CAR RESALE VALUE PREDICTION

NALAIYA THIRAN PROJECT BASED LEARNING

On

PROFESSIONAL READINESS FOR INNOVATION,EMPLOYABILITY AND ENTREPRENEURSHIP

Submitted By

TEAM ID: PNT2022TMID10192

SHIVANESAN S	19106107
SIVASANJEEVI R	19106111
SRIRAM B S	19106114
THANGA SUGANTHAN R	19106125

in partial fulfillment for the award of the degree

of

Bachelor of ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution,Affiliated to Anna University,Chennai)

COIMBATORE - 641032

NOVEMBER 2022

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution,Affiliated to Anna University,Chennai)

COIMBATORE - 641032

INTERNAL MENTOR

K.SURESHKUMAR

Assistant Professor

Department of Electronics and communication Engineering

Hindusthan college of engineering and technology

Coimbatore - 641032

INDUSTRY MENTOR

Prof Swetha

ABSTRACT

The Car Resale value prediction which implements ,that the price of a new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes.

So, customers buying a new car can be assured of the money they invest to be worthy. But, due to the increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features.

Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and it's value in the present day scenario. In fact, seller also has no idea about the car's existing value or the price he should be selling the car at.

To overcome this problem we have developed a model which will be highly effective. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value. Because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of a car according to user's inputs.

TABLE OF CONTENTS

1. INTRODUCTION

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is 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 various learning methods on a dataset consisting of the sale prices of different makes and models . We will compare the performance of various machine learning algorithms like Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Decision Tree Regressor and choose the best out of it. Depending on various parameters we will determine the price of the car. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of a car according to user's inputs.

1.1 PROJECT OVERVIEW

With difficult economic conditions, it is likely that sales of second-hand imported (reconditioned) cars and used cars will increase. In many developed countries, it is common to lease a car rather than buying it outright. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e. its expected resale value. Thus, it is of commercial interest to sellers/financers to be able to predict the salvage value (residual value) of cars with accuracy.

In order to predict the resale value of the car, we proposed an intelligent, flexible, and effective system that is based on using regression algorithms. Considering the main factors which would affect the resale value of a vehicle a regression model is to be built that would give the nearest resale value of the vehicle. We will be using various regression algorithms and algorithm with the best accuracy will be taken as a solution, then it will be integrated to the web-based application where the user is notified with the status of his product.

1.2 PURPOSE

Due to the huge requirement of used cars and lack of experts who can determine the correct valuation, there is an utmost need of bridging this gap between sellers and buyers. This project focuses on building a system that can accurately predict a resale value of the car based on minimal features like kms driven, year of purchase etc.

2. LITERATURE SURVEY

CAR RESALE VALUE PREDICTION SURVEYS

1. Car Price Prediction using Machine Learning Techniques Authors: Enis gegic. 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 2 was integrated into Java application. Furthermore, the model was evaluated using test data and the accuracy of 87.38% was obtained.

2. Price Evaluation Model In Second Hand Car System Based On BP Neural Network Theory Authors: Ning sun.

With the rapid growth of the number of private cars and the development of the second-hand car market, second-hand cars have become the main choice when people buy cars. The online second-hand car platform provides both buyers and sellers the chance of online P2P trade. In such systems, the accuracy of second-hand car price evaluation largely determines whether the seller and the buyer can get more efficient trading experience.

3. Prediction of Car Price using Linear Regression Authors: A. Rengarajan .
In this paper, we look at how supervised machine learning techniques can be used to forecast car prices in India. Data from the online marketplace quikr was used to make the predictions. The predictions were made using a variety of methods, including multiple linear regression analysis, Random forest regressor and Randomized search CV. The predictions are then analyzed and compared to determine which ones provide the best results. **4. Vehicle Price Prediction System using Machine Learning Techniques Authors: Kanwal Noor.**

In this paper, they proposed a model to predict the price of the cars through multiple linear regression method. Here system were able to achieve high level of accuracy using Multiple linear regression models to predict the price of cars collected from used cars website in Pakistan called Pak Wheels that totalled to 1699 records after pre-processing, and where 3 able to achieve accuracy of 98%, this was done after reducing the total amount of attributes using variable selection technique to include significant attributes only and to reduce the complexity of the model.

5. Predicting the Price of Second-hand Cars using Artificial Neural Networks Authors: Saamiyah Peerun.

The aim of this study is to assess whether it is possible to predict the price of second hand cars using artificial neural networks. Thus, data for 200 cars from different sources was gathered and fed to four different machine learning algorithms. We found that support vector machine regression produced slightly better results than using a neural network or linear regression. However, some of the predicted values are quite far away from the actual prices, especially for higher priced cars.

6. Used Car Price Prediction using K-Nearest Neighbor Based Model Authors: K.Samruddhi.

In this paper, a machine learning model is proposed to estimate the cost of the used cars using the K-Nearest Neighbor algorithm. The model is trained with used cars 7 data for different trained and test ratios. Then the proposed model is cross-validated using K fold method to examine the performance to avoid the over fit.

7. Prediction of Prices for Used Car by Using Regression Models Authors: Nitis Monburinon.

In this paper, the authors selected the data from the German ecommerce site. The main goal of this work is to find a suitable predictive model to predict the used cars price. They used different machine learning techniques for comparison and used the mean absolute error(MAE) as the metric. They proposed that their model with gradient boosted regression has a lower error with MAE value 0.28 and this gives the higher performance where linear regression has the MAE value 0.55, random forest with MAE value 0.35.

8. Used car price prediction using SVM Authors: Gegic..

In this paper, using data scrapped from a local Bosnian website for used cars totalled at 797 car samples after pre-processing, and proposed using these methods: Support Vector Machine, Random Forest and Artificial Neural network. Results have shown using only one machine learning algorithm achieved results less than 50%, whereas after combing the algorithms with pre calcification of prices using Random Forest, results with accuracies up to 87.38% was recorded

2.1 EXISTING PROBLEM

Unknown history,You may not know the accident and/or mechanical history of a used vehicle. Higher financing rates: Used cars tend to come with higher financing rates than their new counterparts, leading to increased costs down the line.

2.2 REFERENCES

[1] Sameerchand Pudaruth, “Predicting the Price of Used Cars using Machine Learning Techniques”;(IJICT 2014)

[2] Enis gegic, Becir Isakovic, Dino Keco, Zerina Masetic, Jasmin Kevric, ”Car Price Prediction Using Machine Learning”; (TEM Journal 2019)

2.3 PROBLEM STATEMENT DEFINITION

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
Customer need to resale a car	Customer	Buy resale car	Could not trust any seller	The condition of the battery seems, so poor in seller car	Sad
To get complete details of the car	Car enthusiastic	Get more number of cars in different brands	Can't get a trustworthy retailer	No warranty of buying pre-owned cars	Reluctant to buy
To complete review of the car	Car buyer	Search car based on expected amount to be affordable	Can't able to filter the car based on the particular amount	There is no such options to search based on amount	To search another platform that are better that it
To check the refinement of the car	Speed freak	To get high mileage of cars	An used car is not reliable	Because of low maintenance	Inconvenient

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas: An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

3.1 EMPATHY MAP CANVAS

Ideation Phase
Empathize & Discover

Date	26 september 2022
Team ID	PNT2022TMID10192
Project Name	Project - Car Resale value Prediction
Maximum Marks	4 Marks

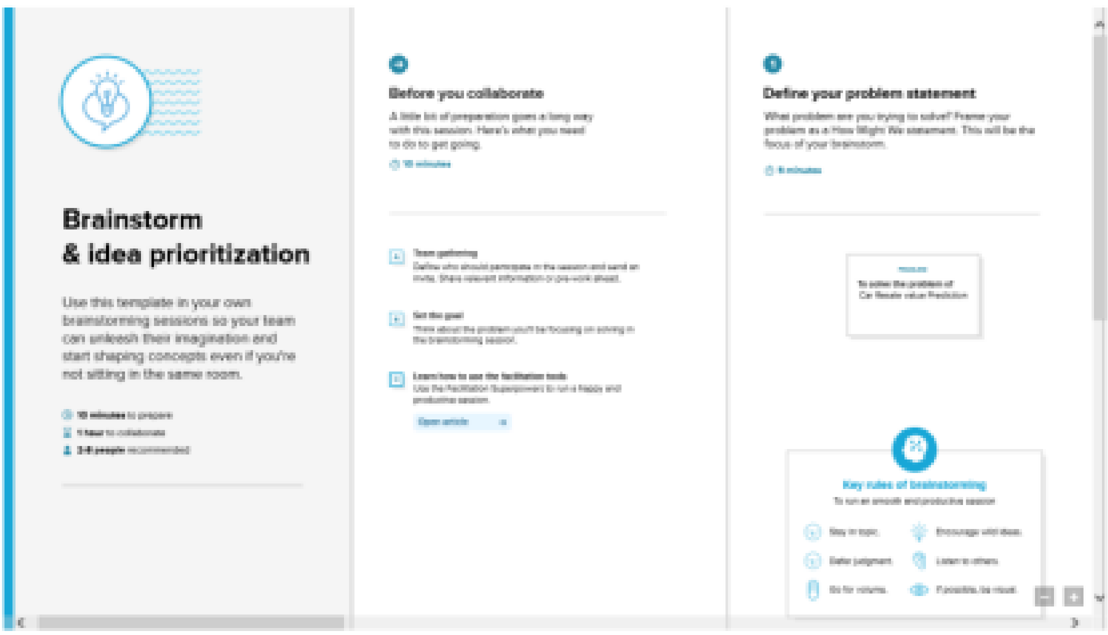
Car Resale value Prediction

3.2 IDEATION & BRAINSTORMING

Brainstorm &Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: https://www.mural.co/templates/empathy-map-canvas

Step-1: Team Gathering, Collaboration and Select the Problem Statement



3.3 PROPOSED SOLUTION

CAR RESALE VALUABLE PREDICTION

Team Id

PNT2022TMID10192

Team Lead

S SHIVANESAN

Team Members

R SIVA SANJEEVI
R THANGA SUGANTHAN
B S SRIRAM

Problems

- DIFFERENT FACTORS WILL AFFECT THE VALUE OF YOUR VEHICLE SUCH AS THE MILEAGE, THE CONDITION, YOUR LOCATION, AND THE COLOR OF THE CAR.

S SHIVANESAN

USER FRIENDLY APPLICATION

Price predictions

SUPERVISED LEARNING

LINEAR REGRESSION

R SIVA SANJEEVI

GRADIENT BOOSTING

DATA PREPROCESSING

CUSTOMER SATISFACTION

PREDICTIVE DATA MINING

B S SRIRAM

MACHINE LEARNING

SELLING AND BUYERS MANAGEMENT

VALUE DEPRECIATION

MANUFACTURING AND MARKETING

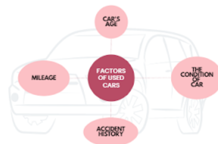
R THANGA SUGANTHAN

STICK PRICE DATASET

RESULT VISUALIZATION

FINANCIAL DOMAIN

DATA ACCURACY



TECHNOLOGIES

Obtain benchmark of base car Pricing

Use typical, easily accessible features of cars

Cross-check with used car listing price and car brands

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The main objective of the project is to predict the price of second hand cars using the various Machine Learning (ML) models. This can enable the customers to make decisions based on different inputs or factors namely Brand or Type of the car one prefers like Ford, Hyundai, Model of the car namely Ford, Hyundai Year of manufacturing like 2001...Type of fuel namely Petrol, Diesel, Price range or Budget, Type of transmission which the customer prefers like Automatic or Manual, Mileage to name a few characteristic features required by the customer. This project Car Price Prediction deals with providing the solution to these problems. Different techniques like multiple linear regression analysis, k-nearest neighbours, naive bayes and decision trees have been used to make the predictions. The predictions are then evaluated and compared in order to find those which provide the best performances.
2.	Idea / Solution description	New cars of a particular make, model, and year all have the same retail price, excluding optional features. This price is set by the manufacturer. Used car, however, are subject to supply-and-demand pricing. Further, used cars have additional attributes that factor into the price. These include the condition, mileage, and repair history, which sets cars that may have shared a retail price apart.
3.	Novelty / Uniqueness	The purpose of this thesis is to evaluate several different machine learning models for used car price prediction and draw conclusions about how they behave. This will deepen the knowledge of machine learning applied to car valuations and other similar price prediction problems.

4.	Social Impact / Customer Satisfaction	This work will focus on answering the research questions. They all entail a comparison of different ML algorithms for price prediction. This will be accomplished by sourcing and preparing a dataset on which all the algorithms can be trained on and compared fairly. The algorithms selected must therefore be similar enough for the same dataset to be used for all of them. This also means that no large optimization efforts on the dataset will be made to boost the performance, if these changes do not benefit the other models. Maximizing price prediction performance of any one algorithm in ways that do not offer better comparisons is outside the scope of this work.
5.	Business Model (Revenue Model)	A revenue model is a blueprint that shows how a start up business will earn revenue or gross income from its standard business operations, and how it will pay for operating costs and expenses.
6.	Scalability of the Solution	Which of the models and parameters gives the best overall accuracy in making price predictions for used cars. The optimal parameters were determined in the process of implementing the models, and thus each model was implemented with the parameters that yielded the best performance by trial and error. All of the models approximated geometric appreciation, meaning that a constant percentage of value is lost every year independent of the age of the vehicle. Random Forest Regression had a significantly higher assessed average depreciation at approximately 13.8%, compared to the others with 9.7%. This is closer to the range of 15%-31% assessed by Karl Stockman in his analysis of international depreciation rates

3.4 PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why.

4. REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

4.1 FUNCTIONAL REQUIREMNTS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Car Details	Mandatory field for analyzing the price
FR-2	Result	The Price will be shown based on the given details

4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional Requirements:

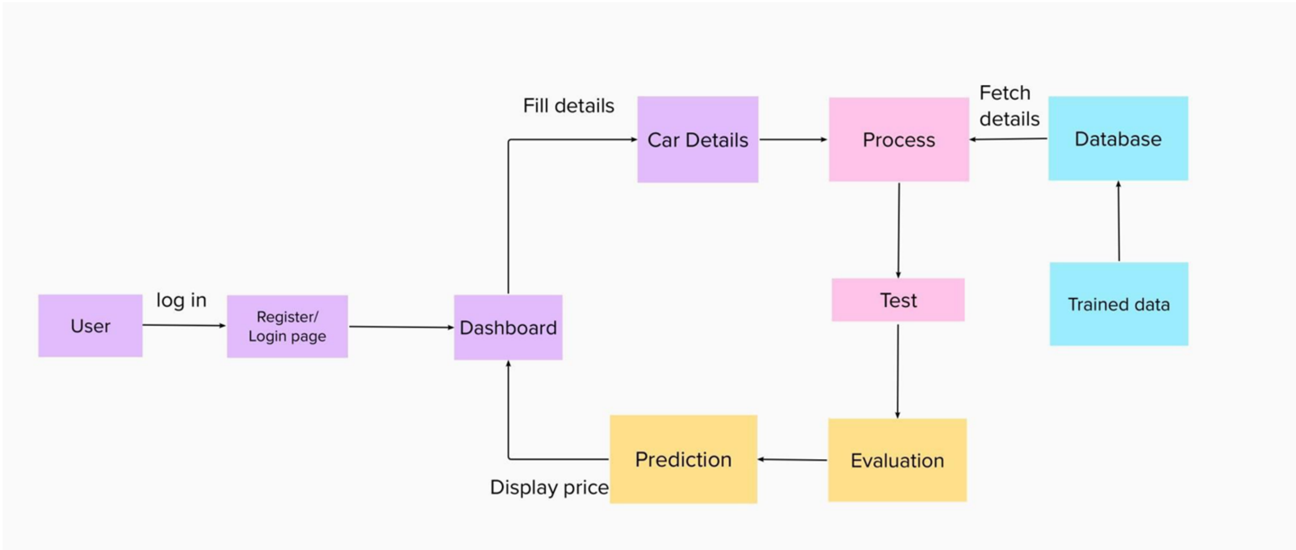
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	To create an UI makes as a user friendly, it makes a simple way to Understand
NFR-2	**Security**	Aware about fraudulent sites, it gives a fake information about the vehicle.
NFR-3	**Reliability**	Application must perform good and without failure
NFR-4	**Performance**	Website performance measures how quickly the pages of a website load and display in the web browser.
NFR-5	**Availability**	Website availability (also called website uptime) refers to the ability of the users to access and use a website or web service. A website's availability is typically communicated as a percentage for a given span of time.
NFR-6	**Scalability**	Application scalability is the ability of an application to handle a growing number of users and load, without compromising on performance and causing disruptions to user experience. To put it another way, scalability reflects the ability of the software to grow or change with the user's demands..

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

Data Flow Diagrams: A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



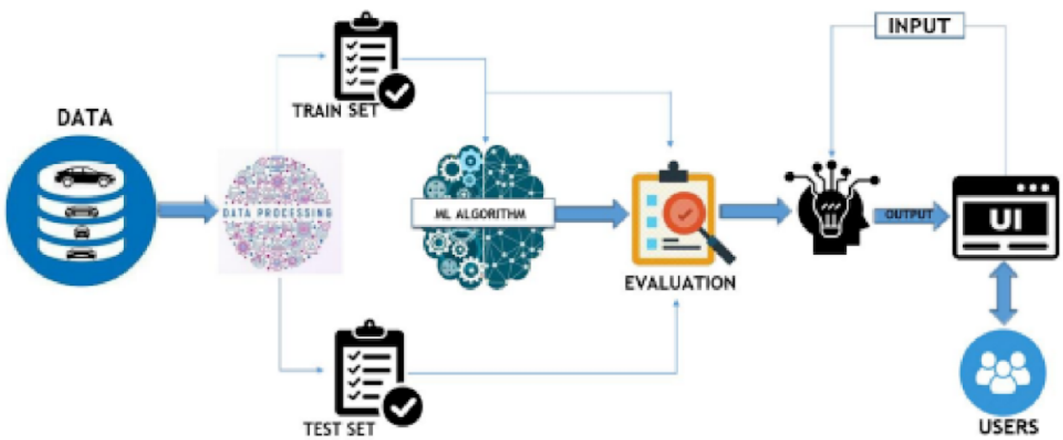
5.2 SOLUTION & TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to: Find the best tech solution to solve existing business problems. Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders. Define features, development phases, and solution requirements. Provide specifications according to which the solution is defined, managed, and delivered.

Project Design Phase-I
Solution Architecture

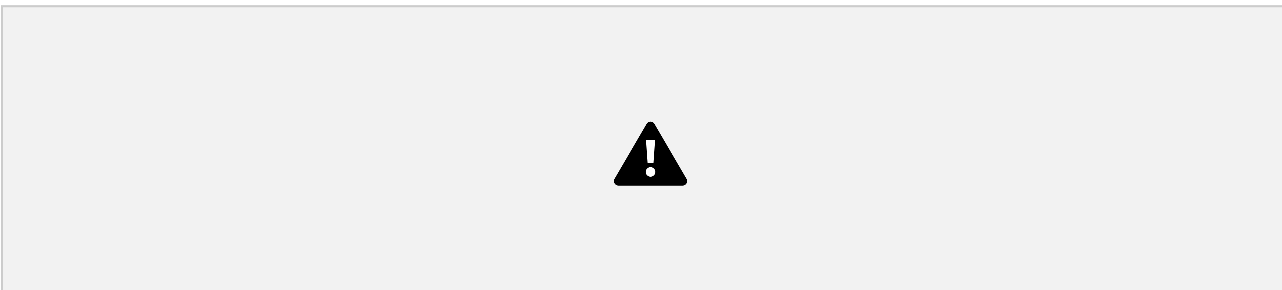
Date	26 September 2022
Team ID	PNT2022TMID10192
Project Name	Project – Car Resale Value Prediction
Maximum Marks	4 Marks

Solution Architecture Diagram for Car Resale Value Prediction



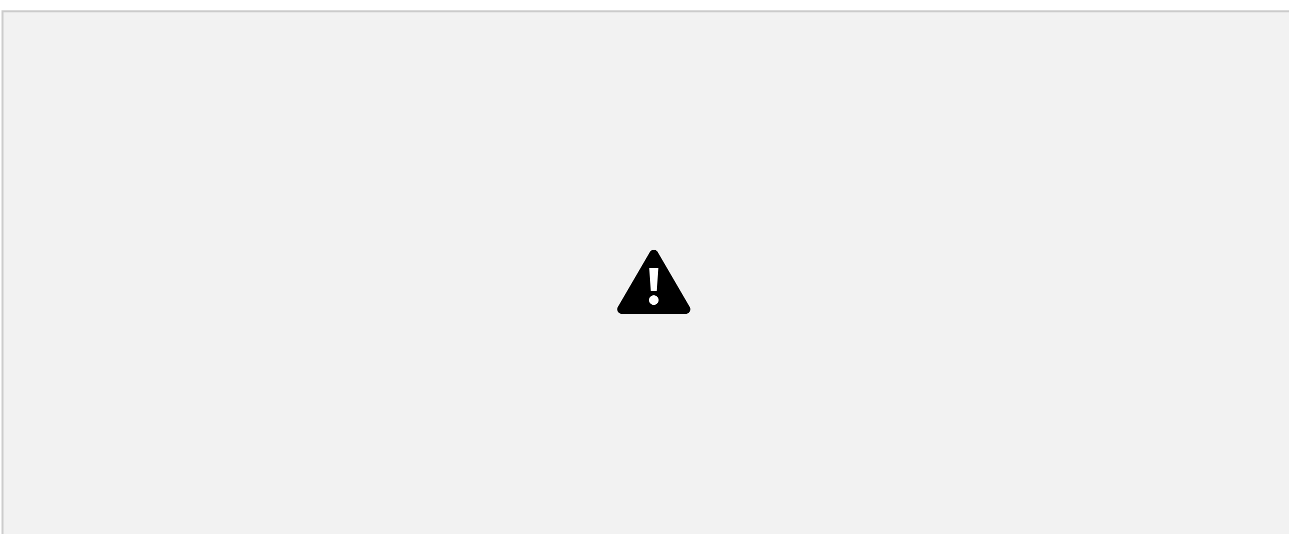
5.3 USER STORIES

A user story is an informal, natural language description of features of a software system. They are written from the perspective of an end user or user of a system, and may be recorded on index cards, post-it notes, or digitally in project management software. Depending on the project, user stories may be written by different stakeholders like client, user, manager, or development team.

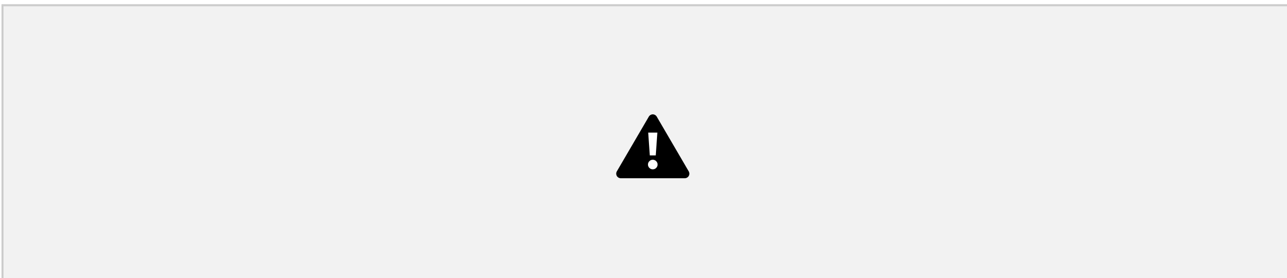


6. PROJECT PLANNING & SCHEDULING

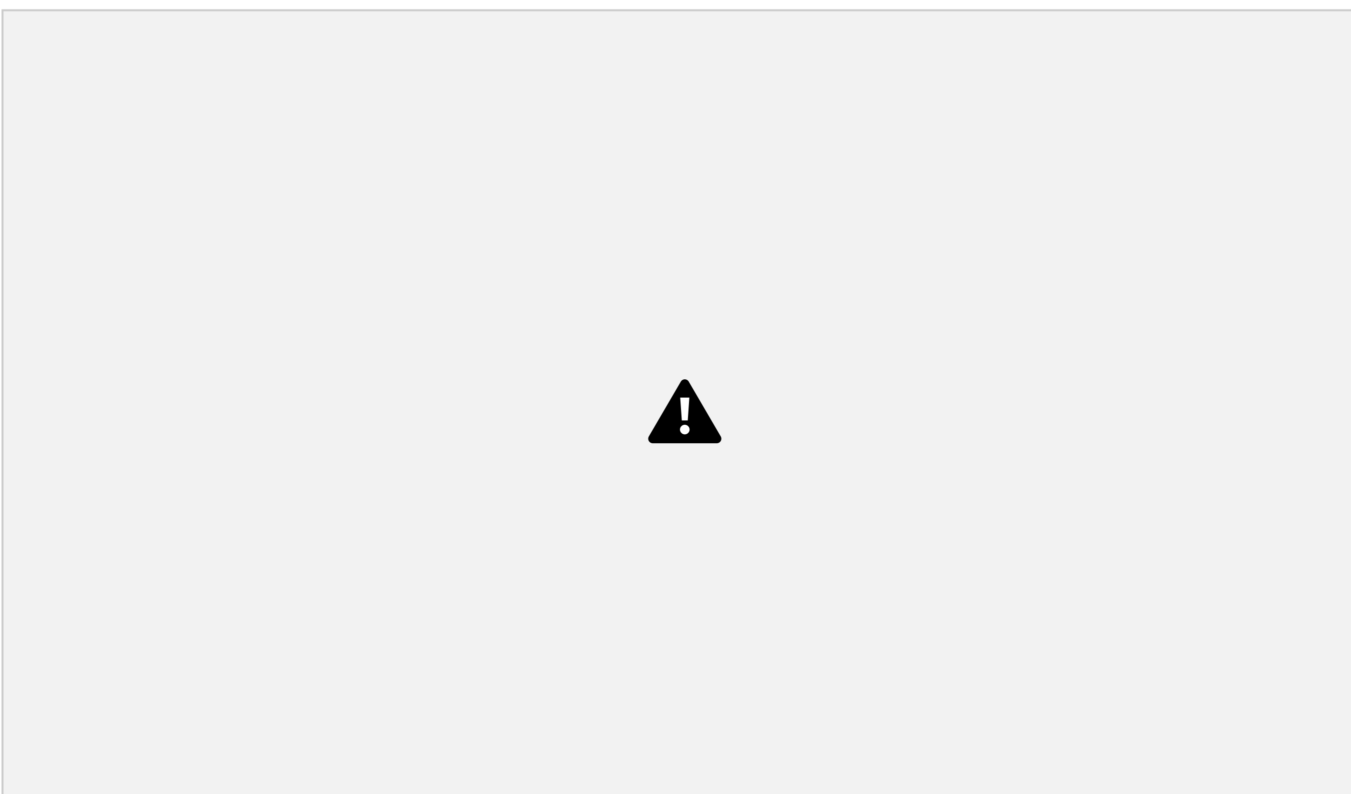
6.1 SPRINT PLANNING & ESTIMATION



6.2 SPRINT DELIVERY SCHEDULE



6.3 REPORT FROM JIRA



7. CODING & SOLUTION

7.1 FEATURE 1 Flask App

HANDLING MISSING VALUES*

In [ ]:

#after looking at the head of the ***HANDLING MISSING VALUES****

In [ ]:

```
#after looking at the head of the dataset we have NaN and missing values
#To find the of missing values in each column
#if present it shows true otherwise it shows false
data.isna().any()
```

Out[ ]:

```
dateCrawled          False
name                 False
seller               False
offerType            False
price                False
abtest               False
vehicleType          True
yearOfRegistration   False
gearbox              True
powerPS              False
model                True
kilometer            False
monthOfRegistration  False
fuelType             True
brand                False
notRepairedDamage    True
dateCreated          False
nrOfPictures         False
postalCode           False
lastSeen             False
dtype: bool
```

In [ ]:

```
#To find the count of missing values each column using sum function
data.isnull().sum()
```

Out[ ]:

```
dateCrawled          0
name                 0
seller               0
offerType            0
price                0
abtest               0
vehicleType          37869
yearOfRegistration   0
gearbox              20209
powerPS              0
model                20484
kilometer            0
```

monthOfRegistration	0
fuelType	33386
brand	0
notRepairedDamage	72060
dateCreated	0
nrOfPictures	0
postalCode	0
lastSeen	0

dtype: int64

In [ ]:

```
#Finding the description of the dataset using describe function like mean,median etc.,
data.describe()
```

Out[ ]:

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode
count	3.715280e+05	371528.000000	371528.000000	371528.000000	371528.000000	371528.0	371528.00000
mean	1.729514e+04	2004.577997	115.549477	125618.688228	5.734445	0.0	50820.66764
std	3.587954e+06	92.866598	192.139578	40112.337051	3.712412	0.0	25799.08247
min	0.000000e+00	1000.000000	0.000000	5000.000000	0.000000	0.0	1067.00000
25%	1.150000e+03	1999.000000	70.000000	125000.000000	3.000000	0.0	30459.00000
50%	2.950000e+03	2003.000000	105.000000	150000.000000	6.000000	0.0	49610.00000
75%	7.200000e+03	2008.000000	150.000000	150000.000000	9.000000	0.0	71546.00000
max	2.147484e+09	9999.000000	20000.000000	150000.000000	12.000000	0.0	99998.00000

In [ ]:

```
#Finding the mode of vehicleType column using mode function
```

data['vehicleType'].mode()

Out[ ]:

```
0    limousine
dtype: object
```

In [ ]:

```
#total value_counts in vehicleType column
data['vehicleType'].value_counts()
```

Out[ ]:

```
limousine     95894
kleinwagen    80023
kombi         67564
bus           30201
cabrio        22898
coupe         19015
suv           14707
andere         3357
Name: vehicleType, dtype: int64
```

In [ ]:

```
#Replacing all NaN values in vehicleType column using mode
data['vehicleType'].fillna("limousine",inplace=True)
```

In [ ]:

```
#Finding the mode of vehicleType column using mode function
data['gearbox'].mode()
```

Out[ ]:

```
0    manuell
dtype: object
```

In [ ]:

```
#Replacing all NaN values in gearbox column using mode
data['gearbox'].fillna("manuell",inplace=True)
```

In [ ]:

```
#Finding the mode of model column using mode function
data['model'].mode()
```

Out[ ]:

```
0    golf
dtype: object
```

In [ ]:

```
#Replacing all NaN values in model column using mode
data['model'].fillna("golf",inplace=True)
```

In [ ]:

```
#Finding the mode of fueltype column using mode function
data['fuelType'].mode()
```

Out[ ]:

```
0    benzin
dtype: object
```

In [ ]:

```
#Replacing all NaN values in model column using mode
data['fuelType'].fillna("benzin",inplace=True)
```

In [ ]:

```
#Finding the mode of notRepairedDamage column using mode function
data['notRepairedDamage'].mode()
```

Out[ ]:

```
0    nein
dtype: object
```

In [ ]:

```
#Replacing all NaN values in notRepairedDamage column using mode
data['notRepairedDamage'].fillna("nein",inplace=True)
```

In [ ]:

```
data.head()
```

Out[ ]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	limousine	1993	manuell	0	golf	150000	0	benzin	volkswagen	nein	2016-03-24 00:00:00	0	70435	2016-04-07 03:1

																				6:57
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	golf	125000	5	diesel	audi	ja	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163	grand	125000	8	diesel	jeep	nein	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:46
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75	golf	150000	6	benzin	volkswagen	nein	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17

4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69	fabia	90000	7	diesel	skoda	nein	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

OUTLIERS DETECTION AND REPLACING OUTLIERS

In [ ]:

```
sns.boxplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#finding the interquartilerange of price column
q1=data['price'].quantile(0.25)
q3=data['price'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
```

In [ ]:

```
#replacing the outliers of price column with mean
data['price']=np.where(data['price']>upper_bound,upper_bound,np.where(data['price']<lower_bound,upper_bound,data['price']))
```

In [ ]:

```
#boxplot for price column
sns.boxplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
```

following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning

Out[ ]:

In [ ]:

```
#finding the interquartilerange of kilometer column and replacing the outliers with mean
q1=data['kilometer'].quantile(0.25)
q3=data['kilometer'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['kilometer']=np.where(data['kilometer']>upper_bound,data['kilometer'].mean(),np.where(
data['kilometer']<lower_bound,data['kilometer'].mean(),data['kilometer']))
```

In [ ]:

```
#boxplot for kilometer column
sns.boxplot(data['kilometer'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning

Out[ ]:

In [ ]:

```
#finding the interquartilerange of powerPS column and replacing the outliers with lower_bound,upper_bound
q1=data['powerPS'].quantile(0.25)
q3=data['powerPS'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['powerPS']=np.where(data['powerPS']>upper_bound,upper_bound,np.where(data['powe
rPS']<lower_bound,lower_bound,data['powerPS']))
```

In [ ]:

```
#boxplot for powerPS column
sns.boxplot(data['powerPS'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument

will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning

Out[ ]:

In [ ]:

```
#finding the interquartilerange of yearOfRegistration column and replacing the outliers with mean
q1=data['yearOfRegistration'].quantile(0.25)
q3=data['yearOfRegistration'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['yearOfRegistration']=np.where(data['yearOfRegistration']>upper_bound,data['yearOfRegistration'].mode(),np.where(data['yearOfRegistration']<lower_bound,data['yearOfRegistration'].mode(),data['yearOfRegistration']))
```

In [ ]:

```
#boxplot for yearOfRegistration column
sns.boxplot(data['yearOfRegistration'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning

Out[ ]:

In [ ]:

```
#boxplot for monthOfRegistation column
sns.boxplot(data['monthOfRegistration'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
FutureWarning

Out[ ]:

In [ ]:

```
#Reading the first five rows of cleaned dataset using head function
data.head()
```

Out[ ]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480.0	test	limousine	1993	manuell	0.0	golf	150000.0	0	benzin	volkswagen	nein	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	16275.0	test	coupe	2011	manuell	190.0	golf	125000.0	5	diesel	audi	ja	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800.0	test	suv	2004	automatik	163.0	grand	125000.0	8	diesel	jeep	nein	2016-03-14 00:00:00	0	90480	2016-04-05 12:4

																				7:46
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500.0	test	kleinwagen	2001	manuell	75.0	golf	150000.0	6	benzin	volkswagen	nein	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600.0	test	kleinwagen	2008	manuell	69.0	fabia	90000.0	7	diesel	skoda	nein	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

EXPLORATORY DATA ANALYSIS

Exploring Categorical Features

In [ ]:

```
#list of all categorical columns
list(data.select_dtypes('object'))
```

Out[ ]:

```
['dateCrawled',
 'name',
 'seller',
 'offerType',
```

```
 'abtest',
 'vehicleType',
 'gearbox',
 'model',
 'fuelType',
 'brand',
 'notRepairedDamage',
 'dateCreated',
'lastSeen']
```

In [ ]:

```
data['seller'].value_counts()
```

Out[ ]:

```
privat       371525
gewerblich        3
Name: seller, dtype: int64
```

In [ ]:

```
#counting public and gewerblich types in seller column using countplot
sns.countplot(data['seller'],palette='coolwarm',saturation=0.9)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['abtest'].value_counts()
```

Out[ ]:

```
test       192585
control    178943
Name: abtest, dtype: int64
```

In [ ]:

```
#counting the percentage of different types in abtest column using pie chart
plt.pie(data['abtest'].value_counts(),startangle=90,labels=['test','control'],shadow=True,autopct='%1.2f%%')
plt.legend()
plt.title("abtest")
```

Out[ ]:

```
Text(0.5, 1.0, 'abtest')
```

In [ ]:

```
data['offerType'].value_counts()
```

Out[ ]:

```
Angebot    371516
Gesuch         12
Name: offerType, dtype: int64
```

In [ ]:

```
#counting angebot and gesuch types in offerType column using countplot
sns.countplot(data['offerType'],palette='spring')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['vehicleType'].value_counts()
```

Out[ ]:

```
limousine     133763
kleinwagen     80023
kombi          67564
bus            30201
cabrio         22898
coupe          19015
suv            14707
andere          3357
Name: vehicleType, dtype: int64
```

In [ ]:

```
#count of each type in vehicleType column
sns.countplot(data['vehicleType'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#count of each type in gearbox column
sns.countplot(data['gearbox'],palette='pastel')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['model'].value_counts()
```

Out[ ]:

```
golf              50554
andere            26400
3er               20567
polo              13092
corsa             12573
                  ...
serie_2               8
rangerover            6
serie_3               4
serie_1               2
discovery_sport       1
Name: model, Length: 251, dtype: int64
```

In [ ]:

```
#top 10 models in model column
plt.figure(figsize =(15,6))
sns.countplot(data['model'].value_counts().head(10))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['fuelType'].value_counts()
```

Out[ ]:

```
benzin     257243
diesel     107746
lpg          5378
cng           571
hybrid        278
andere        208
elektro       104
Name: fuelType, dtype: int64
```

In [ ]:

```
plt.figure(figsize =(15,6))
sns.countplot(data['fuelType'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['brand'].value_counts().head()
```

Out[ ]:

```
volkswagen       79640
bmw              40274
opel             40136
mercedes_benz    35309
audi             32873
Name: brand, dtype: int64
```

In [ ]:

```
#count of eaach brand in brand column
plt.figure(figsize =(10,6))
sns.countplot(data['brand'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['notRepairedDamage'].value_counts()
```

Out[ ]:

```
nein    335242
ja       36286
Name: notRepairedDamage, dtype: int64
```

In [ ]:

```
sns.countplot(data['notRepairedDamage'],palette='spring')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
a=list(data.select_dtypes('number'))
for i in a:
   fig = plt.figure(figsize=(9, 6))
   ax = fig.gca()
   feature = data[i]
   feature.hist(bins=50, ax = ax)
   ax.axvline(feature.mean(), color='magenta', linestyle='dashed', linewidth=2)
   ax.axvline(feature.median(), color='cyan', linestyle='dashed', linewidth=2)
   ax.set_title(i)
plt.show()
```

In [ ]:

```
#correlation of  dataset using correaltion function
correlation=data.corr()
correlation
```

Out[ ]:

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode

price	*1.000000*	*0.498059*	*0.547702*	*-0.333261*	*0.107701*	*NaN*	*0.092355*
yearOfRegistration	*0.498059*	*1.000000*	*0.148963*	*-0.250738*	*0.032619*	*NaN*	*0.036769*
powerPS	*0.547702*	*0.148963*	*1.000000*	*-0.012199*	*0.133211*	*NaN*	*0.087730*
kilometer	*-0.333261*	*-0.250738*	*-0.012199*	*1.000000*	*-0.022828*	*NaN*	*-0.028500*
monthOfRegistration	*0.107701*	*0.032619*	*0.133211*	*-0.022828*	*1.000000*	*NaN*	*0.014963*
nrOfPictures	*NaN*	*NaN*	*NaN*	*NaN*	*NaN*	*NaN*	*NaN*
postalCode	*0.092355*	*0.036769*	*0.087730*	*-0.028500*	*0.014963*	*NaN*	*1.000000*

In [ ]:

```
#exploring the correlation using heatmap
plt.figure(figsize=(15,10))
sns.heatmap(correlation, vmax=1, square=True,annot=True,cmap='cubehelix')
```

Out[ ]:

1.SELLER VS PRICE

In [ ]:

```
plt.figure(figsize=(8,4))
sns.barplot(x='seller',y='price',data=data,palette='dark')
```

Out[ ]:

2.VEHICLETYPE VS PRICE

In [ ]:

```
plt.figure(figsize=(8,6))
sns.barplot(x='vehicleType',y='price',data=data,ci=100,capsize=0.3,saturation=0.8)
```

Out[ ]:

3.MODEL VS PRICE

In [ ]:

```
plt.figure(figsize=(15,5))
sns.barplot(x='model',y='price',data=data)
```

Out[ ]:

4.KILOMETER VS PRICE

In [ ]:

```
sns.kdeplot(x='kilometer',y='price',data=data,palette='husl')
```

Out[ ]:

5.BRAND VS PRICE

In [ ]:

```
plt.figure(figsize=(25,5))
sns.barplot(x='brand',y='price',data=data)
```

Out[ ]:

6. YEAR OF REGISTRATION VS PRICE

In [ ]:

```
plt.figure(figsize=(15,5))
sns.stripplot(x='yearOfRegistration',y='price',data=data)
```

Out[ ]:

7.FUEL TYPE VS PRICE

In [ ]:

```
sns.barplot(x='fuelType',y='price',data=data)
```

Out[ ]:

8.GEARBOX VS KILOMETER

In [ ]:

```
sns.pointplot(x='gearbox',y='kilometer',hue='fuelType',data=data,ci=99,saturation=0.8,capsize=0.3)
```

9.KILOMETER VS PRICE

In [ ]:

```
sns.scatterplot(x='fuelType',y='kilometer',data=data)
```

Out[ ]:

DISTRIBUTION PLOT

In [ ]:

```
#examing the distribution of price column using distplot in seaborn library
plt.figure(figsize=(15,5))
sns.distplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[ ]:

In [ ]:

```
parameters={'seller':{'privat':0,'gewerblich':1},
        'abtest':{'test':0,'control':1},
        'notRepairedDamage':{'nein':0,'ja':1},
        'vehicleType':{'limousine':0,'kleinwagen':1,'kombi':2,'bus':3,'cabrio':4,'coupe':5,'suv':6,'andere':7},
        'fuelType':{'benzin':0,'diesel':1,'lpg':2,'cng':3 ,'hybrid':4,'andere':5,'elektro':6}}
data_df=data.replace(parameters)
data_df.head()
```

Out[ ]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
0	2016-03-24 11:52:17	Golf_3_1.6	0	Angebot	480.0	0	0	1993	manuell	0.0	golf	150000.0	0	0	volkswagen	0	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	0	Angebot	16275.0	0	5	2011	manuell	190.0	golf	125000.0	5	1	audi	1	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	0	Angebot	9800.0	0	6	2004	automatik	163.0	grand	125000.0	8	1	jeep	0	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:4

6

3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	0	Angebot	1500.0	0	1	2001	manuell	75.0	golf	150000.0	6	0	volkswagen	0	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	0	Angebot	3600.0	0	1	2008	manuell	69.0	fabia	90000.0	7	1	skoda	0	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

In [ ]:

```
#converting all catogorical columns into numerical columns using get_dummies function
Fe_df_cleaned=pd.get_dummies(data_df,columns=['offerType','gearbox'],drop_first=True)
Fe_df_cleaned.head()
```

Out[ ]:

	dateCrawled	name	seller	price	abtest	vehicleType	yearOfRegistration	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen	offerType_Gesuch	gearbox_manuell

0	*2016-03-24 11:52:17*	*Golf_3_1.6*	*0*	*480.0*	*0*	*0*	*1993*	*0.0*	*golf*	*150000.0*	*0*	*0*	*volkswagen*	*0*	*2016-03-24 00:00:00*	*0*	*70435*	*2016-04-07 03:16:57*	*0*	*1*
1	*2016-03-24 10:58:45*	*A5_Sportback_2.7_Tdi*	*0*	*16275.0*	*0*	*5*	*2011*	*190.0*	*golf*	*125000.0*	*5*	*1*	*audi*	*1*	*2016-03-24 00:00:00*	*0*	*66954*	*2016-04-07 01:46:50*	*0*	*1*
2	*2016-03-14 12:52:21*	*Jeep_Grand_Cherokee_"Overland"*	*0*	*9800.0*	*0*	*6*	*2004*	*163.0*	*grand*	*125000.0*	*8*	*1*	*jeep*	*0*	*2016-03-14 00:00:00*	*0*	*90480*	*2016-04-05 12:4*	*0*	*0*

																		7:46		
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	0	1500.0	0	1	2001	75.0	golf	150000.0	6	0	volkswagen	0	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17	0	1
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	0	3600.0	0	1	2008	69.0	fabia	90000.0	7	1	skoda	0	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21	0	1

In [ ]:

```
#shape of the dataset after label encoding
Fe_df_cleaned.shape
```

Out[ ]:

(371528, 20)

In [ ]:

```
Fe_df_cleaned.columns
```

Out[ ]:

```
Index(['dateCrawled', 'name', 'seller', 'price', 'abtest', 'vehicleType',
       'yearOfRegistration', 'powerPS', 'model', 'kilometer',
       'monthOfRegistration', 'fuelType', 'brand', 'notRepairedDamage',
       'dateCreated', 'nrOfPictures', 'postalCode', 'lastSeen',
       'offerType_Gesuch', 'gearbox_manuell'],
dtype='object')
```

In [ ]:

```
#removing unncessary columns in the dataset
main_df=Fe_df_cleaned.drop(columns=['dateCrawled','dateCreated','name','lastSeen','brand','model'],axis=1)
main_df.head()
```

Out[ ]:

	seller	price	abtest	vehicleType	yearOfRegistration	powerPS	kilometer	monthOfRegistration	fuelType	notRepairedDamage	nrOfPictures	postalCode	offerType_Gesuch	gearbox_manuell
0	0	480.0	0	0	1993	0.0	150000.0	0	0	0	0	70435	0	1
1	0	16275.0	0	5	2011	190.0	125000.0	5	1	1	0	66954	0	1
2	0	9800.0	0	6	2004	163.0	125000.0	8	1	0	0	90480	0	0
3	0	1500.0	0	1	2001	75.0	150000.0	6	0	0	0	91074	0	1
4	0	3600.0	0	1	2008	69.0	90000.0	7	1	0	0	60437	0	1

In [ ]:

```
#multivariate analysis
plt.figure(figsize=(15,5))
sns.pairplot(data)
```

Out[ ]:

In [ ]:

```
#dividing the dataset into dependent and independent feature
Independent=main_df.drop(['price'],axis=1)
Dependent=main_df['price']
Independent.head()
```

Out[ ]:

	seller	abtest	vehicleType	yearOfRegistration	powerPS	kilometer	monthOfRegistration	fuelType	notRepairedDamage	nrOfPictures	postalCode	offerType_Gesuch	gearbox_manuell
0	0	0	0	1993	0.0	150000.0	0	0	0	0	70435	0	1
1	0	0	5	2011	190.0	125000.0	5	1	1	0	66954	0	1
2	0	0	6	2004	163.0	125000.0	8	1	0	0	90480	0	0
3	0	0	1	2001	75.0	150000.0	6	0	0	0	91074	0	1
4	0	0	1	2008	69.0	90000.0	7	1	0	0	60437	0	1

In [ ]:

```
Dependent.head()
```

Out[ ]:

```
0      480.0
1    16275.0
2     9800.0
3     1500.0
4     3600.0
Name: price, dtype: float64
```

dataset we have NaN and missing values
#To find the of missing values in each column
#if present it shows true otherwise it shows false

```
data.isna().any()
```

Out[ ]:

```
dateCrawled           False
name                  False
seller                False
offerType             False
price                 False
abtest                False
vehicleType           True
yearOfRegistration    False
gearbox               True
powerPS               False
model                 True
kilometer             False
monthOfRegistration   False
fuelType              True
brand                 False
notRepairedDamage     True
dateCreated           False
nrOfPictures          False
postalCode            False
lastSeen              False
dtype: bool
```

In [ ]:

```
#To find the count of missing values each column using sum function
data.isnull().sum()
```

Out[ ]:

```
dateCrawled               0
name                      0
seller                    0
offerType                 0
price                     0
abtest                    0
vehicleType           37869
yearOfRegistration        0
gearbox               20209
powerPS                   0
model                 20484
kilometer                 0
monthOfRegistration       0
fuelType              33386
brand                     0
notRepairedDamage     72060
```

dateCreated	0
nrOfPictures	0
postalCode	0
lastSeen	0

dtype: int64

In [ ]:

```
#Finding the description of the dataset using describe function like mean,median etc.,
data.describe()
```

Out[ ]:

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode
count	3.715280e+05	371528.000000	371528.000000	371528.000000	371528.000000	371528.0	371528.00000
mean	1.729514e+04	2004.577997	115.549477	125618.688228	5.734445	0.0	50820.66764
std	3.587954e+06	92.866598	192.139578	40112.337051	3.712412	0.0	25799.08247
min	0.000000e+00	1000.000000	0.000000	5000.000000	0.000000	0.0	1067.00000
25%	1.150000e+03	1999.000000	70.000000	125000.000000	3.000000	0.0	30459.00000
50%	2.950000e+03	2003.000000	105.000000	150000.000000	6.000000	0.0	49610.00000
75%	7.200000e+03	2008.000000	150.000000	150000.000000	9.000000	0.0	71546.00000
max	2.147484e+09	9999.000000	20000.000000	150000.000000	12.000000	0.0	99998.00000

In [ ]:

```
#Finding the mode of vehicleType column using mode function
data['vehicleType'].mode()
```

Out[ ]:

0 limousine

dtype: object

In [ ]:

```
#total value_counts in vehicleType column
data['vehicleType'].value_counts()
```

Out[ ]:

```
limousine     95894
kleinwagen    80023
kombi         67564
bus           30201
cabrio        22898
coupe         19015
suv           14707
andere         3357
```

Name: vehicleType, dtype: int64

In [ ]:

```
#Replacing all NaN values in vehicleType column using mode
data['vehicleType'].fillna("limousine",inplace=True)
```

In [ ]:

```
#Finding the mode of vehicleType column using mode function
data['gearbox'].mode()
```

Out[ ]:

```
0    manuell
```

dtype: object

In [ ]:

```
#Replacing all NaN values in gearbox column using mode
data['gearbox'].fillna("manuell",inplace=True)
```

In [ ]:

```
#Finding the mode of model column using mode function
data['model'].mode()
```

Out[ ]:

```
0    golf
```

dtype: object

In [ ]:

```
#Replacing all NaN values in model column using mode
data['model'].fillna("golf",inplace=True)
```

In [ ]:

```
#Finding the mode of fueltype column using mode function
data['fuelType'].mode()
```

Out[ ]:

```
0    benzin
dtype: object
```

In [ ]:

```
#Replacing all NaN values in model column using mode
data['fuelType'].fillna("benzin",inplace=True)
```

In [ ]:

```
#Finding the mode of notRepairedDamage column using mode function
data['notRepairedDamage'].mode()
```

Out[ ]:

```
0    nein
dtype: object
```

In [ ]:

```
#Replacing all NaN values in notRepairedDamage column using mode
data['notRepairedDamage'].fillna("nein",inplace=True)
```

In [ ]:

```
data.head()
```

Out[ ]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	limousine	1993	manuell	0	golf	150000	0	benzin	volkswagen	nein	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57

1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	golf	125000	5	diesel	audi	ja	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163	grand	125000	8	diesel	jeep	nein	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:46
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75	golf	150000	6	benzin	volkswagen	nein	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
4	2016-03-31 17:25:	Skoda_Fabia_1.4_TDI_PD_Classic	priva	Angebot	3600	test	kleinwagen	2008	manuell	69	fabia	90000	7	diesel	skoda	nein	2016-03-31 00:00:00:	0	60437	2016-04-

20	t	00	0 6 1 0: 1 7: 2 1

OUTLIERS DETECTION AND REPLACING OUTLIERS

In [ ]:

```
sns.boxplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#finding the interquartilerange of price column
q1=data['price'].quantile(0.25)
q3=data['price'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
```

In [ ]:

```
#replacing the outliers of price column with mean
data['price']=np.where(data['price']>upper_bound,upper_bound,np.where(data['price']<lower_bound,upper_bound,data['price']))
```

In [ ]:

```
#boxplot for price column
sns.boxplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#finding the interquartilerange of kilometer column and replacing the outliers with mean
q1=data['kilometer'].quantile(0.25)
q3=data['kilometer'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['kilometer']=np.where(data['kilometer']>upper_bound,data['kilometer'].mean(),np.where(
data['kilometer']<lower_bound,data['kilometer'].mean(),data['kilometer']))
```

In [ ]:

```
#boxplot for kilometer column
sns.boxplot(data['kilometer'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#finding the interquartilerange of powerPS column and replacing the outliers with lower_bound,upper_bound
q1=data['powerPS'].quantile(0.25)
q3=data['powerPS'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['powerPS']=np.where(data['powerPS']>upper_bound,upper_bound,np.where(data['powe
rPS']<lower_bound,lower_bound,data['powerPS']))
```

In [ ]:

```
#boxplot for powerPS column
sns.boxplot(data['powerPS'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#finding the interquartilerange of yearOfRegistration column and replacing the outliers with mean
q1=data['yearOfRegistration'].quantile(0.25)
q3=data['yearOfRegistration'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['yearOfRegistration']=np.where(data['yearOfRegistration']>upper_bound,data['yearOfRegistration'].mode(),np.where(data['yearOfRegistration']<lower_bound,data['yearOfRegistration'].mode(),data['yearOfRegistration']))
```

In [ ]:

```
#boxplot for yearOfRegistration column
sns.boxplot(data['yearOfRegistration'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#boxplot for monthOfRegistation column
sns.boxplot(data['monthOfRegistration'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
#Reading the first five rows of cleaned dataset using head function
data.head()
```

Out[ ]:

dat eCr	name	s e	of fe	p ri	a b	ve hic	yearO fRegis	g ea	p o	m o	kil o	month OfRegi	fu el	br an	notRe paired	dat eCr	nr Of	po sta	la st

	awled		ller	rType	ce	test	leType	tration	rbox	werPS	del	meter	stration	Type	d	Damage	eated	Pictures	ICode	Seen
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480.0	test	limousine	1993	manuell	0.0	golf	150000.0	0	benzin	volkswagen	nein	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	16275.0	test	coupe	2011	manuell	190.0	golf	125000.0	5	diesel	audi	ja	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800.0	test	suv	2004	automatik	163.0	grand	125000.0	8	diesel	jeep	nein	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:46

3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500.0	test	kleinwagen	2001	manuell	75.0	golf	150000.0	6	benzin	volkswagen	nein	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600.0	test	kleinwagen	2008	manuell	69.0	fabia	90000.0	7	diesel	skoda	nein	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

EXPLORATORY DATA ANALYSIS

Exploring Categorical Features

In [ ]:

```
#list of all categorical columns
list(data.select_dtypes('object'))
```

Out[ ]:

```
['dateCrawled',
 'name',
 'seller',
 'offerType',
 'abtest',
 'vehicleType',
 'gearbox',
 'model',
 'fuelType',
```

```
 'brand',
 'notRepairedDamage',
 'dateCreated',
 'lastSeen']
```

In [ ]:

```
data['seller'].value_counts()
```

Out[ ]:

```
privat       371525
gewerblich        3
Name: seller, dtype: int64
```

In [ ]:

```
#counting public and gewerblich types in seller column using countplot
sns.countplot(data['seller'],palette='coolwarm',saturation=0.9)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['abtest'].value_counts()
```

Out[ ]:

```
test       192585
control    178943
Name: abtest, dtype: int64
```

In [ ]:

```
#counting the percentage of different types in abtest column using pie chart
plt.pie(data['abtest'].value_counts(),startangle=90,labels=['test','control'],shadow=True,autopct='%1.2f%%')
plt.legend()
plt.title("abtest")
```

Out[ ]:

```
Text(0.5, 1.0, 'abtest')
```

In [ ]:

```
data['offerType'].value_counts()
```

Out[ ]:

Angebot   371516
Gesuch   12

Name: offerType, dtype: int64

In [ ]:

```
#counting angebot and gesuch types in offerType column using countplot
sns.countplot(data['offerType'],palette='spring')
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning

Out[ ]:

In [ ]:

```
data['vehicleType'].value_counts()
```

Out[ ]:

limousine   133763
kleinwagen   80023
kombi   67564
bus   30201
cabrio   22898
coupe   19015
suv   14707
andere   3357

Name: vehicleType, dtype: int64

In [ ]:

```
#count of each type in vehicleType column
sns.countplot(data['vehicleType'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning

Out[ ]:

In [ ]:

```
#count of each type in gearbox column
sns.countplot(data['gearbox'],palette='pastel')
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the

following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning

Out[ ]:

In [ ]:

```
data['model'].value_counts()
```

Out[ ]:

```
golf              50554
andere            26400
3er               20567
polo              13092
corsa             12573
                  ...
serie_2               8
rangerover            6
serie_3               4
serie_1               2
discovery_sport       1
Name: model, Length: 251, dtype: int64
```

In [ ]:

```
#top 10 models in model column
plt.figure(figsize =(15,6))
sns.countplot(data['model'].value_counts().head(10))
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning

Out[ ]:

In [ ]:

```
data['fuelType'].value_counts()
```

Out[ ]:

```
benzin    257243
diesel    107746
lpg         5378
cng          571
hybrid       278
```

```
andere      208
elektro     104
Name: fuelType, dtype: int64
```

In [ ]:

```
plt.figure(figsize =(15,6))
sns.countplot(data['fuelType'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['brand'].value_counts().head()
```

Out[ ]:

```
volkswagen       79640
bmw              40274
opel             40136
mercedes_benz    35309
audi             32873
Name: brand, dtype: int64
```

In [ ]:

```
#count of eaach brand in brand column
plt.figure(figsize =(10,6))
sns.countplot(data['brand'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
data['notRepairedDamage'].value_counts()
```

Out[ ]:

```
nein    335242
ja       36286
```

Name: notRepairedDamage, dtype: int64

In [ ]:

```
sns.countplot(data['notRepairedDamage'],palette='spring')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:

In [ ]:

```
a=list(data.select_dtypes('number'))
for i in a:
   fig = plt.figure(figsize=(9, 6))
   ax = fig.gca()
   feature = data[i]
   feature.hist(bins=50, ax = ax)
   ax.axvline(feature.mean(), color='magenta', linestyle='dashed', linewidth=2)
   ax.axvline(feature.median(), color='cyan', linestyle='dashed', linewidth=2)
   ax.set_title(i)
plt.show()
```

In [ ]:

```
#correlation of  dataset using correaltion function
correlation=data.corr()
correlation
```

Out[ ]:

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode
price	1.000000	0.498059	0.547702	-0.333261	0.107701	NaN	0.092355

yearOfRegistration	0.498059	1.000000	0.148963	-0.250738	0.032619	NaN	0.036769
powerPS	0.547702	0.148963	1.000000	-0.012199	0.133211	NaN	0.087730
kilometer	-0.333261	-0.250738	-0.012199	1.000000	-0.022828	NaN	-0.028500
monthOfRegistration	0.107701	0.032619	0.133211	-0.022828	1.000000	NaN	0.014963
nrOfPictures	NaN	NaN	NaN	NaN	NaN	NaN	NaN
postalCode	0.092355	0.036769	0.087730	-0.028500	0.014963	NaN	1.000000

In [ ]:

```
#exploring the correlation using heatmap
plt.figure(figsize=(15,10))
sns.heatmap(correlation, vmax=1, square=True,annot=True,cmap='cubehelix')
```

Out[ ]:

1.SELLER VS PRICE

In [ ]:

```
plt.figure(figsize=(8,4))
sns.barplot(x='seller',y='price',data=data,palette='dark')
```

Out[ ]:

2.VEHICLETYPE VS PRICE

In [ ]:

```
plt.figure(figsize=(8,6))
sns.barplot(x='vehicleType',y='price',data=data,ci=100,capsize=0.3,saturation=0.8)
```

Out[ ]:

3.MODEL VS PRICE

In [ ]:

```
plt.figure(figsize=(15,5))
sns.barplot(x='model',y='price',data=data)
```

Out[ ]:

4.KILOMETER VS PRICE

In [ ]:

```
sns.kdeplot(x='kilometer',y='price',data=data,palette='husl')
```

Out[ ]:

5.BRAND VS PRICE

In [ ]:

```
plt.figure(figsize=(25,5))
sns.barplot(x='brand',y='price',data=data)
```

Out[ ]:

6. YEAR OF REGISTRATION VS PRICE

In [ ]:

```
plt.figure(figsize=(15,5))
sns.stripplot(x='yearOfRegistration',y='price',data=data)
```

Out[ ]:

7.FUEL TYPE VS PRICE

In [ ]:

```
sns.barplot(x='fuelType',y='price',data=data)
```

Out[ ]:

8.GEARBOX VS KILOMETER

In [ ]:

```
sns.pointplot(x='gearbox',y='kilometer',hue='fuelType',data=data,ci=99,saturation=0.8,capsize=0.3)
```

9.KILOMETER VS PRICE

In [ ]:

```
sns.scatterplot(x='fuelType',y='kilometer',data=data)
```

Out[ ]:

DISTRIBUTION PLOT

In [ ]:

```
#examing the distribution of price column using distplot in seaborn library
plt.figure(figsize=(15,5))
sns.distplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[ ]:

In [ ]:

```
parameters={'seller':{'privat':0,'gewerblich':1},
         'abtest':{'test':0,'control':1},
         'notRepairedDamage':{'nein':0,'ja':1},

'vehicleType':{'limousine':0,'kleinwagen':1,'kombi':2,'bus':3,'cabrio':4,'coupe':5,'suv':6,'andere':7},
         'fuelType':{'benzin':0,'diesel':1,'lpg':2,'cng':3 ,'hybrid':4,'andere':5,'elektro':6}}
data_df=data.replace(parameters)
data_df.head()
```

Out[ ]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
0	2016-03-24 11:52:17	Golf_3_1.6	0	Angebot	480.0	0	0	1993	manuell	0.0	golf	150000.0	0	0	volkswagen	0	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	0	Angebot	16275.0	0	5	2011	manuell	190.0	golf	125000.0	5	1	audi	1	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	0	Angebot	9800.0	0	6	2004	automatik	163.0	grand	125000.0	8	1	jeep	0	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:4

6

3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	0	Angebot	1500.0	0	1	2001	manuell	75.0	golf	150000.0	6	0	volkswagen	0	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	0	Angebot	3600.0	0	1	2008	manuell	69.0	fabia	90000.0	7	1	skoda	0	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

In [ ]:

```
#converting all catogorical columns into numerical columns using get_dummies function
Fe_df_cleaned=pd.get_dummies(data_df,columns=['offerType','gearbox'],drop_first=True)
Fe_df_cleaned.head()
```

Out[ ]:

	dateCrawled	name	seller	price	abtest	vehicleType	yearOfRegistration	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen	offerType_Gesuch	gearbox_manuell

0	2016-03-24 11:52:17	Golf_3_1.6	0	480.0	0	0	1993	0.0	golf	150000.0	0	0	volkswagen	0	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57	0	1
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	0	16275.0	0	5	2011	190.0	golf	125000.0	5	1	audi	1	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50	0	1
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	0	9800.0	0	6	2004	163.0	grand	125000.0	8	1	jeep	0	2016-03-14 00:00:00	0	90480	2016-04-05 12:4	0	0

																		7:46		
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	0	1500.0	0	1	2001	75.0	golf	150000.0	6	0	volkswagen	0	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17	0	1
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	0	3600.0	0	1	2008	69.0	fabia	90000.0	7	1	skoda	0	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21	0	1

In [ ]:

```
#shape of the dataset after label encoding
Fe_df_cleaned.shape
```

Out[ ]:

(371528, 20)

In [ ]:

```
Fe_df_cleaned.columns
```

Out[ ]:

```
Index(['dateCrawled', 'name', 'seller', 'price', 'abtest', 'vehicleType',
       'yearOfRegistration', 'powerPS', 'model', 'kilometer',
       'monthOfRegistration', 'fuelType', 'brand', 'notRepairedDamage',
       'dateCreated', 'nrOfPictures', 'postalCode', 'lastSeen',
       'offerType_Gesuch', 'gearbox_manuell'],
dtype='object')
```

In [ ]:

```
#removing unncessary columns in the dataset
main_df=Fe_df_cleaned.drop(columns=['dateCrawled','dateCreated','name','lastSeen','brand',
'model'],axis=1)
main_df.head()
```

Out[ ]:

	seller	price	abtest	vehicleType	yearOfRegistration	powerPS	kilometer	monthOfRegistration	fuelType	notRepairedDamage	nrOfPictures	postalCode	offerType_Gesuch	gearbox_manuell
0	0	480.0	0	0	1993	0.0	150000.0	0	0	0	0	70435	0	1
1	0	16275.0	0	5	2011	190.0	125000.0	5	1	1	0	66954	0	1
2	0	9800.0	0	6	2004	163.0	125000.0	8	1	0	0	90480	0	0
3	0	1500.0	0	1	2001	75.0	150000.0	6	0	0	0	91074	0	1
4	0	3600.0	0	1	2008	69.0	90000.0	7	1	0	0	60437	0	1

In [ ]:

```
#multivariate analysis
plt.figure(figsize=(15,5))
sns.pairplot(data)
```

Out[ ]:

In [ ]:

```
#dividing the dataset into dependent and independent feature
Independent=main_df.drop(['price'],axis=1)
Dependent=main_df['price']
Independent.head()
```

Out[ ]:

	seller	abtest	vehicleType	yearOfRegistration	powerPS	kilometer	monthOfRegistration	fuelType	notRepairedDamage	nrOfPictures	postalCode	offerType_Gesuch	gearbox_manuell
0	0	0	0	1993	0.0	150000.0	0	0	0	0	70435	0	1
1	0	0	5	2011	190.0	125000.0	5	1	1	0	66954	0	1
2	0	0	6	2004	163.0	125000.0	8	1	0	0	90480	0	0
3	0	0	1	2001	75.0	150000.0	6	0	0	0	91074	0	1
4	0	0	1	2008	69.0	90000.0	7	1	0	0	60437	0	1

In [ ]:

```
Dependent.head()
```

Out[ ]:

```
0      480.0
1    16275.0
2     9800.0
3     1500.0
4     3600.0
Name: price, dtype: float64
```

7.2 FEATURE 2-User Interface

EVALUATION METRICS*

In [ ]:

```
#importing necessary libraries to find evaluation of the model
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import math
```

In [ ]:

```
#mean squared error
MSE=mean_squared_error(y_test,y_pred)
print("MSE:",MSE)
```

MSE: 3837929.3862338685

In [ ]:

```
#Root mean squared error
RMSE=math.sqrt(MSE)
print("RMSE:",RMSE)
```

RMSE: 1959.063395154396

In [ ]:

```
#checking the performance of the model using r2_score
r2=r2_score(y_test,y_pred)
print("R2_score:",r2)
```

R2_score: 0.840904862881962

In [ ]:

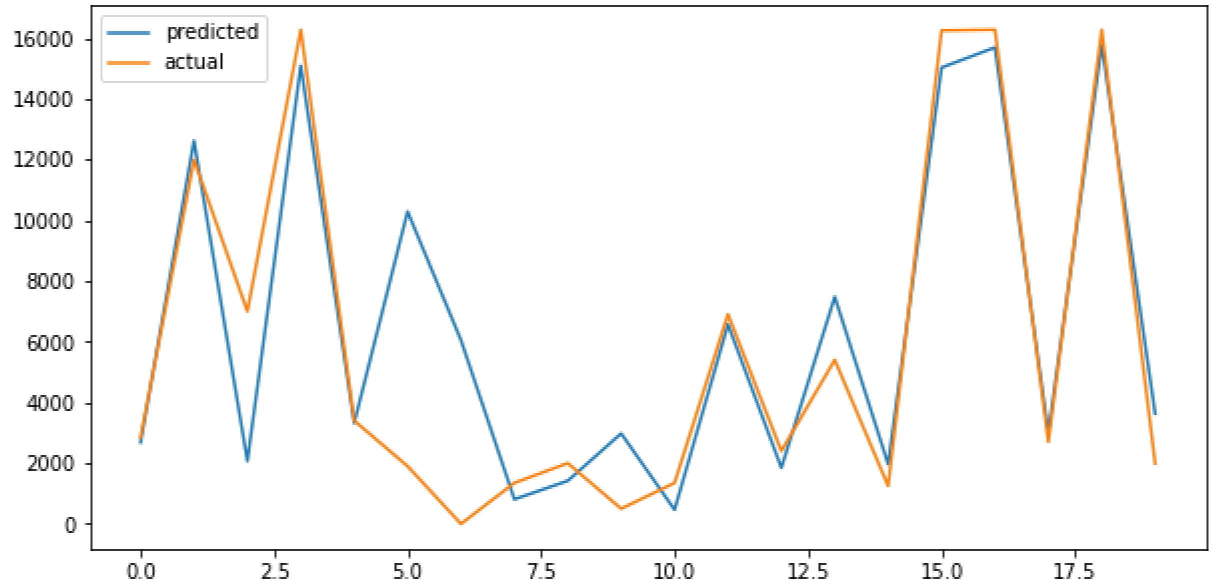
```
#Adjusted R square
Adjusted_R2=1-(1-r2*((x_test.shape[0]-1)/(x_test.shape[0]-x_test.shape[1]-1)))
print("Adjusted R2:",Adjusted_R2)
```

Adjusted R2: 0.841022575799409

In [ ]:

```
#plot for predicted and actual price
plt.figure(figsize=(10,5))
plt.plot(y_pred[0:20])
plt.plot(np.array(y_test[0:20]))
plt.legend(["predicted","actual"])
```

plt.show()



In [ ]:

print("The accuracy of the RandomForestRegression:",r2)

The accuracy of the RandomForestRegression: 0.840904862881962

7.3 DATABASE SCHEMA

```
.heade
r{

        min-height: 100vh;

        width: 100%;
```

```
 background-image:
linear-gradient(rgba(25,30,30,0.7),rgba(25,30,30,0.7)),url(../Images/car6.png);

 background-position: center;

 background-size: cover;

 position: relative;

}

.text-box{

 text-align: center;

 position: relative;
```

```
 color:  #FFE4C4;

 top:50%;

}

.text-box h1{

 margin-top: 50px;

 font-size: 55px;

}

.text-box p{
```

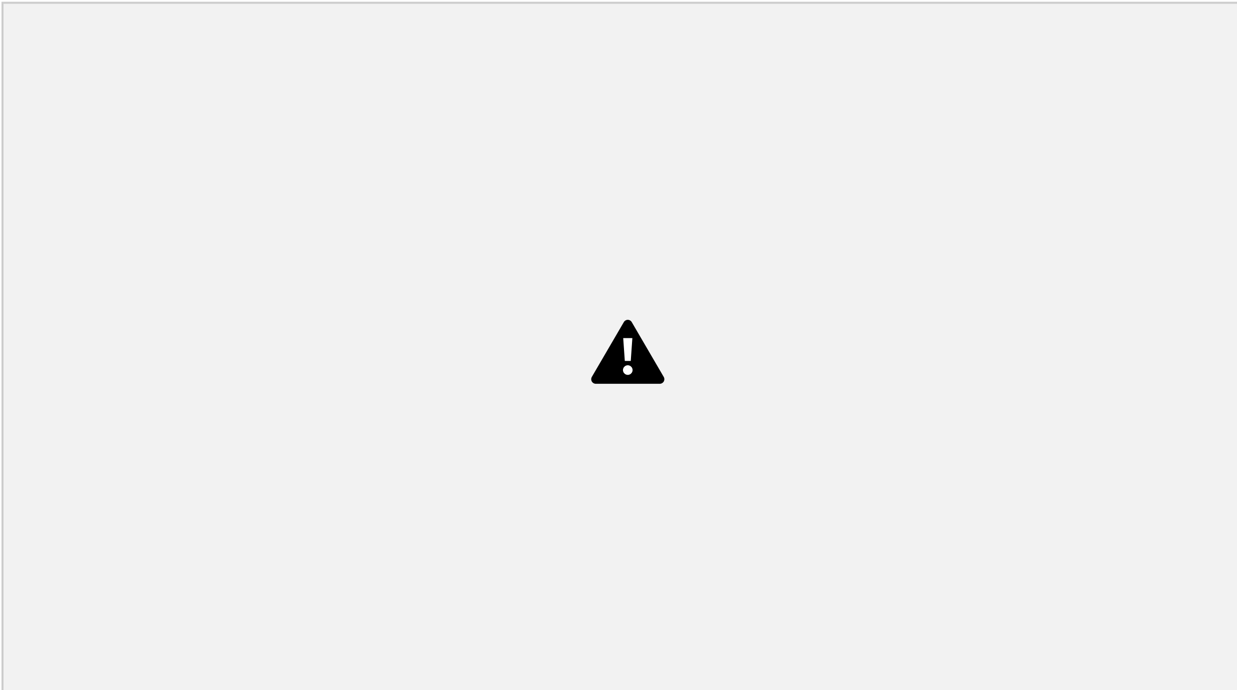
```
 margin: 10px 0 40px;

 font-size: 15px;

}

body{

       margin: 0;

}

nav{
```

```
display:flex;

padding: 2% 6%;

justify-content: space-between;

align-items: center;

}
```

Footer

8. TESTING

8.1 TEST CASES



8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing are done. The User Acceptance of this product is not surveyed enough to give a solid conclusion. The theoretical and hypothetical acceptance is calculated to be high enough to conclude that this product is usable and valuable.

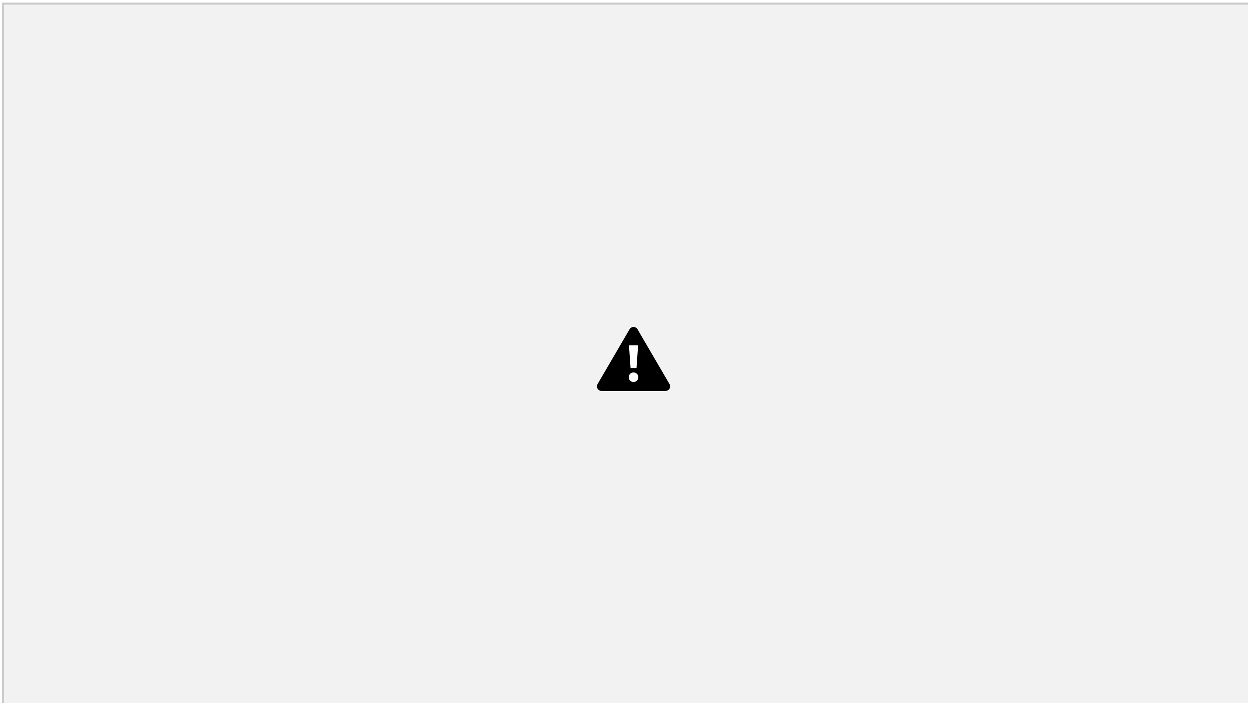
9. RESULTS

9.1 PERFORMANCE METRICS

The Performance is the Accuracy of the model trained.

The training accuracy of the model is 92%.

The testing accuracy of the model is 89%.



10.ADVANTAGES & DISADVANTAGES

Pros:

- Good at learning complex and non-linear relationships
- Highly explainable and easy to interpret

- Robust to outliers
- No feature scaling is required

Cons:

- Consumes more time
- Requires high computational power

11.CONCLUSION

We have successfully developed an application using python flask, HTML, CSS. By using the application, we can predict weather we can get admission in the desired University or not.

The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. The proposed system will help to determine the accurate price of used car price prediction. This paper compares 3 different algorithms for machine learning : Linear Regression, Lasso Regression and Ridge Regression.

12.FUTURE SCOPE

In future this machine learning model may bind with various website which can provide real time data for price prediction. Also we may add large historical data of car price which can help to improve accuracy of the machine learning model. We can build an android app as user interface for interacting with user. 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

13.APPENDIX

PROJECT DEMO LINK

(https://IBM-EPBL/IBM-Project-34064-1660231086) DEMO LINK