Within each of these, you have static or streaming data.

Structured data—Think a table of rows and columns, an Excel spreadsheet of customer transactions, a database of patient records. Columns can be numerical, such as average heart rate, categorical, such as sex, or ordinal, such as chest pain intensity.

Unstructured data—Anything not immediately able to be put into row and column format, images, audio files, natural language text.

Static data—Existing historical data which is unlikely to change. Your companies customer purchase history is a good example.

Streaming data—Data which is constantly updated, older records may be changed, newer records are constantly being added.

3. Evaluation—What defines success? Is a 95% accurate machine learning model good enough?

We have defined our business problem in machine learning terms and we have data. Now we define what defines success.

There are different evaluation metrics for classification, regression and recommendation problems. Which one you choose will depend on your goal.

A 95% accurate model may sound pretty good for predicting who's at fault in an insurance claim. But for predicting heart disease, you'll likely want better results.

Other things you should take into consideration for classification problems.

False negatives—Model predicts negative, actually positive. In some cases, like email spam prediction, false negatives aren't too much to worry about. But if a self-driving cars computer vision system predicts no pedestrian when there was one, this is not good.

False positives—Model predicts positive, actually negative. Predicting someone has heart disease when they don't, might seem okay. Better to be safe right? Not if it negatively affects the person's lifestyle or sets them on a treatment plan they don't need.

True negatives—Model predicts negative, actually negative. This is good.

True positives—Model predicts positive, actually positive. This is good.

Precision—What proportion of positive predictions were actually correct? A model that produces no false positives has a precision of 1.0.

Recall—What proportion of actual positives were predicted correctly? A model that produces no false negatives has a recall of 1.0.

F1 score—A combination of precision and recall. The closer to 1.0, the better.

Receiver operating characteristic (ROC) curve & Area under the curve (AUC)—The ROC curve is a plot comparing true positive and false positive rate. The AUC metric is the area under the ROC curve. A model whose predictions are 100% wrong has an AUC of 0.0, one whose predictions are 100% right has an AUC of 1.0.

4. Features—What features does your data have and which can you use to build your model?

Not all data is the same. And when you hear someone referring to features, they're referring to different kinds of data within data.

The three main types of features are categorical, continuous (or numerical) and derived.

Categorical features—One or the other(s). For example, in our heart disease problem, the sex of the patient. Or for an online store, whether or not someone has made a purchase or not.

Continuous (or numerical) features—A numerical value such as average heart rate or the number of times logged in.

Derived features—Features you create from the data. Often referred to as feature engineering. Feature engineering is how a subject matter expert takes their knowledge and encodes it into the data. You might combine the number of times logged in with timestamps to make a feature called time since last login. Or turn dates from numbers into "is a weekday (yes)" and "is a weekday (no)".

Text, images and almost anything you can imagine can also be a feature. Regardless, they all get turned into numbers before a machine learning algorithm can model them.

5. Modelling—Which model should you choose? How can you improve it? How do you compare it with other models?

Once you've defined your problem, prepared your data, evaluation criteria and features it's time to model.

Modelling breaks into three parts, choosing a model, improving a model, comparing it with others.

Choosing a model

When choosing a model, you'll want to take into consideration, interpretability and ease to debug, amount of data, training and prediction limitations.

Interpretability and ease to debug—Why did a model make a decision it made? How can the errors be fixed?

Amount of data—How much data do you have? Will this change?

Training and prediction limitations—This ties in with the above, how much time and resources do you have for training and prediction?

To address these, start simple. A state of the art model can be tempting to reach for. But if it requires 10x the compute resources to train and prediction times are 5x longer for a 2% boost in your evaluation metric, it might not be the best choice.

Linear models such as logistic regression are usually easier to interpret, are very fast for training and predict faster than deeper models such as neural networks.

Tuning and improving a model

A model's first results isn't its last. Like tuning a car, machine learning models can be tuned to improve performance.

Tuning a model involves changing hyperparameters such as learning rate or optimizer. Or model-specific architecture factors such as number of trees for random forests and number of and type of layers for neural networks.

These used to be something a practitioner would have to tune by hand but are increasingly becoming automated. And should be wherever possible.

Using a pre-trained model through transfer learning often has the added benefit of all of these steps been done.

Comparing models

Compare apples to apples.

Model 1, trained on data X, evaluated on data Y.

Model 2, trained on data X, evaluated on data Y.

Where model 1 and 2 can vary but not data X or data Y.

6. **Experimentation—What else could we try? How do the other steps change based on what we've found? Does our deployed model do as we expected?**

This step involves all the other steps. Because machine learning is a highly iterative process, you'll want to make sure your experiments are actionable.

Your biggest goal should be minimising the time between offline experiments and online experiments.

Offline experiments are steps you take when your project isn't customer-facing yet. Online experiments happen when your machine learning model is in production.

All experiments should be conducted on different portions of your data.

Training data set—Use this set for model training, 70–80% of your data is the standard.

Validation/development data set—Use this set for model tuning, 10–15% of your data is the standard.

Test data set—Use this set for model testing and comparison, 10–15% of your data is the standard.

These amounts can fluctuate slightly, depending on your problem and the data you have.

Poor performance on training data means the model hasn't learned properly. Try a different model, improve the existing one, collect more data, collect better data.

Poor performance on test data means your model doesn't generalise well. Your model may be overfitting the training data. Use a simpler model or collect more data.

Poor performance once deployed (in the real world) means there’s a difference in what you trained and tested your model on and what is actually happening. Revisit step 1 & 2. Ensure your data matches up with the problem you’re trying to solve.

When you implement a large experimental change, document what and why. Remember, like model tuning, someone, including your future self, should be able to reproduce what you’ve done.

This means saving updated models and updated datasets regularly.

5.2 Methodology

There are two primary phases in the system:

1. Training phase: The system is trained by using the data in the data set and fits a model (line/curve) based on the algorithm chosen accordingly.

2. Testing phase: the system is provided with the inputs and is tested for its working. The accuracy is checked. And therefore, the data that is used to train the model or test it, has to be appropriate.

The system is designed to detect and predict price of used car and hence appropriate algorithms must be used to do the two different tasks. Before the algorithm is selected for further use, different algorithms were compared for its accuracy. The well-suited one for the task was chosen.

Chapter 6
Technology Used

6.1 Hardware Specification

- Pentium 4 processor or higher
- 1 GB RAM
- 40 GB Hard Disk Space
- Internet Access

6.2 Software Requirements

- Operating System : Windows 7 or higher
- Python with its libraries
- PIP 2.7
- Jupyter notebook
- Chrome
- Dataset

6.3 Programming Language Used

Python was the major technology used for the implementation of machine learning concepts the reason being that there are numerous inbuilt methods in the form of packaged libraries present in python.

Following are prominent libraries/tools we used in our project.

NUMPY

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

PANDAS

Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. It has features which are used for exploring, cleaning, transforming and visualizing from data.

Pandas is an open-source python package built on top of Numpy developed by Wes McKinney. Pandas provides in-memory 2- d(2-dimensional) table object referred to as data frame. It is sort of a spreadsheet with row labels and column names. Therefore, with 2-d tables, pandas is capable of providing several other additional functions like building pivot tables, computing columns based on other columns and plotting graphs. Pandas additionally offer the simple mathematical functions such as addition, subtraction, etc. It also provides conditional operations and broadcasting along with basic mathematical functionalities. A spreadsheet with cell values, column names, and row index labels is represented by Pandas data frame object.

SCIKIT-LEARN

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Scikits is conventionally the name used for the extensions or modules of SciPy. The learning algorithms are provided by this module and it is called scikit-learn. A level of robustness and support needed for use in production systems is the vision for this library. This basically implies a major focus on considerations namely ease in use, quality of code, collaboration, documentation and performance. The main focus of the library is on modelling the data. It does not target on loading, manipulating and summarizing the data.

JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

TRAIN-TEST SPLIT

The dataset was then split into training and testing data. The train data is used for training our model while the test data was employed for testing and making predictions using our model. For this study we used 80% data to train our model while 20% data was then employed as test data for predictions. A Linear Regression model was employed for this study.

MATPLOTLIB

Matplotlib is especially deployed for basic plotting. Bars, pies, lines, scatter plots and so on are part of visualization using matplotlib. Multiple figures of this module can be opened, however have to be closed explicitly. Only the current figure is closed by plt.close() while plt.close('all') would shut them all. For data visualization in Python, Matplotlib is a graphics package well integrated with NumPy and Pandas. The MATLAB plotting commands are closely mirrored by the pyplot module. Therefore, the MATLAB users could simply transit to plotting with Python. Matplotlib has different stateful APIs for plotting and works with data frames and arrays. The object represents the figures and aces and therefore plot() like calls without parameters suffices, avoiding any need to manage parameters. Matplotlib is extremely customizable and powerful. Pandas uses Matplotlib and it is also a neat wrapper around Matplotlib.

SEABORN

Seaborn provides various visualization patterns. It has easy and interesting default themes and uses fewer syntax. Statistics visualization is the speciality of seaborn and it is employed while summarizing data in visuals and additionally depict the data distribution. Seaborn creates multiple figures which typically results in out of memory issues. Seaborn is additionally integrated for functioning with Pandas data frames. It extends the Matplotlib library for making ideal graphics with Python employing simple and easy methods. Seaborn is much more intuitive than Matplotlib and works with an entire dataset. In Seaborn, replot() is the API used with 'kind' parameter which specifies the type of plot that can be line, bar, or many of the other types.

Chapter 7
Test Case Design

Testing of the system proves that the system meets all requirements, including those for performance and security. The in depth security testing of this phase identifies any parts of the system that will not satisfy accreditation criteria.

7.1 Unit testing

With the help of Unit testing, which was performed manually, we fixed bugs at an early stage of development. This lead us to make changes in the code quickly and understand it better. Also it helps us to remove unnecessary dependencies by isolating the function.
Example, we had a function that needed variables or objects that are not created yet. In unit testing, these were accounted for in the form of mock objects created solely for the purpose of the unit testing done on that section of code.

7.2 Integration testing

First, we determined the Integration Test Strategy that could be adopted and later prepared the test cases and test data accordingly. Then we identified the Critical Modules. These needed to be tested on priority. Obtained the interface designs and created test cases to verify all of the interfaces in detail. Interface to database/external hardware/software application were tested in detail.
Following steps were done in order to perform integration testing:

- Prepare the Integration Tests Plan
- Design the Test Scenarios, Cases, and Scripts.
- Executing the test Cases followed by reporting the defects.
- Tracking & re-testing the defects
- Steps 3 and 4 were repeated until the completion of Integration was successful.

7.3 White Box Testing

White box Testing is usually reserved for critical systems and components. Following are the steps that were taken to perform White box testing:

Step 1: Identify the feature, component, program to be tested

Step 2: Plot all possible paths in a flowgraph we're trying to understand all the possible paths that can be tested for a given feature, component, module. Identifying all possible paths helps in writing test cases to cover each one of them. And the best way to do this, is to draw a flowgraph that brings out these paths

Step 3: Identify all possible paths from the flowgraph
Identify every permutation and combination for how the journey could flow from start to end. Identify any midway drop off points

Step 4: Write Test Cases to cover every single path on the flowgraph
When you have all available paths plotted on the flowgraph, then go ahead and write test cases to test each of these paths

Step 5: Execute, rinse, repeat

7.4 Black Box Testing

- For Black Box testing initially, the requirements and specifications of the system were examined.
- Valid inputs (positive test scenario) were chosen to check whether SUT processes them correctly.
- Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Expected outputs for all those inputs were determined.
- Test cases were constructed with the selected inputs. The test cases are then executed.
- The actual outputs were compared with the expected outputs.
- If any defects were found we fixed and re-tested.

Chapter 8
Results

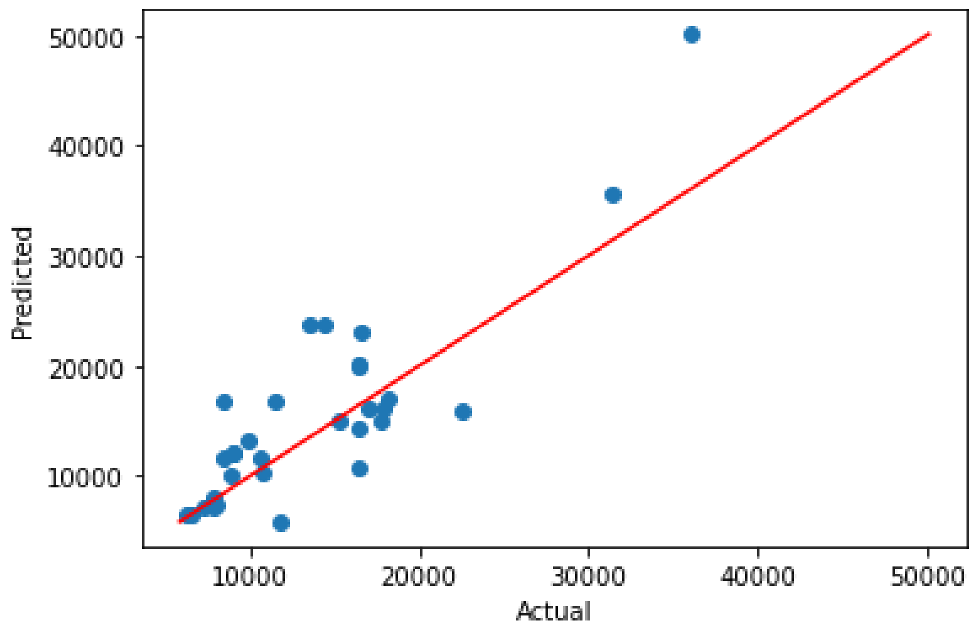


Fig. 5: Regression Graph 1

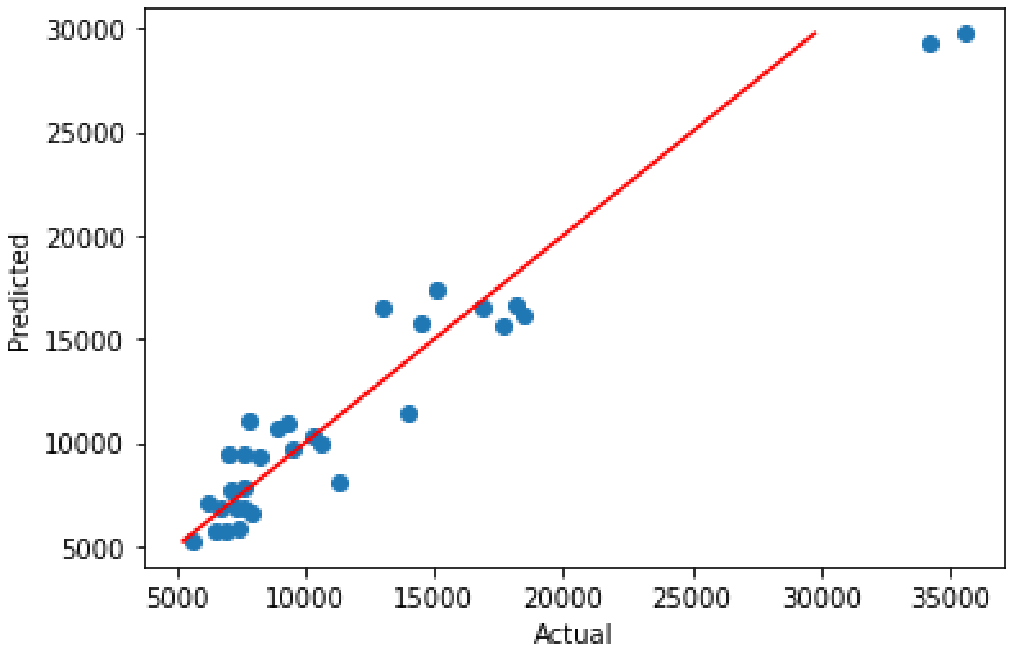
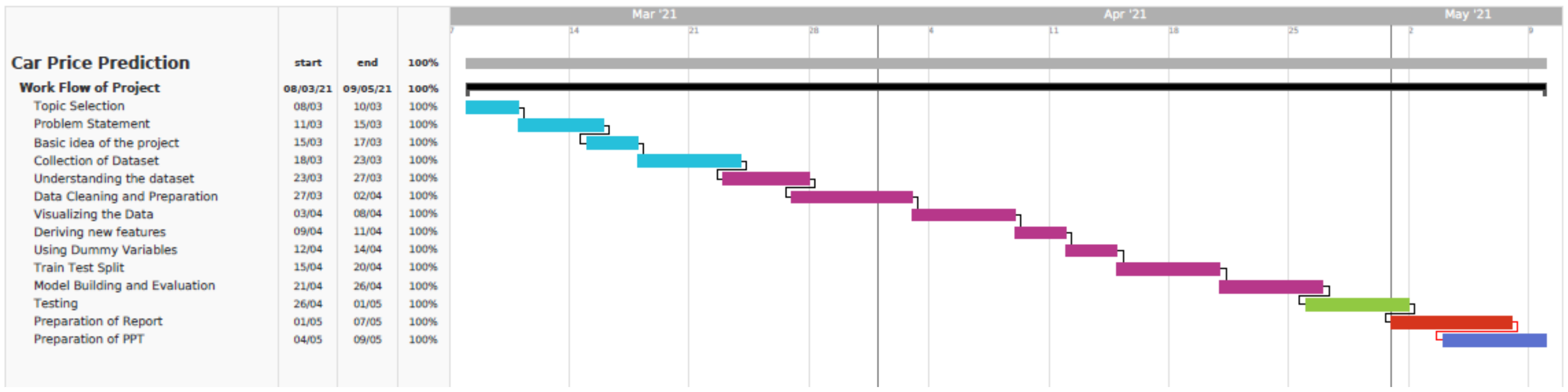


Fig. 6: Regression Graph 2

Chapter 9
Project Timeline



Car Price Prediction	start	end	100%
Work Flow of Project	**08/03/21**	**09/05/21**	**100%**
Topic Selection	08/03	10/03	100%
Problem Statement	11/03	15/03	100%
Basic idea of the project	15/03	17/03	100%
Collection of Dataset	18/03	23/03	100%
Understanding the dataset	23/03	27/03	100%
Data Cleaning and Preparation	27/03	02/04	100%
Visualizing the Data	03/04	08/04	100%
Deriving new features	09/04	11/04	100%
Using Dummy Variables	12/04	14/04	100%
Train Test Split	15/04	20/04	100%
Model Building and Evaluation	21/04	26/04	100%
Testing	26/04	01/05	100%
Preparation of Report	01/05	07/05	100%
Preparation of PPT	04/05	09/05	100%

Fig. 7: Project Timeline

Chapter 10
Task Distribution

Dividing up assignments for the team members requires forethought and planning. A key element to delegate strategy is making sure our team members are crystal clear on their roles. For our project Car price Prediction there were three members and we divided the task in equally so that there won't be a workload on any of the members. Below is a table with all the task and how it was divided.

Sr. No	Tasks	Members
1	Problem Statement	Marzook
2	Collecting Data sets	Nida
3	Data Exploration	Mustansir
4	Data Cleaning	Marzook
5	Data Preparation	Nida
6	Model Building and Evaluation	Mustansir
7	Report	Nida, Mustansir, Marzook

Chapter 11
Conclusion and Future Scope

11.1 Conclusion

This project evaluates used-car price prediction using Kaggle dataset which gives an accuracy of 91%. The most relevant features used for this prediction are price, horsepower, boreratio, fueltype and enginesize by filtering out outliers and irrelevant features of the dataset.
Being a simple regression model, Linear Regression gives good accuracy in comparison to prior work using these datasets. Our final model has satisfied the classical assumptions.
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.

11.2 Future Scope

As a part of future work, we aim at the variable choices over the algorithms that were used in the project. We could only explore an algorithms whereas many other algorithms which exist and might be more accurate. Keeping the current model as a baseline, we intend to use some advanced techniques like fuzzy logic and genetic algorithms to predict car prices as our future work. We intend to develop a fully automatic, interactive system that contains a repository of used-cars with their prices. This enables a user to know the price of a similar car using a recommendation engine, which we would work in the future. Also for better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset. More specifications will be added in a system or providing more accuracy in terms of price in the system.

References

- Mitchell, Tom (1997). Machine Learning. New York: McGraw Hill. ISBN 0-07-042807-7. OCLC 36417892
- Ian H. Witten and Eibe Frank (2011). Data Mining: Practical machine learning tools and techniques Morgan Kaufmann, 664pp., ISBN 978-0-12-374856-0.
- https://towardsdatascience.com/predicting-used-car-prices-with-machine-learning-techniques-8a9d8313952
- https://www.kaggle.com/c/1056lab-used-cars-price-prediction
- https://en.wikipedia.org/wiki/Linear_regression
- www.geeksforgeeks.com
- www.tutorialspoint.com
- www.researchgate.net

Source Code

1. Data Understanding

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import linear_model

from sklearn.linear_model import LinearRegression

# reading the dataset

cars = pd.read_csv("E:/ML_notes-master/ML_notes-master/Case Studies/Car Price Prediction/1.1. CarPrice_Data.csv")

cars.head()

print(cars.info())

cars['drivewheel'].astype('category').value_counts()

cars['aspiration'].astype('category').value_counts()

cars['symboling'].astype('category').value_counts()
```

Data Exploration

```
# all numeric (float and int) variables in the dataset

cars_numeric = cars.select_dtypes(include=['int64','float64'])

cars_numeric.head()

#cars_numeric = cars_numeric.drop(['symboling', 'car_ID'], axis=1)

cars_numeric.head()

cor = cars_numeric.corr()

cor

plt.figure(figsize=(16,8))


# heatmap
```

```
sns.heatmap(cor, cmap="YlGnBu", annot=True)

plt.show()
```

2. Data Cleaning

```
cars.info()

cars['symboling'] = cars['symboling'].astype('object')

cars.info()

cars['car_company'][:30]

carnames = cars['CarName'].apply(lambda x: x.split(" ")[0])

carnames[:30]

import re

p = re.compile(r'\w+-?\w+')

carnames = cars['car_company'].apply(lambda x: re.findall(p, x)[0])

print(carnames)

cars['car_company'].astype('category').value_counts()


# replacing misspelled car_company names

# volkswagen

cars.loc[(cars['car_company'] == "vw") |

        (cars['car_company'] == "vokswagen")

        , 'car_company'] = 'volkswagen'

# porsche

cars.loc[cars['car_company'] == "porcshce", 'car_company'] = 'porsche'

# toyota

cars.loc[cars['car_company'] == "toyouta", 'car_company'] = 'toyota'

# nissan

cars.loc[cars['car_company'] == "Nissan", 'car_company'] = 'nissan'
```

```
# mazda

cars.loc[cars['car_company'] == "maxda", 'car_company'] = 'mazda'

cars['car_company'].astype('category').value_counts()

cars.info()
```

3. Data Preparation

```
# split into X and y

X = cars[['symboling', 'fueltype', 'aspiration', 'doornumber',

      'carbody', 'drivewheel', 'enginelocation', 'wheelbase', 'carlength',

      'carwidth', 'carheight', 'curbweight', 'enginetype', 'cylindernumber',

      'enginesize', 'fuelsystem', 'boreratio', 'stroke', 'compressionratio',

      'horsepower', 'peakrpm', 'citympg', 'highwaympg',

      'car_company']]

y = cars['price']

cars_categorical = X.select_dtypes(include=['object'])

cars_categorical.head()

cars_dummies = pd.get_dummies(cars_categorical, drop_first=True)

cars_dummies.head()

X = pd.concat([X, cars_dummies], axis=1)

# scaling the features

from sklearn.preprocessing import scale


# storing column names in cols, since column names are (annoyingly) lost after

# scaling (the df is converted to a numpy array)

cols = X.columns

X = pd.DataFrame(scale(X))

X.columns = cols
```

```
X.columns

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.7,
                                                    test_size = 0.3, random_state=100)
```

4. Model Building and Evaluation

```
x=cars[['enginesize','horsepower']]

y=cars['price']

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.15,random_state= 10)

print (x_train.shape)

print (x_test.shape)

print(y_train.shape)

print(y_test.shape)


from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()  # Make an instance of the model

lin_model.fit(x_train, y_train)


y_train_predict = lin_model.predict(x_train)

rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))

print("The model performance for training set")

print('RMSE is {}'.format(rmse))

print("\n")
```

```
# on testing set

y_test_predict = lin_model.predict(x_test)

rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))

print("The model performance for testing set")

print('RMSE is {}'.format(rmse))


from sklearn.metrics import r2_score

print(r2_score(y_test,y_test_predict))


plt.scatter(y_test, y_test_predict)

plt.plot([min(y_test_predict),max(y_test_predict)],[min(y_test_predict),max(y_test_predict)],
color='red')

plt.xlabel('Actual')

plt.ylabel('Predicted')


x=cars[['enginesize','horsepower','stroke','compressionratio','peakrpm','citympg', 'highwaympg']]

y=cars['price']


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.15, random_state=5)

lin_model.fit(x_train, y_train)

y_train_predict = lin_model.predict(x_train)

rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))


print("The model performance for training set")

print('RMSE is {}'.format(rmse))

print("\n")


# on testing set
```

```
y_test_predict = lin_model.predict(x_test)

rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))

print("The model performance for testing set")

print('RMSE is {}'.format(rmse))


from sklearn.metrics import r2_score

print(r2_score(y_test,y_test_predict))

plt.scatter(y_test, y_test_predict)

plt.plot([min(y_test_predict),max(y_test_predict)],[min(y_test_predict),max(y_test_predict)],
color='red')

plt.xlabel('Actual')

plt.ylabel('Predicted')
```