VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Project Report on

Car Rental System

In partial fulfillment of the Third Year, Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai Academic Year 2020-2021.

Submitted by

Abhishek Odrani D12C 48

Mohit Peshwani D12C 49

Anuraag Punjabi D12C 69

Project Mentor

Mr. Richard Joseph

(2020-2021)

VIVEKANAND EDUCATION SOCIETY’S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Certificate

This is to certify that ***Mohit Peshwani, Abhishek Odrani and Anuraag Punjabi*** of Third Year Computer Engineering studying under the University of Mumbai have satisfactorily completed the mini project on “***Car Rental System***” as a part of their coursework of Mini Project for Semester-VI under the guidance of their mentor ***Mr. Richard Joseph*** in the year 2020-2021.

This mini project report entitled (***Car Rental System***) by (***Mohit Peshwani, Abhishek Odrani and Anuraag Punjabi***) is approved for the degree of ____________ **(Data Warehouse & Mining)**.

Programme Outcomes	Grade
PO1,PO2,PO3,PO4,PO5,PO6,PO7, PO8, PO9, PO10, PO11, PO12 PSO1, PSO2	

Date:

Project Guide: Mr. Richard Joseph

Mini Project Report Approval for T.E (Computer Engineering)

This mini project report entitled ***Car Rental System*** by ***Mohit Peshwani, Abhishek Odrani and Anuraag Punjabi*** is approved for the degree of T.E Computer Engg.

Internal Examiner

External Examiner

Head of the Department

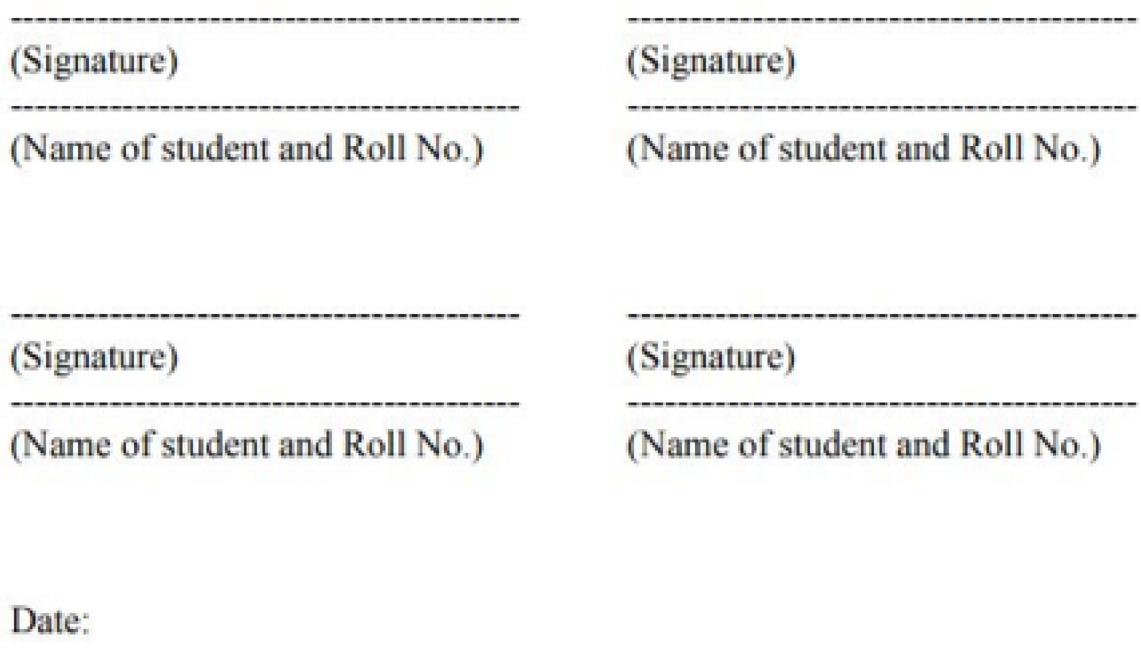
Principal

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.



--	--
(Signature)	(Signature)
--	--
(Name of student and Roll No.)	(Name of student and Roll No.)
--	--
(Signature)	(Signature)
--	--
(Name of student and Roll No.)	(Name of student and Roll No.)

Date:

ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society's Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Mr. Richard Joseph** (Project Guide) for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

Index

List of Diagrams

Abstract

Our Aim is to design and create a data management System for a car rental company. This enables the admin to rent a vehicle that can be used by a customer. By paying the money during a specified period of time. This system increases customer retention and simplifies vehicle and staff Management in an efficient way. The car rental system has a very user friendly interface. Thus the users will feel very easy to work on it. By using this system admin can manage their rental, payment, availability of cars and vehicle details.

The car information can be added to the system or existing car information can be edited or deleted too by the Administrator. The transaction reports of the car rental system can be retrieved by the admin, when it's required. Customers should create a new account before logging in or he/she can log into the System with his/her created account. Then they can view the available cars of a brand as per their location and make a reservation for a car.

Introduction

CAR RENTAL SYSTEM (CRS) is a web based system for a company that rents out cars. This system enables the company to make their services available to the public through the internet and also keep records about their services. Car renting is essential to many people who plan to travel or move from one place to another for business purposes, tour, and holidays, for these reasons car renting can be very helpful. In the car rental System, an admin can add cars in each center /branch also having higher priority than others.

The admin can maintain details of car sales along with different cars available at specific locations and also has the record of customers who rent cars. Also the customer's details with proof of identity and driving license is mandatory. It saves time and labor. This tool shall ask the user for information such as the date and time of journey, type of car etc. Using these details, the tool shall help the customer to book a car for the journey.

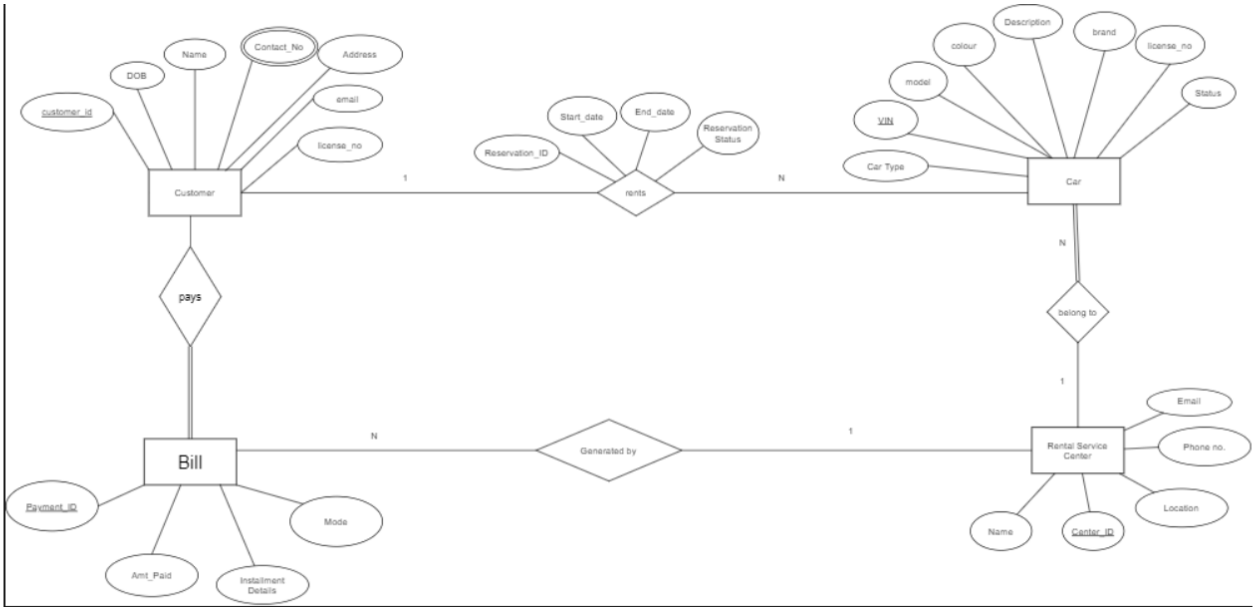
Problem Statement

A car rental system helps customers to use a car that can be used temporarily for a fee during a specified period. Getting a rental car helps people get around despite the fact they do not have access to their own personal vehicle or don't own a vehicle at all. The individual who needs a car must contact a rental car company and contract out for a vehicle. This system increases customer retention and simplifies the booking process.

Proposed Solution

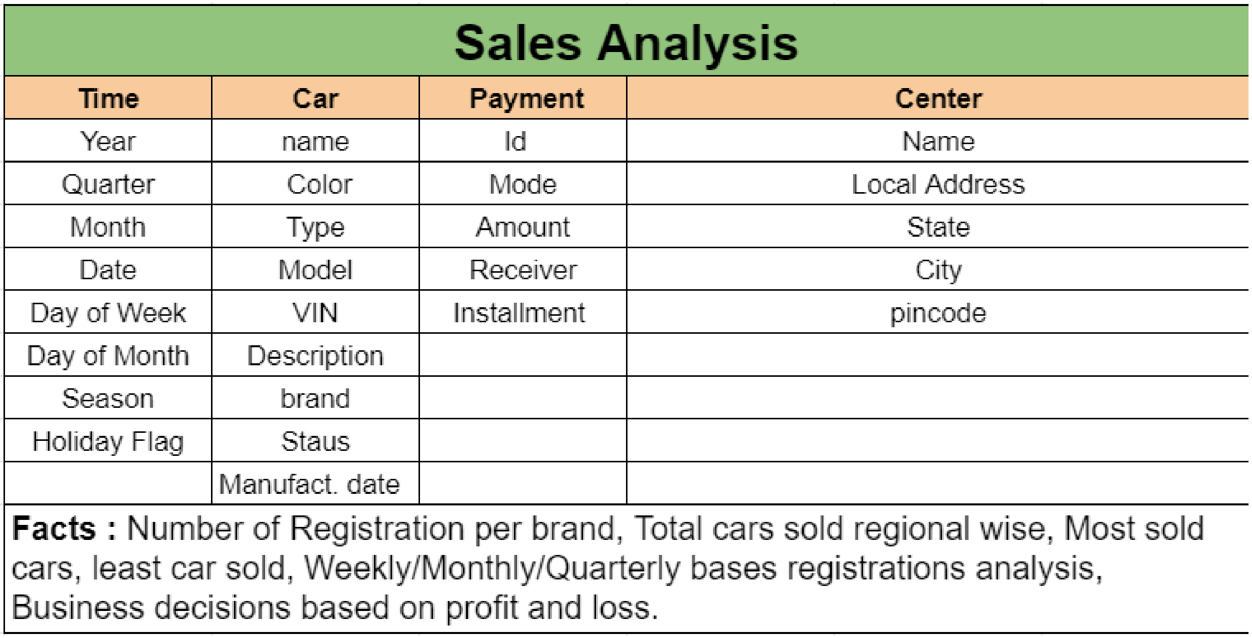
I. ER Diagram

In order to get the clear idea and working of our data warehouse for the Car Rental System we have Information package and ER diagram to get an overview about the design of our data warehouse. To keep the track of sales, schedules, availability of cars over different centers.



ER Diagram

II. Information Packages :

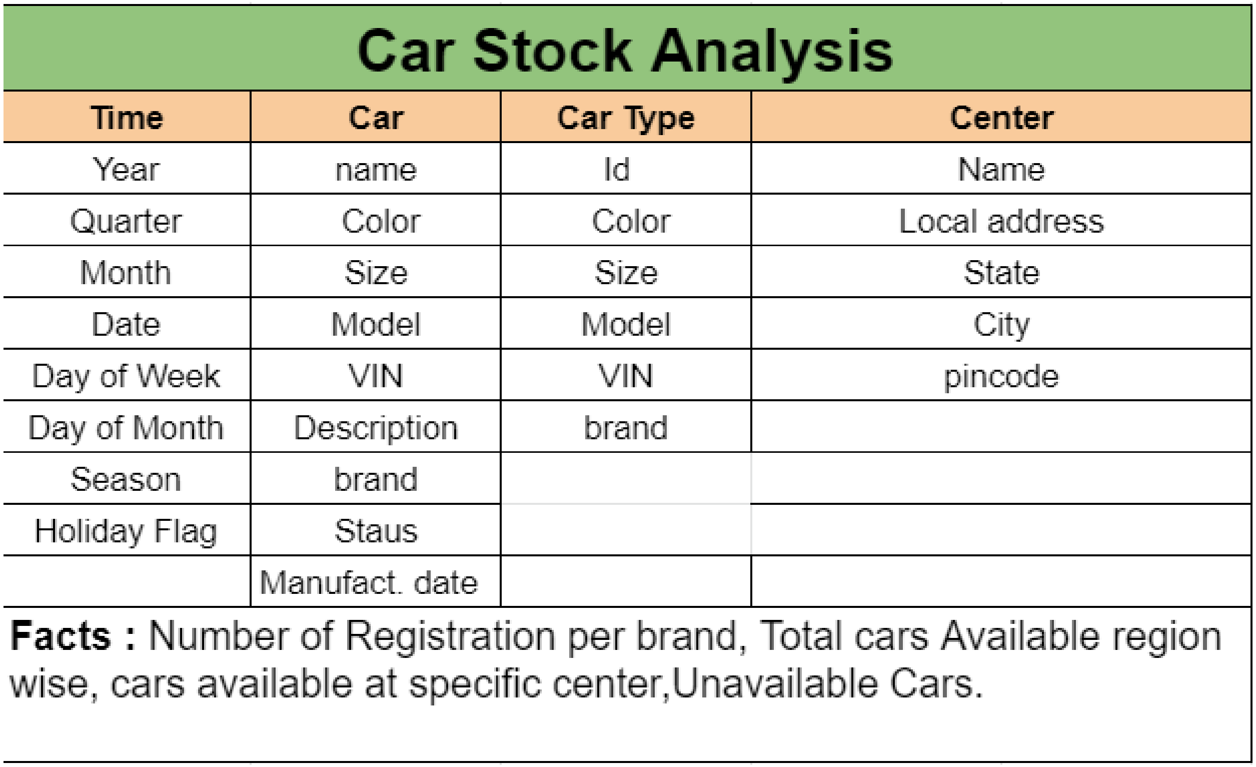


Sales Analysis

Time	Car	Payment	Center
Year	name	Id	Name
Quarter	Color	Mode	Local Address
Month	Type	Amount	State
Date	Model	Receiver	City
Day of Week	VIN	Installment	pincode
Day of Month	Description		
Season	brand		
Holiday Flag	Staus		
	Manufact. date		

Facts : Number of Registration per brand, Total cars sold regional wise, Most sold cars, least car sold, Weekly/Monthly/Quarterly bases registrations analysis, Business decisions based on profit and loss.

Sales Information Package

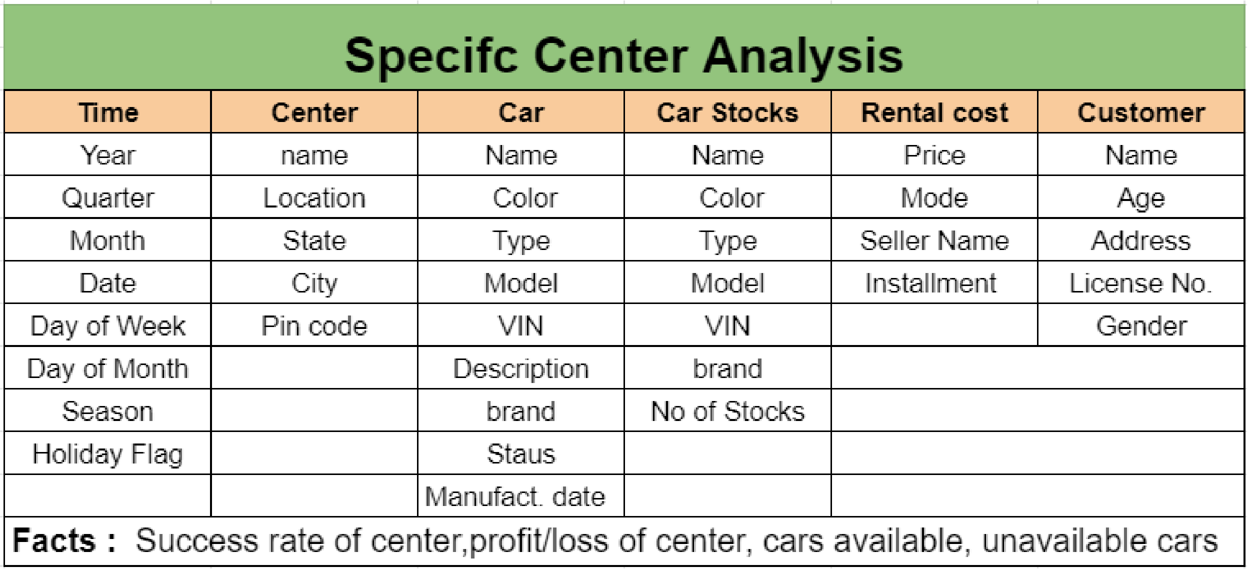


Car Stock Analysis

Time	Car	Car Type	Center
Year	name	Id	Name
Quarter	Color	Color	Local address
Month	Size	Size	State
Date	Model	Model	City
Day of Week	VIN	VIN	pincode
Day of Month	Description	brand	
Season	brand		
Holiday Flag	Staus		
	Manufact. date		

Facts : Number of Registration per brand, Total cars Available region wise, cars available at specific center,Unavailable Cars.

Sales Information Package

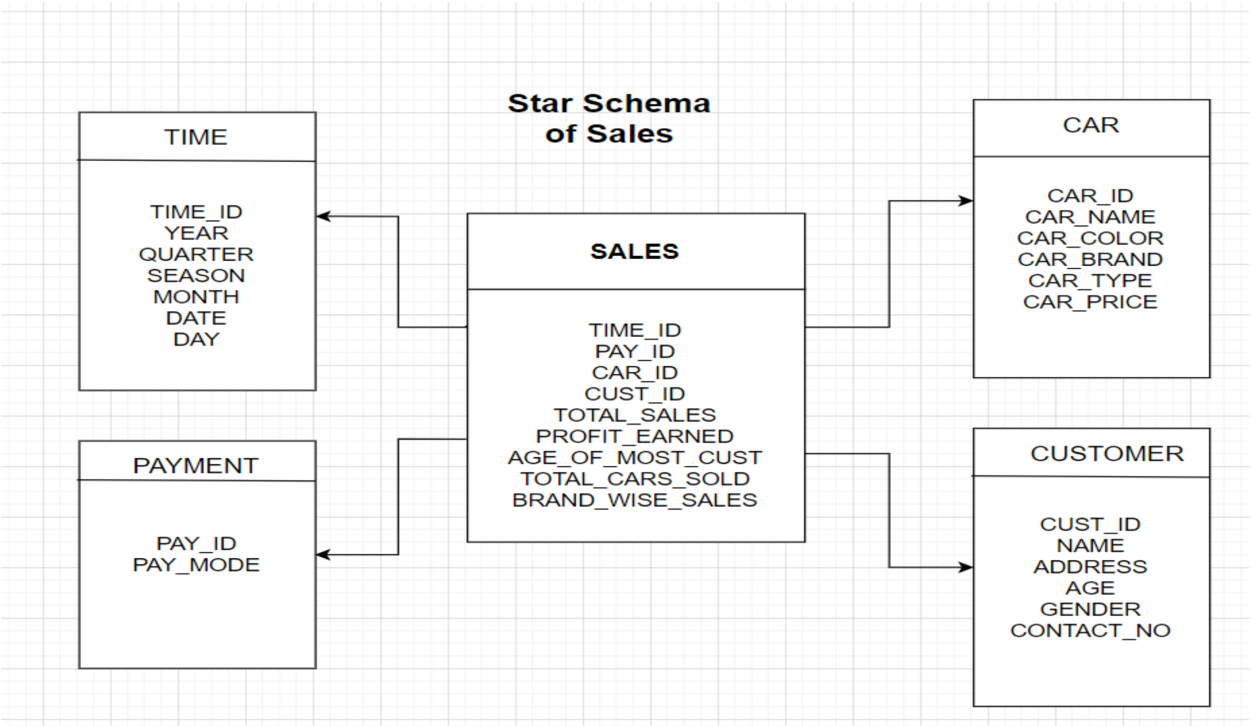


Specifc Center Analysis					
Time	**Center**	**Car**	**Car Stocks**	**Rental cost**	**Customer**
Year	name	Name	Name	Price	Name
Quarter	Location	Color	Color	Mode	Age
Month	State	Type	Type	Seller Name	Address
Date	City	Model	Model	Installment	License No.
Day of Week	Pin code	VIN	VIN		Gender
Day of Month		Description	brand		
Season		brand	No of Stocks		
Holiday Flag		Staus			
		Manufact. date			
Facts : Success rate of center,profit/loss of center, cars available, unavailable cars					

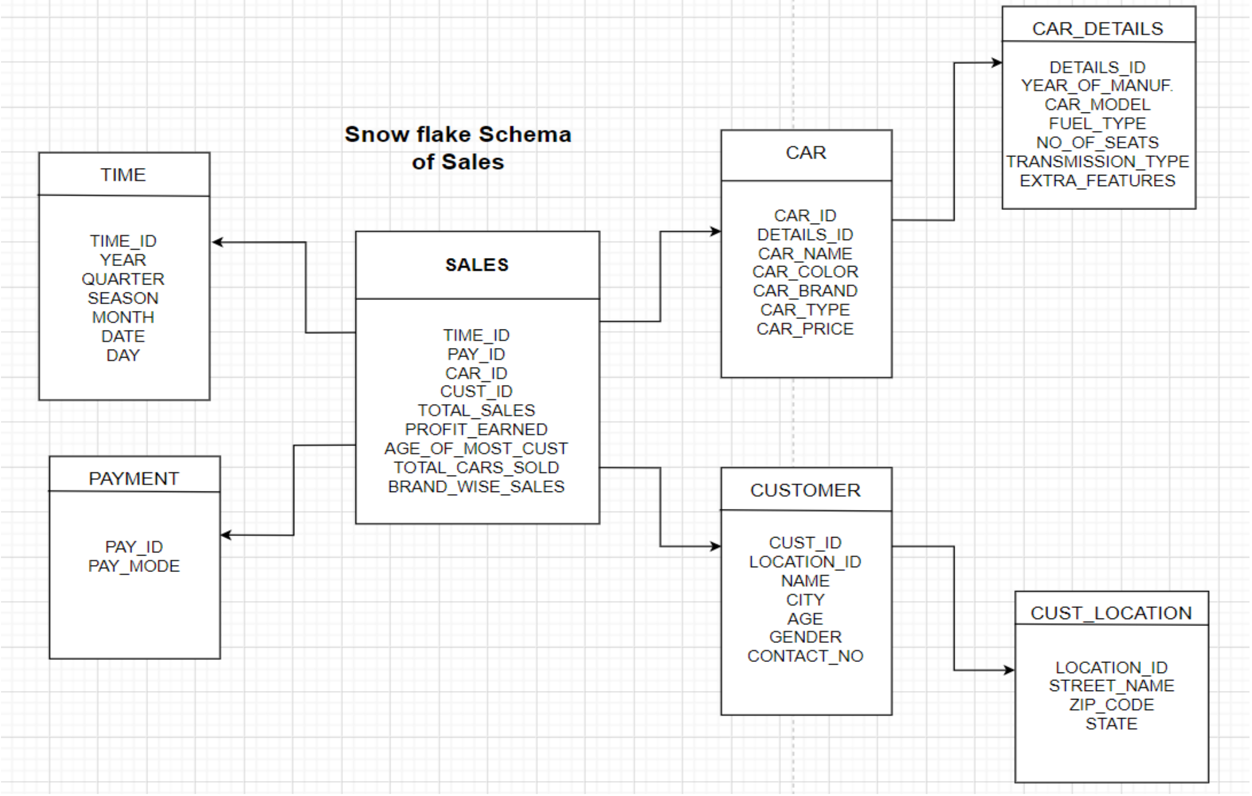
Center Information Package

III. Star schema and snowflake schemas :

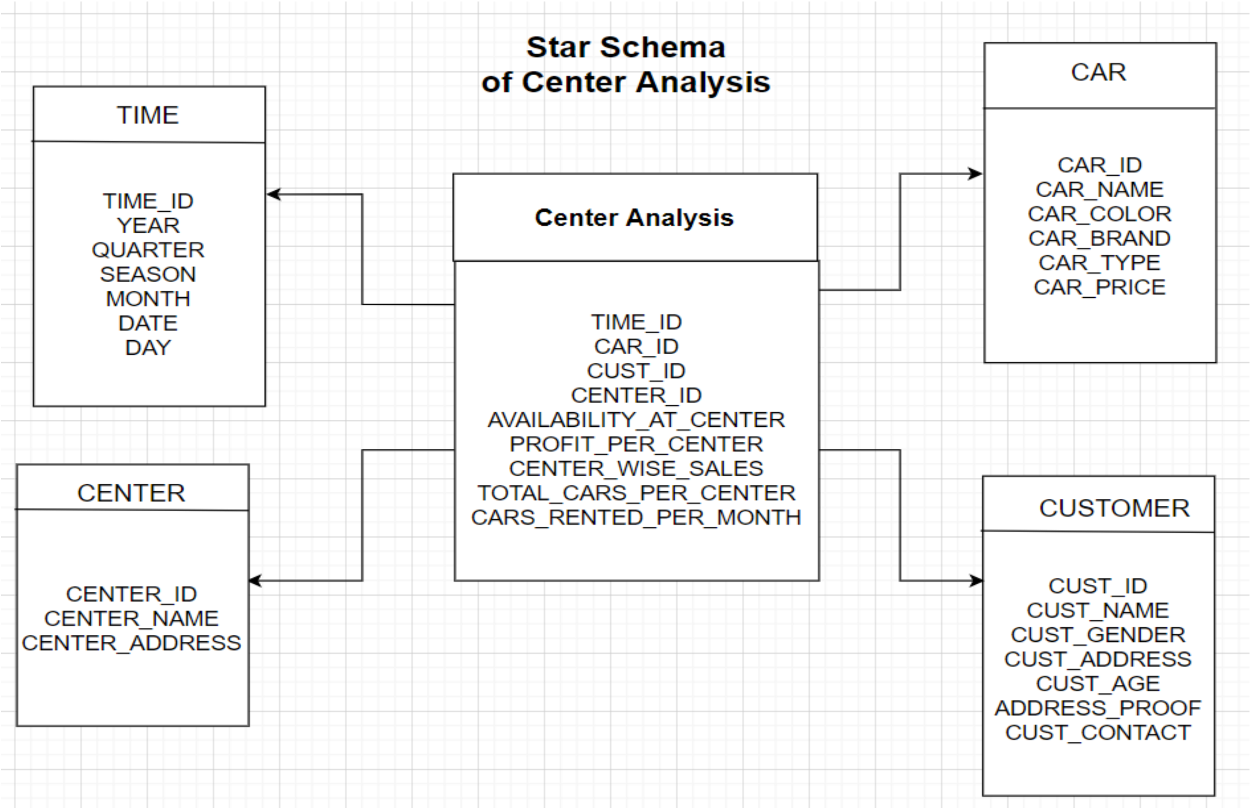
For designing the data warehouse and understanding the relationships between various dimensions star and snowflake schemas were designed. It was useful in deriving the fact table for our warehouse.



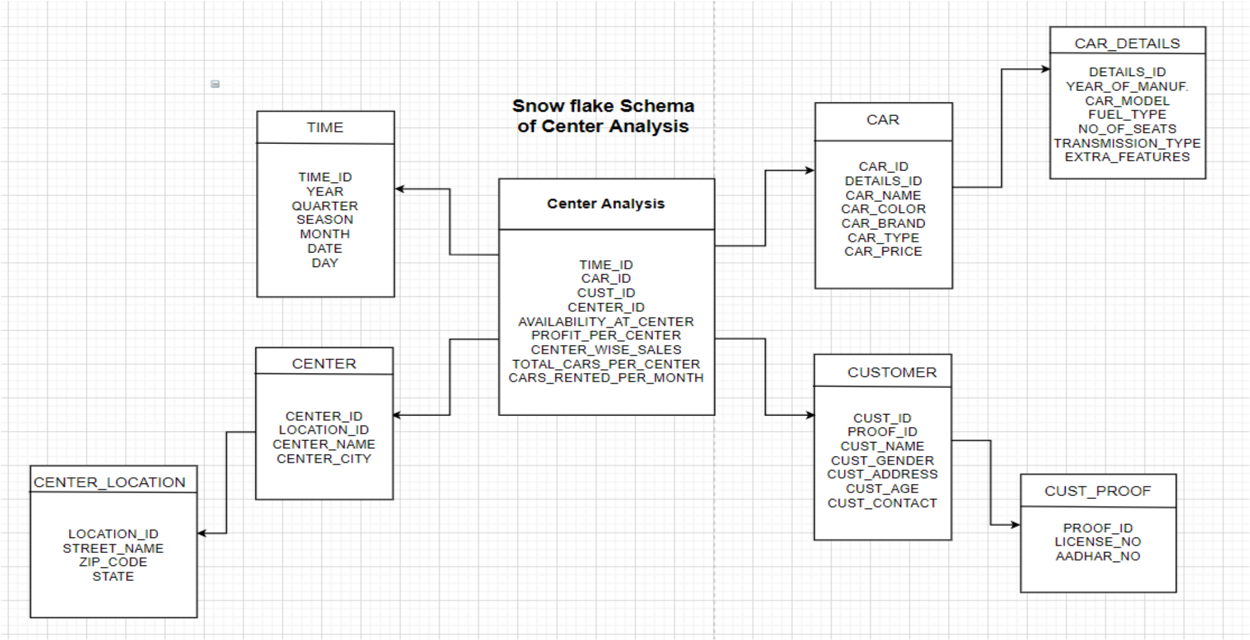
Star Schema of Sales



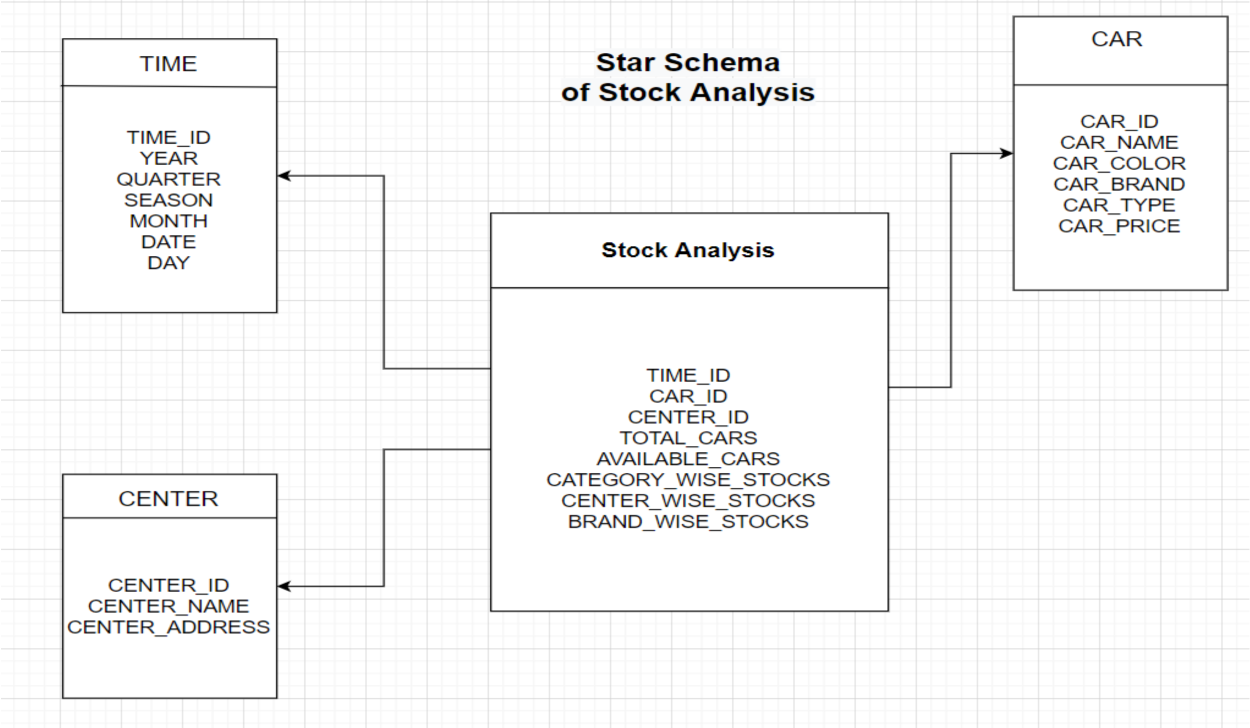
Snowflake Schema of Sales



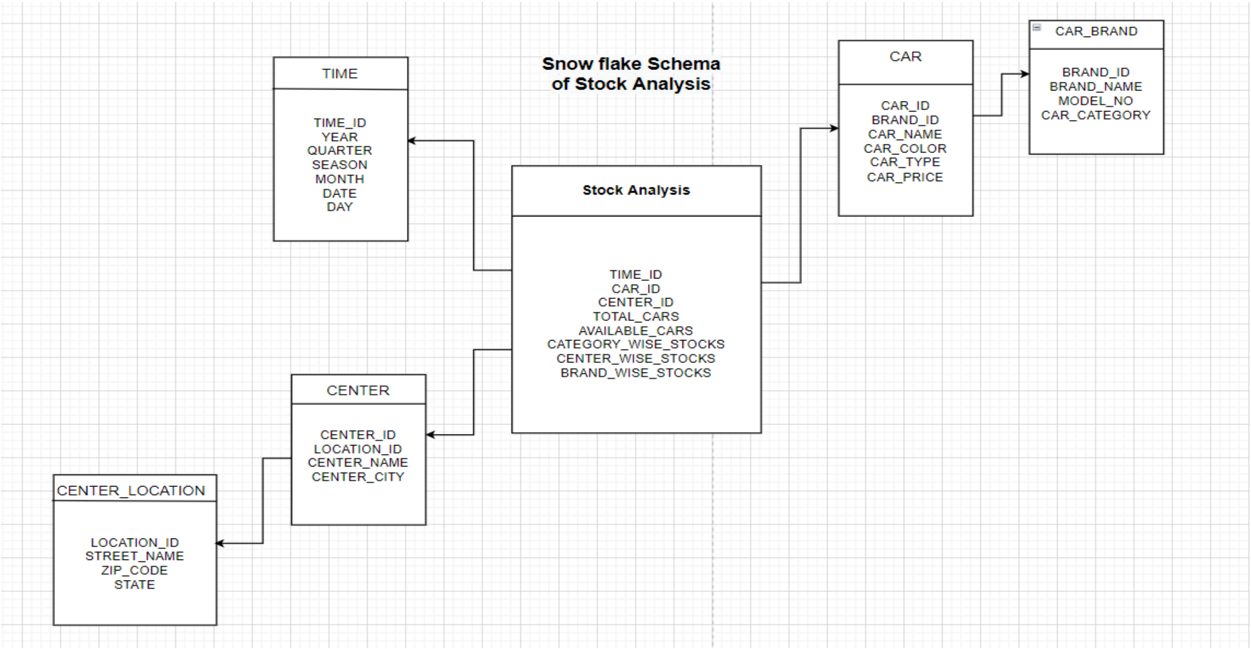
Star Schema of Center



Snowflake Schema of Center



Star Schema of Stocks



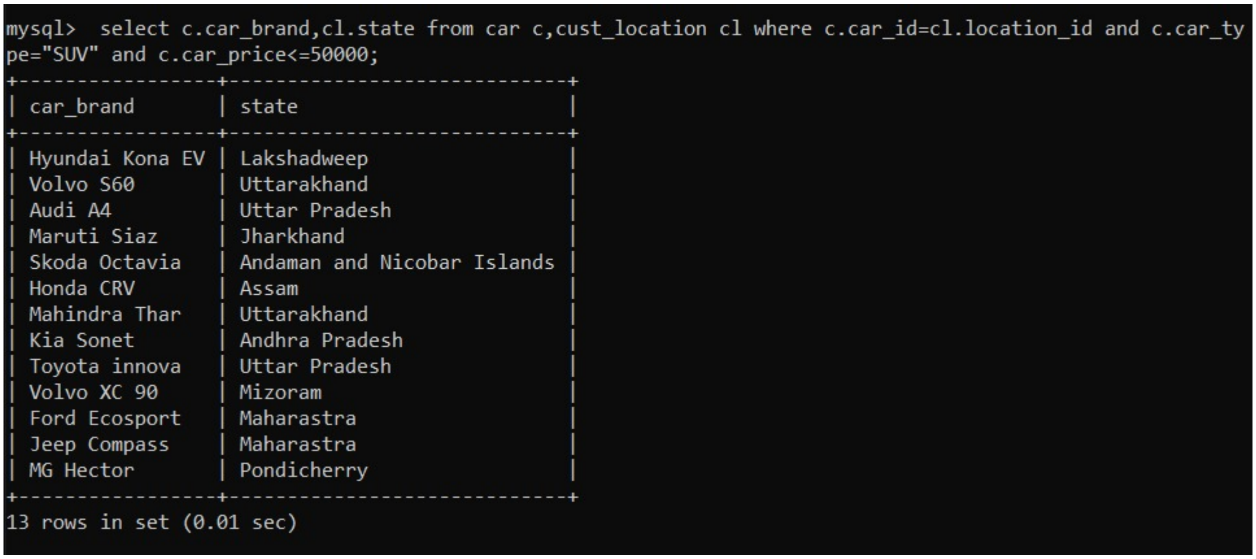
Snowflake Schema of Stocks

IV. OLAP Queries :

Olap queries helped us in processing, analysing our project. Using Olap queries it was easy to understand the annual sales over a particular period of time, location or daily schedules of a particular center. It was also easy for the admin side to understand the car availability at the centers, brand wise sales, category of the cars.

Suv sales in states under 50k

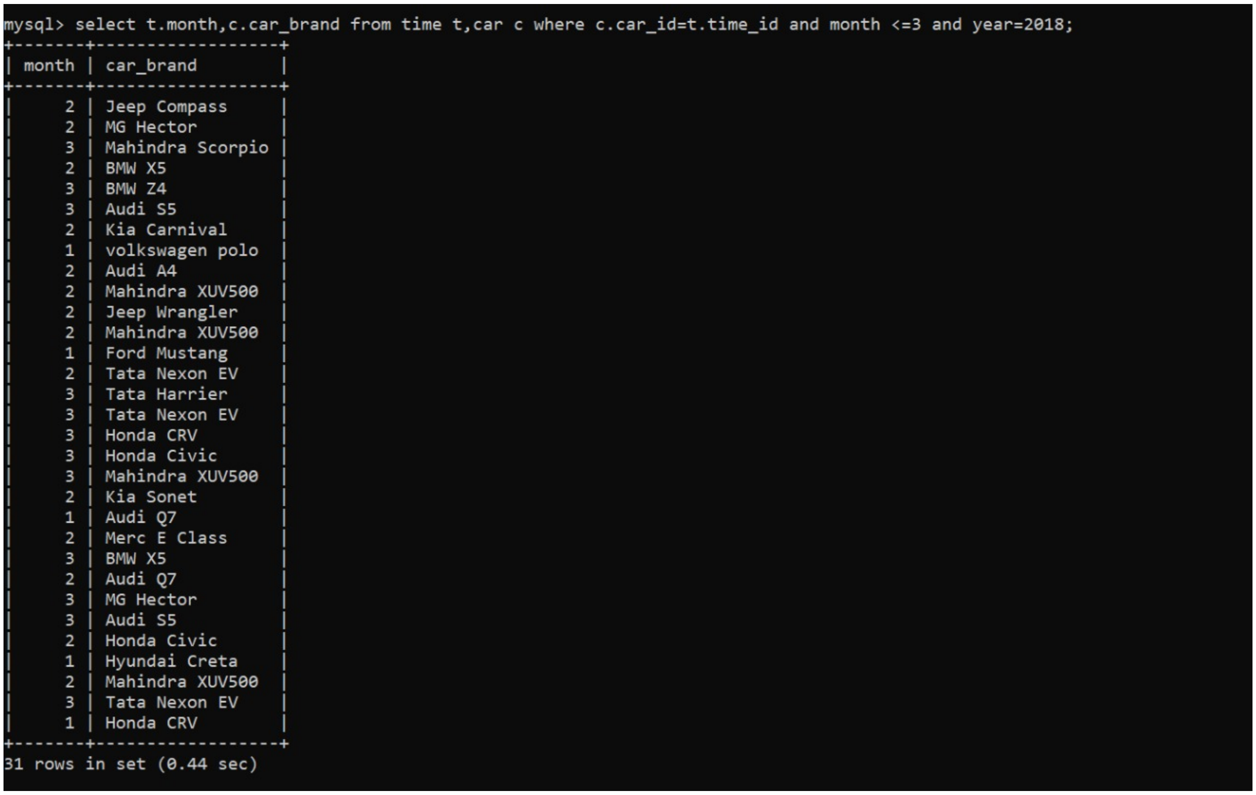
select c.car_brand,cl.state from car c,cust_location cl where c.car_id=cl.location_id and c.car_type="SUV" and c.car_price<=50000;



```
mysql>  select c.car_brand,cl.state from car c,cust_location cl where c.car_id=cl.location_id and c.car_ty
pe="SUV" and c.car_price<=50000;
+-----------------+-----------------------------+
| car_brand       | state                       |
+-----------------+-----------------------------+
| Hyundai Kona EV | Lakshadweep                 |
| Volvo S60       | Uttarakhand                 |
| Audi A4         | Uttar Pradesh               |
| Maruti Siaz     | Jharkhand                   |
| Skoda Octavia   | Andaman and Nicobar Islands |
| Honda CRV       | Assam                       |
| Mahindra Thar   | Uttarakhand                 |
| Kia Sonet       | Andhra Pradesh              |
| Toyota innova   | Uttar Pradesh               |
| Volvo XC 90     | Mizoram                     |
| Ford Ecosport   | Maharastra                  |
| Jeep Compass    | Maharastra                  |
| MG Hector       | Pondicherry                 |
+-----------------+-----------------------------+
13 rows in set (0.01 sec)
```

Sales of Q1 - 2018

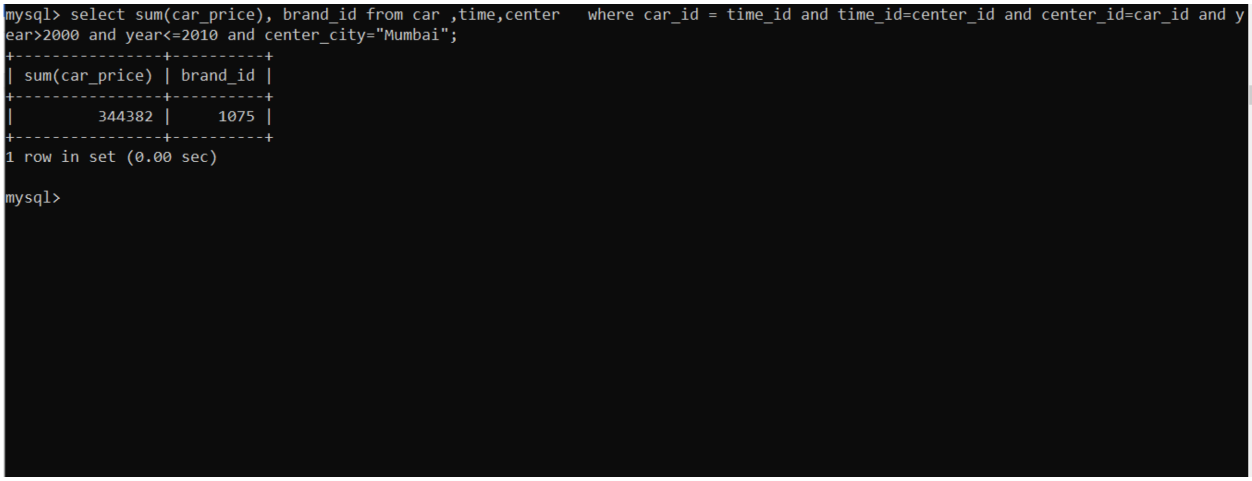
select t.month,c.car_brand from time t,car c where c.car_id=t.time_id and month <=3 and year=2018;



```
mysql> select t.month,c.car_brand from time t,car c where c.car_id=t.time_id and month <=3 and year=2018;
+-------+------------------+
| month | car_brand        |
+-------+------------------+
|     2 | Jeep Compass     |
|     2 | MG Hector        |
|     3 | Mahindra Scorpio |
|     2 | BMW X5           |
|     3 | BMW Z4           |
|     3 | Audi S5          |
|     2 | Kia Carnival     |
|     1 | volkswagen polo  |
|     2 | Audi A4          |
|     2 | Mahindra XUV500  |
|     2 | Jeep Wrangler    |
|     2 | Mahindra XUV500  |
|     1 | Ford Mustang     |
|     2 | Tata Nexon EV    |
|     3 | Tata Harrier     |
|     3 | Tata Nexon EV    |
|     3 | Honda CRV        |
|     3 | Honda Civic      |
|     3 | Mahindra XUV500  |
|     2 | Kia Sonet        |
|     1 | Audi Q7          |
|     2 | Merc E Class     |
|     3 | BMW X5           |
|     2 | Audi Q7          |
|     3 | MG Hector        |
|     3 | Audi S5          |
|     2 | Honda Civic      |
|     1 | Hyundai Creta    |
|     2 | Mahindra XUV500  |
|     3 | Tata Nexon EV    |
|     1 | Honda CRV        |
+-------+------------------+
31 rows in set (0.44 sec)
```

Stocks of Mumbai city ranging between 2000 & 2010

select sum(car_price) from car ,time,center   where car_id = time_id and time_id=center_id and center_id=car_id and year>2000 and year<=2010 and center_city="Mumbai";



```
mysql> select sum(car_price), brand_id from car ,time,center   where car_id = time_id and time_id=center_id and center_id=car_id and y
ear>2000 and year<=2010 and center_city="Mumbai";
+----------------+----------+
| sum(car_price) | brand_id |
+----------------+----------+
|         344382 |     1075 |
+----------------+----------+
1 row in set (0.00 sec)

mysql>
```

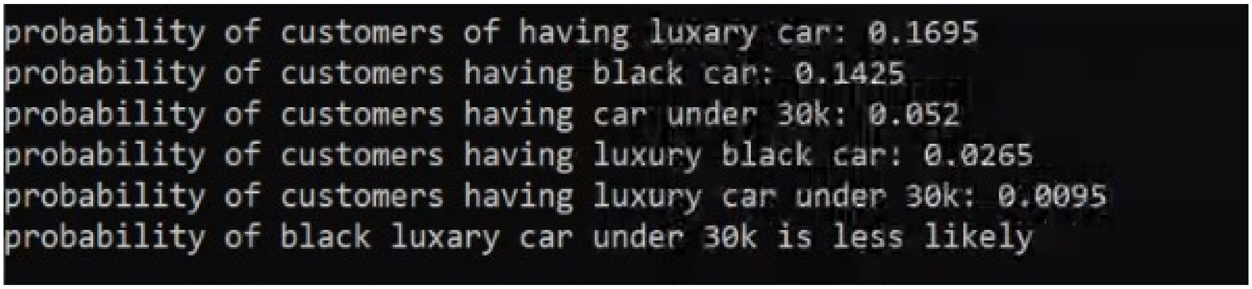
V. Classification algorithm Naive Bayes

Code :

```
import java.io.*;
import java.util.Scanner;
public class Naive2{
    public static void main(String args[]) throws Exception
    {
    float luxary,black,under30,black_luxary,under30_luxary;
    luxary=black=under30=black_luxary=under30_luxary=0;
    float p_luxary,p_black,p_under30,p_black_luxary,p_under30_luxary;
    p_luxary=p_black=p_under30=p_black_luxary=p_under30_luxary=0;
    float n= 0;
    Scanner sc=new Scanner(new File("C:\\Users\\Mohit
peshwani\\Desktop\\cars.csv"));
    String s1,s2,s3,s4;
    float c1,c2;
    String line1=sc.next();

    while (sc.hasNext()) {
        String line=sc.next();
        String[] str = line.split(",");
        s1=str[4];
        s3="luxury";
        if(s1.equals(s3))
        {
            luxury++;
        }
        s2=str[3];
        s4="black";
        if(s2.equals(s4))
        {
            black++;
        }
        if(s1.equals(s3) && s2.equals(s4)){
            black_luxary++;
        }
        c1 = Float.parseFloat(str[5]);
        System.out.println(c1+1);
        if(c1<=30000){
            under30++;
        }
        if(s1.equals(s3) && c1<=30000){
            under30_luxary++;
        }
        n++;
    }p_luxary = luxury/n;
    p_black = black/n;
```

```
        p_under30 = under30/n;
        System.out.println("probability of customers of having luxury car: " + p_luxary);
        System.out.println("probability of customers having black car: " +p_black);
        System.out.println("probability of customers having car under 30k: "
+p_under30);
        p_black_luxary = black_luxary/n;
        p_under30_luxary = under30_luxary/n;
        System.out.println("probability of customers having luxury black car: "
+p_black_luxary);
        System.out.println("probability of customers having luxury car under 30k: "
+p_under30_luxary);
        float
p_luxary_black_under30=(p_black_luxary*p_under30_luxary*p_luxary)/(p_black*p_und
er30);
        System.out.println("probability of black luxury car under 30k is less likely");
        sc.close();
        }
}
```

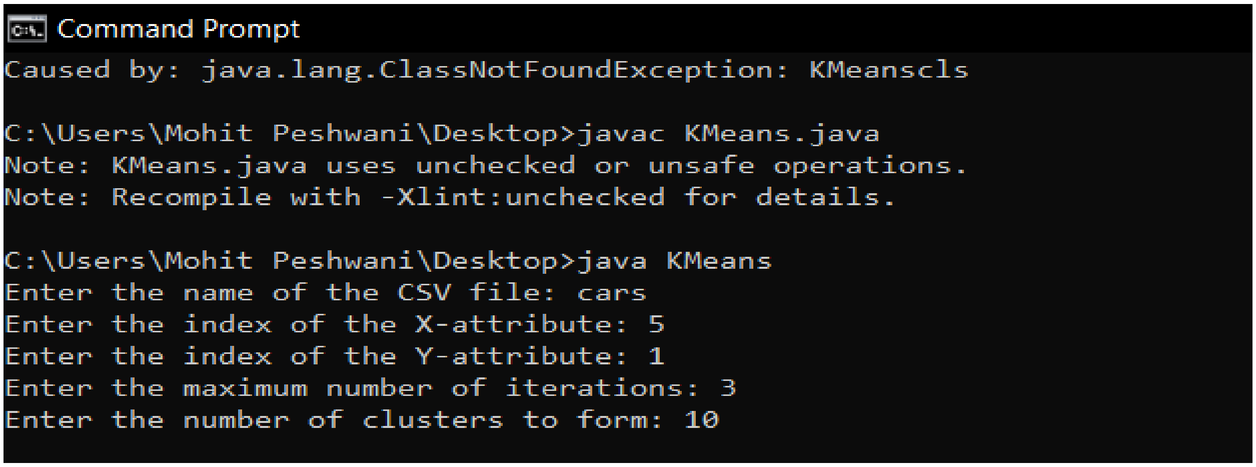
Output :



```
probability of customers of having luxary car: 0.1695
probability of customers having black car: 0.1425
probability of customers having car under 30k: 0.052
probability of customers having luxury black car: 0.0265
probability of customers having luxury car under 30k: 0.0095
probability of black luxary car under 30k is less likely
```

VI. Clustering algorithm (K-means)

Output :



```
Caused by: java.lang.ClassNotFoundException: KMeanscls

C:\Users\Mohit Peshwani\Desktop>javac KMeans.java
Note: KMeans.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\Mohit Peshwani\Desktop>java KMeans
Enter the name of the CSV file: cars
Enter the index of the X-attribute: 5
Enter the index of the Y-attribute: 1
Enter the maximum number of iterations: 3
Enter the number of clusters to form: 10
```

The final clusters are:

```
[(5455.0, 1796.0), (6528.0, 106.0), (6616.0, 406.0), (6804.0, 943.0), (7025.0, 1276.0), (7819.0,
632.0), (8308.0, 1791.0), (8540.0, 1373.0), (8624.0, 904.0), (9072.0, 187.0), (9393.0, 111.0),
(10243.0, 1537.0), (10650.0, 585.0), (10831.0, 835.0), (11045.0, 607.0), (11899.0, 499.0),
(12544.0, 205.0), (12729.0, 1004.0), (12880.0, 190.0), (13542.0, 171.0), (14128.0, 932.0),
(14362.0, 531.0), (14502.0, 32.0), (14526.0, 5.0), (302857.0, 942.0), (302990.0, 1041.0),
(303024.0, 253.0), (303194.0, 915.0), (303795.0, 319.0), (304124.0, 543.0), (304178.0, 1672.0),
(304385.0, 1180.0), (304449.0, 1596.0), (304530.0, 611.0), (304581.0, 881.0), (304758.0,
687.0), (305224.0, 1079.0)]
```

```
[(355464.0, 447.0), (355590.0, 258.0), (356144.0, 704.0), (356274.0, 1931.0), (356483.0,
1553.0), (356880.0, 524.0), (356983.0, 1714.0), (356984.0, 1508.0), (357296.0, 1358.0),
(357297.0, 1965.0), (357371.0, 626.0), (357502.0, 308.0), (357688.0, 168.0), (357696.0,
1662.0), (357838.0, 1488.0), (357860.0, 155.0), (358311.0, 850.0), (358349.0, 870.0),
(358602.0, 19.0), (358980.0, 1660.0), (399944.0, 1797.0), (399971.0, 1305.0), (400395.0,
885.0), (400662.0, 1121.0), (400920.0, 1441.0), (401078.0, 1461.0), (401408.0, 1087.0),
(401728.0, 441.0), (402268.0, 1239.0), (402522.0, 1101.0), (403044.0, 1412.0), (404131.0,
829.0), (404252.0, 1210.0), (404358.0, 77.0), (404444.0, 1375.0)]
```

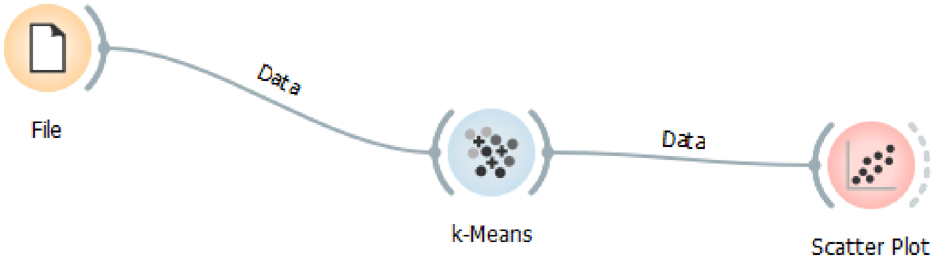
```
[(404682.0, 549.0), (404761.0, 1868.0), (405072.0, 650.0), (405326.0, 955.0), (405362.0, 31.0),
(405493.0, 906.0), (405518.0, 132.0), (406028.0, 329.0), (406174.0, 364.0), (406283.0, 1290.0),
(406460.0, 60.0), (406616.0, 394.0), (406630.0, 1325.0), (406710.0, 1315.0), (406921.0,
1515.0), (407560.0, 1381.0), (446064.0, 539.0), (446450.0, 1028.0), (446517.0, 1080.0),
(446559.0, 1591.0), (446726.0, 459.0), (446836.0, 745.0), (446975.0, 1789.0), (447096.0,
105.0), (447120.0, 1606.0), (447162.0, 1853.0), (448817.0, 1968.0), (449204.0, 140.0),
(449261.0, 555.0), (449268.0, 1971.0), (449373.0, 862.0), (449839.0, 928.0), (450048.0, 923.0),
(450549.0, 1352.0), (450985.0, 1400.0), (451083.0, 1904.0), (451140.0, 1153.0), (451278.0,
321.0), (451896.0, 1793.0), (451915.0, 158.0)]
```

```
[(452642.0, 1979.0), (452838.0, 90.0), (452993.0, 1270.0), (453091.0, 156.0), (453225.0, 504.0),
(453714.0, 1865.0), (453829.0, 635.0), (454246.0, 1342.0), (454674.0, 1502.0), (454898.0,
1905.0), (454999.0, 1142.0), (455244.0, 1726.0), (455499.0, 1492.0), (455703.0, 1837.0),
(455826.0, 1823.0), (456002.0, 1560.0), (456041.0, 88.0), (14526.0, 5.0), (302857.0, 942.0),
(302990.0, 1041.0), (303024.0, 253.0), (303194.0, 915.0), (459139.0, 1048.0), (459325.0,
1117.0), (496939.0, 1514.0), (496999.0, 1594.0), (497179.0, 1031.0), (406710.0, 1315.0),
(406921.0, 1515.0), (407560.0, 1381.0), (446064.0, 539.0), (497196.0, 1582.0), (498312.0,
967.0), (498320.0, 1443.0), (498326.0, 574.0), (498479.0, 551.0), (499211.0, 1344.0),
(499524.0, 340.0), (499901.0, 465.0), (499910.0, 1377.0)]
```

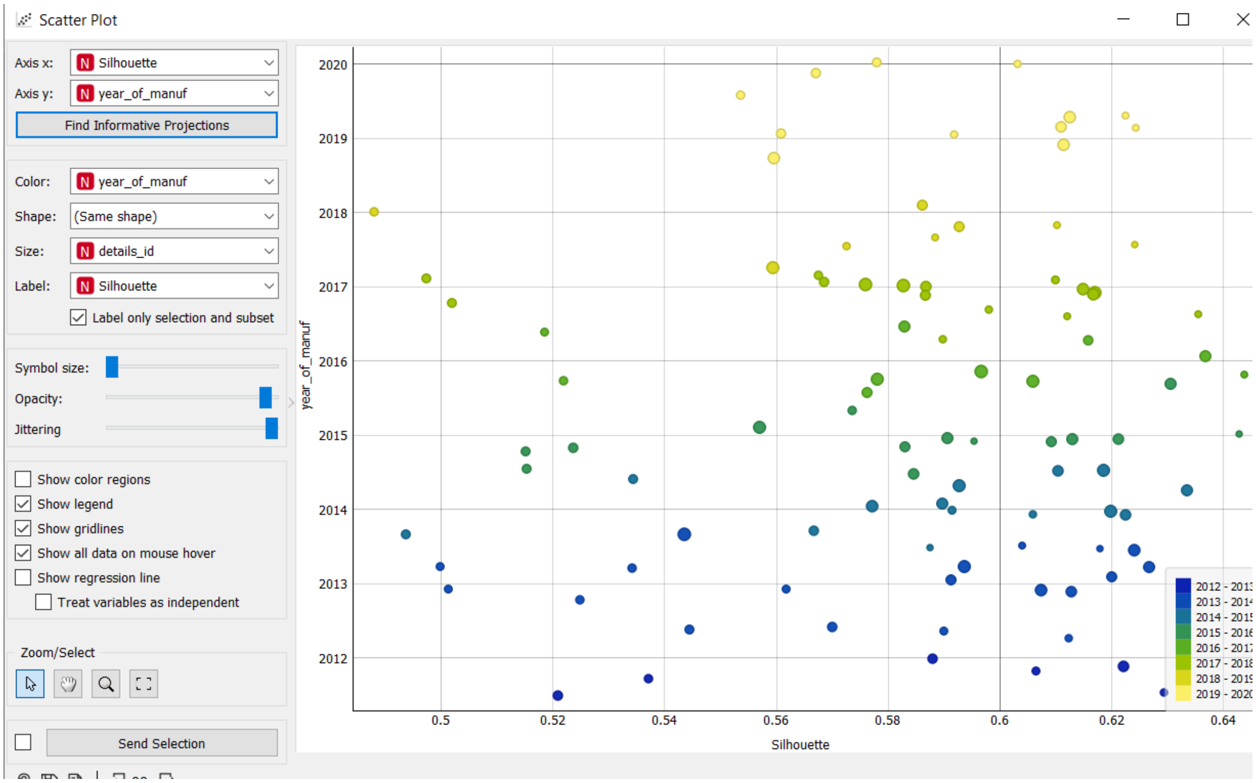
Iterations taken = 4

VII. Using Orange tool :

1) K Means



Apply Scatter Plot :


Scatter Plot

Axis x: Silhouette

Axis y: year_of_manuf

Find Informative Projections

Color: year_of_manuf

Shape: (Same shape)

Size: details_id

Label: Silhouette

☑ Label only selection and subset

Symbol size:

Opacity:

Jittering

☐ Show color regions

☑ Show legend

☑ Show gridlines

☑ Show all data on mouse hover

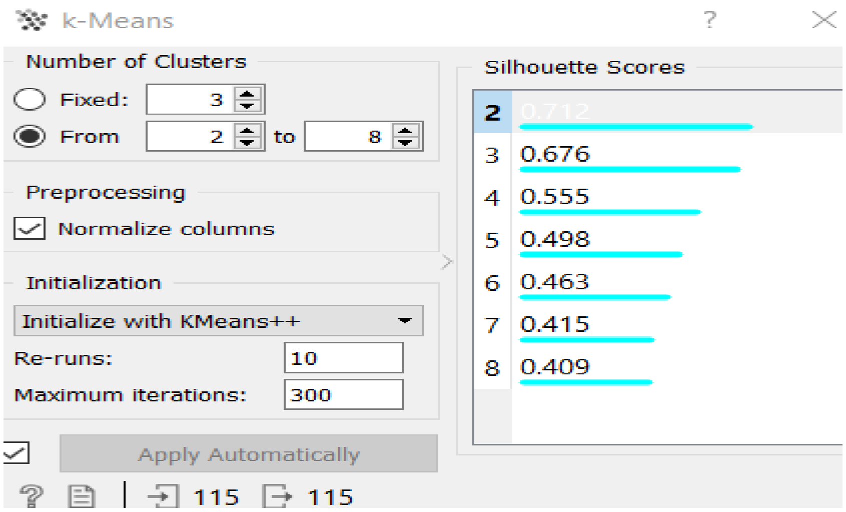
☐ Show regression line

☐ Treat variables as independent

Zoom/Select

☐ Send Selection

Applying K-Means :



k-Means

Number of Clusters

○ Fixed: 3

● From 2 to 8

Preprocessing

☑ Normalize columns

Initialization

Initialize with KMeans++

Re-runs: 10

Maximum iterations: 300

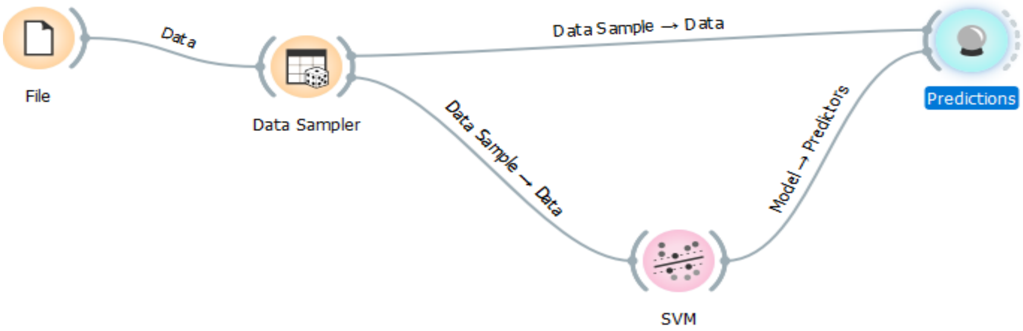
☑ Apply Automatically

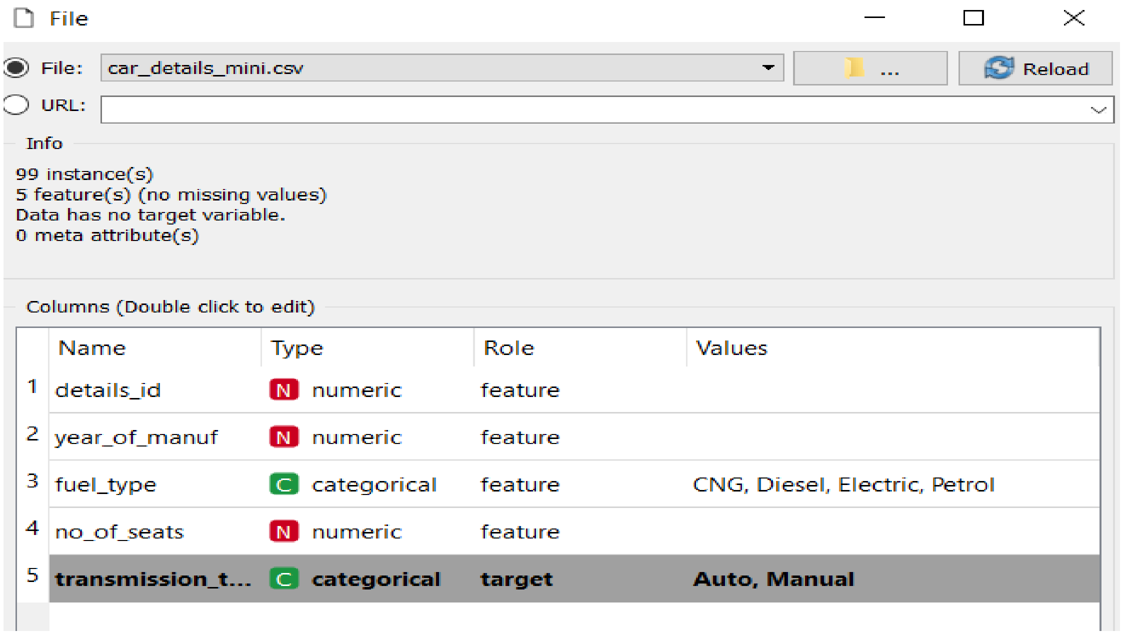
115 115

Silhouette Scores

2	0.712
3	0.676
4	0.555
5	0.498
6	0.463
7	0.415
8	0.409

2) SVM





File

● File: car_details_mini.csv

○ URL:

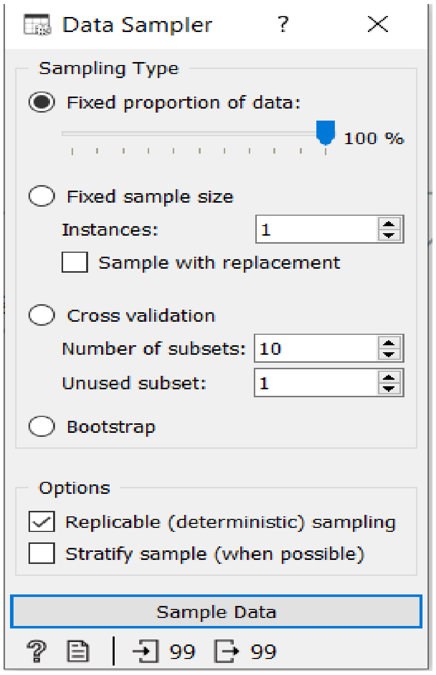
Info

99 instance(s)
5 feature(s) (no missing values)
Data has no target variable.
0 meta attribute(s)

Columns (Double click to edit)

	Name	Type	Role	Values
1	details_id	N numeric	feature	
2	year_of_manuf	N numeric	feature	
3	fuel_type	C categorical	feature	CNG, Diesel, Electric, Petrol
4	no_of_seats	N numeric	feature	
5	**transmission_t…**	**C categorical**	**target**	**Auto, Manual**

Data Sample



Data Sampler

Sampling Type

● Fixed proportion of data: 100 %

○ Fixed sample size

Instances: 1

□ Sample with replacement

○ Cross validation

Number of subsets: 10

Unused subset: 1

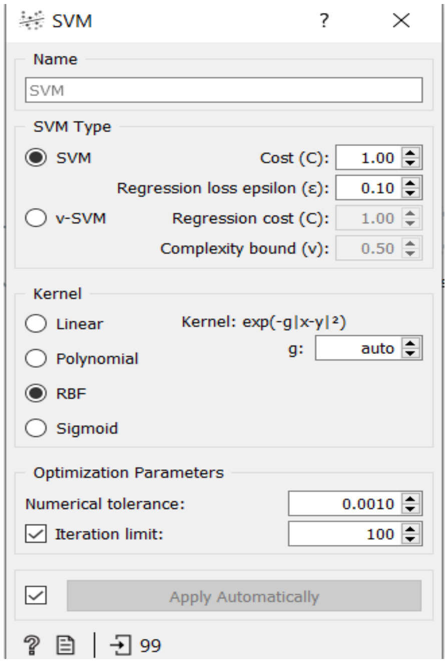
○ Bootstrap

Options

☑ Replicable (deterministic) sampling

□ Stratify sample (when possible)

SVM



SVM

Name

SVM

SVM Type

● SVM Cost (C): 1.00

Regression loss epsilon (ε): 0.10

○ ν-SVM Regression cost (C): 1.00

Complexity bound (ν): 0.50

Kernel

○ Linear Kernel: exp(-g|x-y|²)

g: auto

○ Polynomial

● RBF

○ Sigmoid

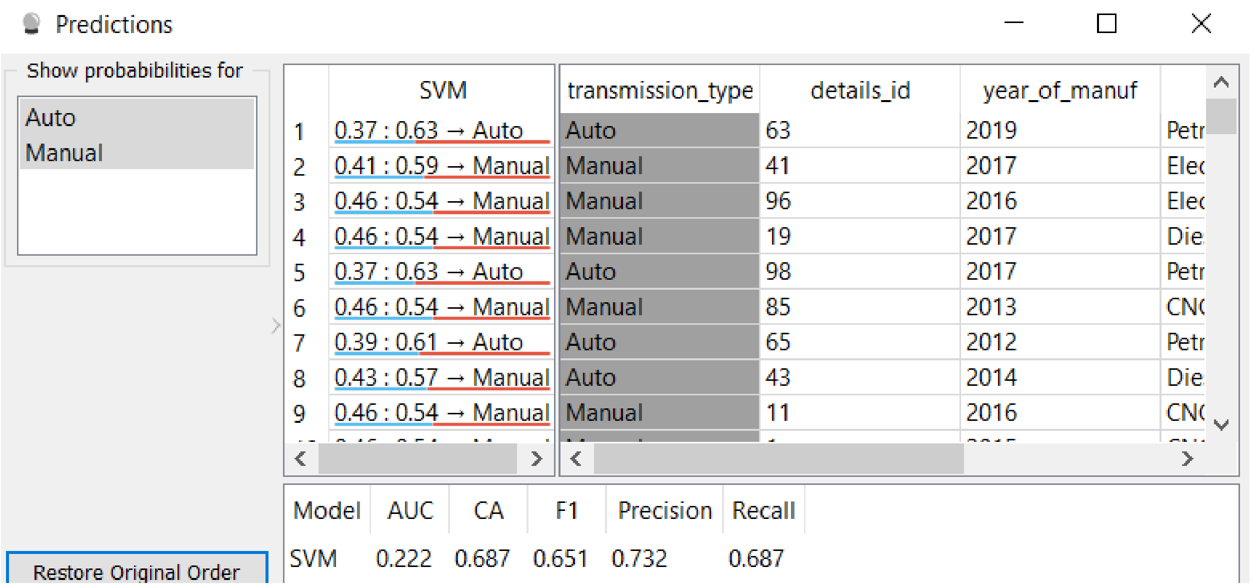
Optimization Parameters

Numerical tolerance: 0.0010

☑ Iteration limit: 100

☑ Apply Automatically

Predictions



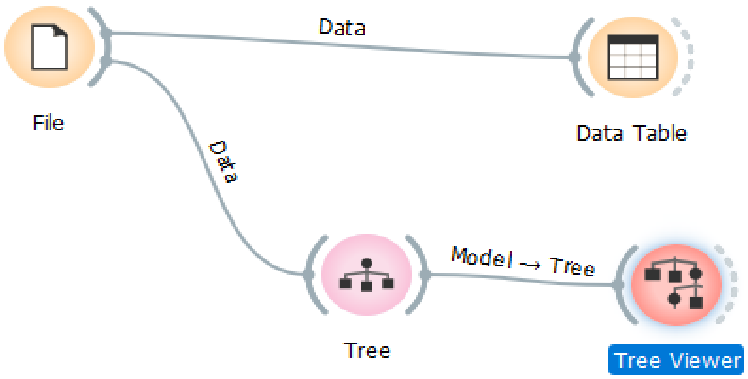
Show probabibilities for

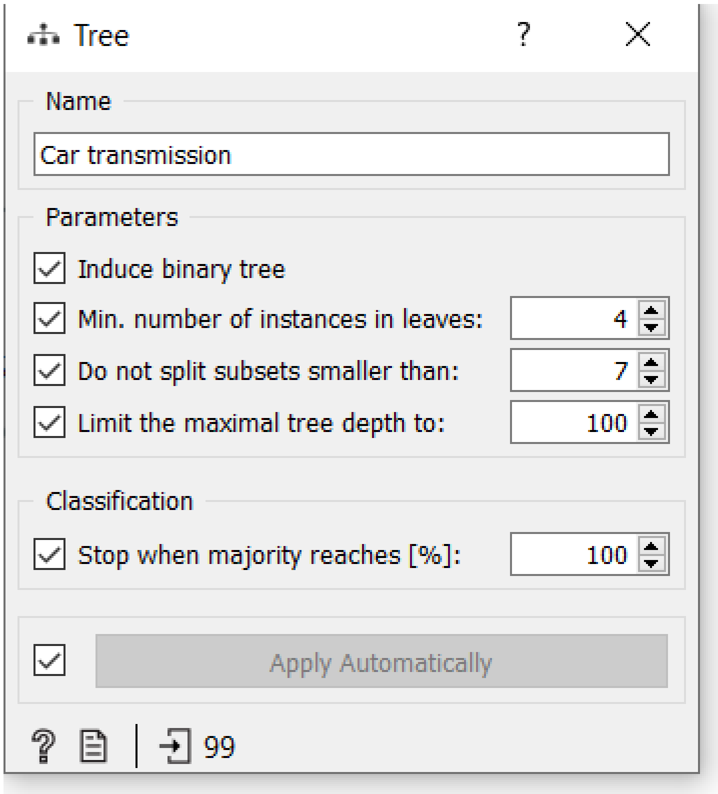
- Auto
- Manual

	SVM	transmission_type	details_id	year_of_manuf	
1	0.37 : 0.63 → Auto	Auto	63	2019	Petr
2	0.41 : 0.59 → Manual	Manual	41	2017	Elec
3	0.46 : 0.54 → Manual	Manual	96	2016	Elec
4	0.46 : 0.54 → Manual	Manual	19	2017	Die
5	0.37 : 0.63 → Auto	Auto	98	2017	Petr
6	0.46 : 0.54 → Manual	Manual	85	2013	CN(
7	0.39 : 0.61 → Auto	Auto	65	2012	Petr
8	0.43 : 0.57 → Manual	Auto	43	2014	Die
9	0.46 : 0.54 → Manual	Manual	11	2016	CN(

Model	AUC	CA	F1	Precision	Recall
SVM	0.222	0.687	0.651	0.732	0.687

3) Decision Tree





Tree ? ×

Name

Car transmission

Parameters

☑ Induce binary tree

☑ Min. number of instances in leaves: 4

☑ Do not split subsets smaller than: 7

☑ Limit the maximal tree depth to: 100

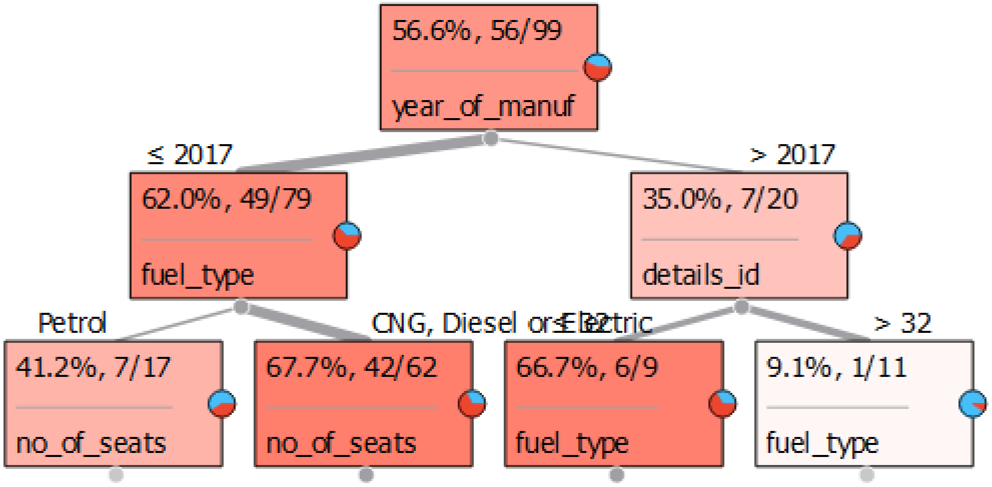
Classification

☑ Stop when majority reaches [%]: 100

☑ Apply Automatically

99

Tree Viewer - Manual Transmission (level 3)



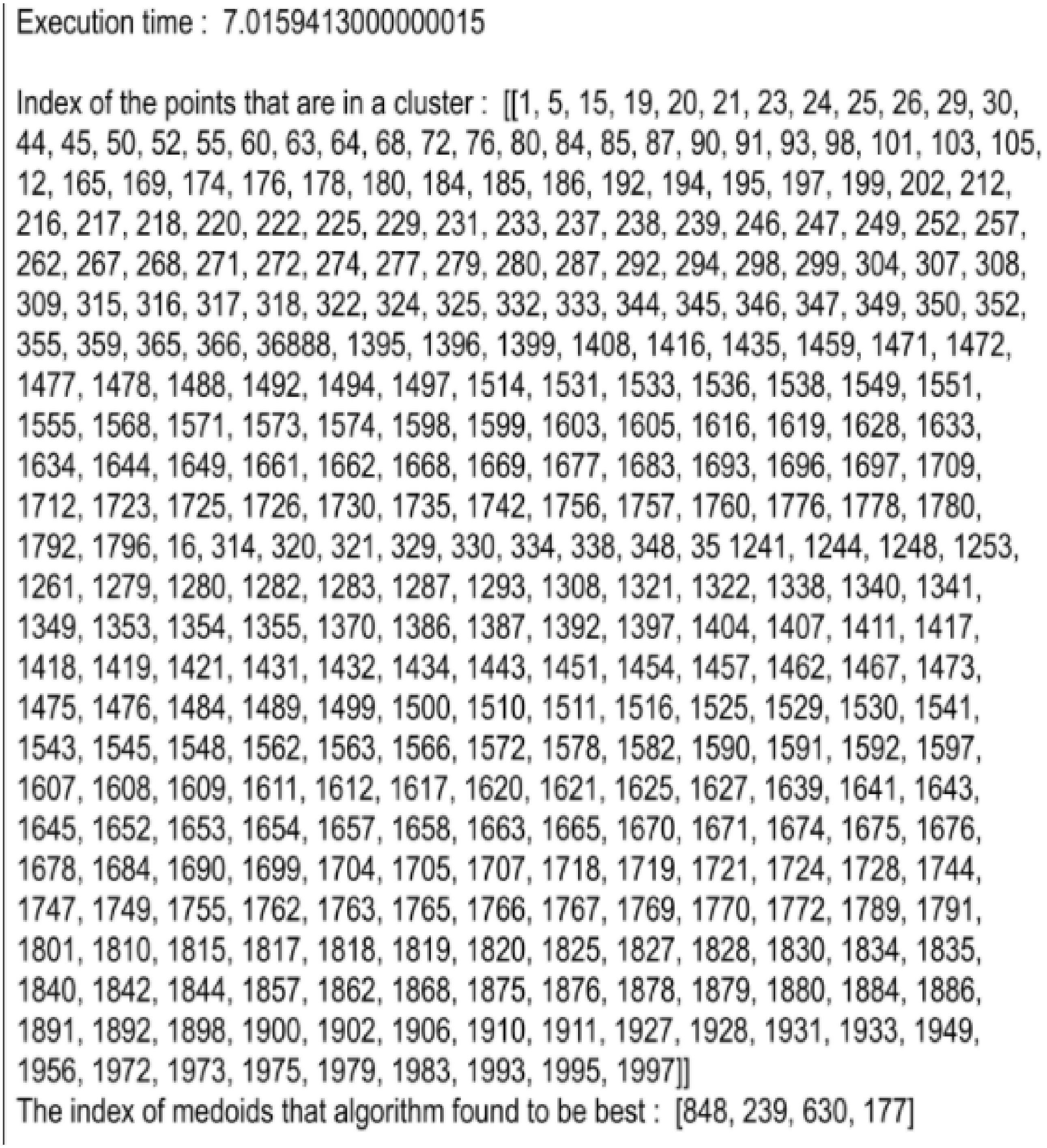
VIII. Spatial Clustering Algorithm - CLARANS Extensions

Clarans is a partitioning method of clustering algorithms. It is useful in recognising patterns and relationships in existing spatial data. We used this algorithm to understand the pattern in the parameters - year_of_manuf and no_of_seats in our dataset.

Code :

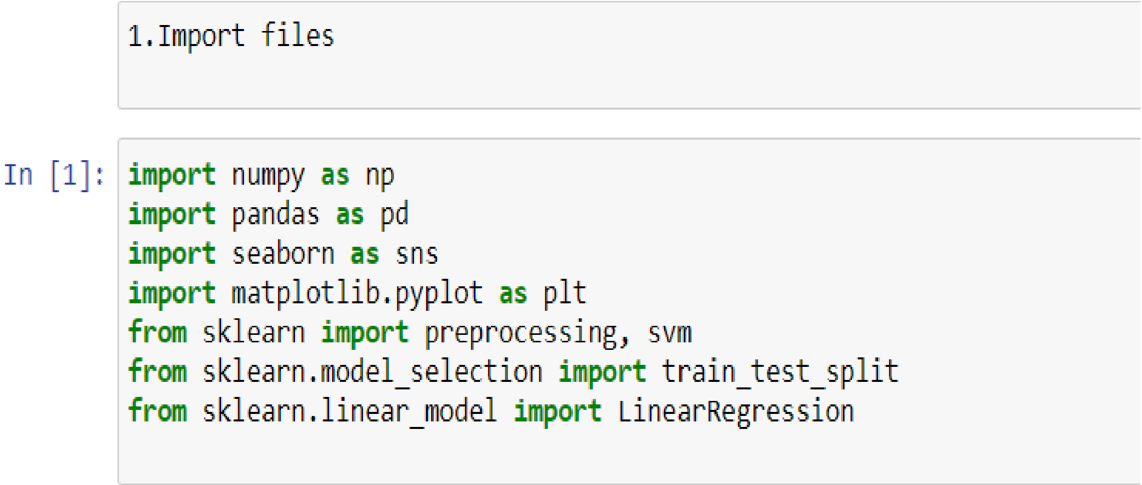
```
from pyclustering.cluster.clarans
import clarans from pyclustering.utils import timedcall
import pandas as pd
import mysql.connector
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
mydb = mysql.connector.connect
(
host="localhost",
user="root",
password="",
database="sale_analysis"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT year_of_manuf,no_of_seats FROM car_details")
x = mycursor.fetchall() data = x
clarans_instance = clarans(data, 4, 6, 4)
(ticks, result) = timedcall(clarans_instance.process)
print("Execution time : ", ticks, "\n")
clusters = clarans_instance.get_clusters()
medoids = clarans_instance.get_medoids()
print("Index of the points that are in a cluster : ",clusters)
print("The index of medoids that algorithm found to be best : ",medoids)
```

Output :



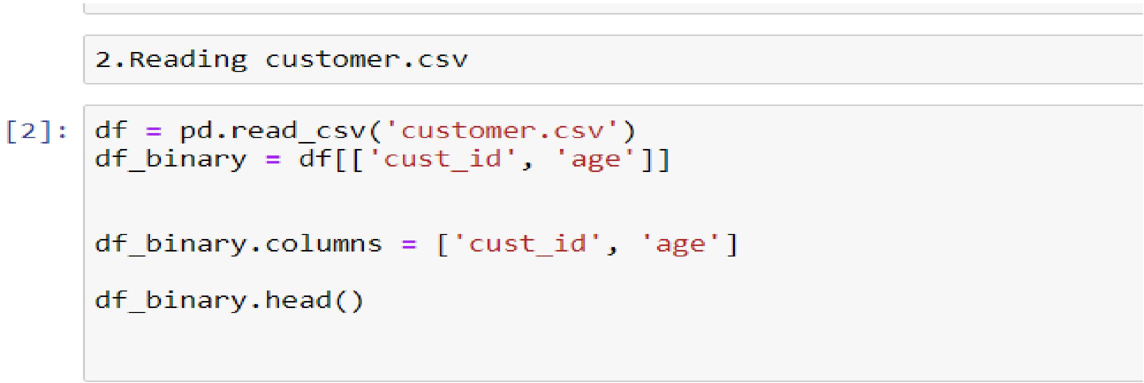
```
Execution time :  7.0159413000000015

Index of the points that are in a cluster :  [[1, 5, 15, 19, 20, 21, 23, 24, 25, 26, 29, 30,
44, 45, 50, 52, 55, 60, 63, 64, 68, 72, 76, 80, 84, 85, 87, 90, 91, 93, 98, 101, 103, 105,
12, 165, 169, 174, 176, 178, 180, 184, 185, 186, 192, 194, 195, 197, 199, 202, 212,
216, 217, 218, 220, 222, 225, 229, 231, 233, 237, 238, 239, 246, 247, 249, 252, 257,
262, 267, 268, 271, 272, 274, 277, 279, 280, 287, 292, 294, 298, 299, 304, 307, 308,
309, 315, 316, 317, 318, 322, 324, 325, 332, 333, 344, 345, 346, 347, 349, 350, 352,
355, 359, 365, 366, 36888, 1395, 1396, 1399, 1408, 1416, 1435, 1459, 1471, 1472,
1477, 1478, 1488, 1492, 1494, 1497, 1514, 1531, 1533, 1536, 1538, 1549, 1551,
1555, 1568, 1571, 1573, 1574, 1598, 1599, 1603, 1605, 1616, 1619, 1628, 1633,
1634, 1644, 1649, 1661, 1662, 1668, 1669, 1677, 1683, 1693, 1696, 1697, 1709,
1712, 1723, 1725, 1726, 1730, 1735, 1742, 1756, 1757, 1760, 1776, 1778, 1780,
1792, 1796, 16, 314, 320, 321, 329, 330, 334, 338, 348, 35 1241, 1244, 1248, 1253,
1261, 1279, 1280, 1282, 1283, 1287, 1293, 1308, 1321, 1322, 1338, 1340, 1341,
1349, 1353, 1354, 1355, 1370, 1386, 1387, 1392, 1397, 1404, 1407, 1411, 1417,
1418, 1419, 1421, 1431, 1432, 1434, 1443, 1451, 1454, 1457, 1462, 1467, 1473,
1475, 1476, 1484, 1489, 1499, 1500, 1510, 1511, 1516, 1525, 1529, 1530, 1541,
1543, 1545, 1548, 1562, 1563, 1566, 1572, 1578, 1582, 1590, 1591, 1592, 1597,
1607, 1608, 1609, 1611, 1612, 1617, 1620, 1621, 1625, 1627, 1639, 1641, 1643,
1645, 1652, 1653, 1654, 1657, 1658, 1663, 1665, 1670, 1671, 1674, 1675, 1676,
1678, 1684, 1690, 1699, 1704, 1705, 1707, 1718, 1719, 1721, 1724, 1728, 1744,
1747, 1749, 1755, 1762, 1763, 1765, 1766, 1767, 1769, 1770, 1772, 1789, 1791,
1801, 1810, 1815, 1817, 1818, 1819, 1820, 1825, 1827, 1828, 1830, 1834, 1835,
1840, 1842, 1844, 1857, 1862, 1868, 1875, 1876, 1878, 1879, 1880, 1884, 1886,
1891, 1892, 1898, 1900, 1902, 1906, 1910, 1911, 1927, 1928, 1931, 1933, 1949,
1956, 1972, 1973, 1975, 1979, 1983, 1993, 1995, 1997]]
The index of medoids that algorithm found to be best :  [848, 239, 630, 177]
```

IX. Linear Regression for age of customer



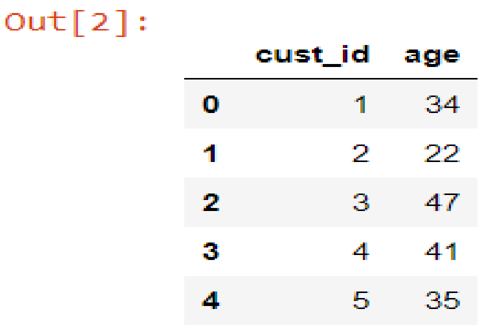
```
1.Import files
```

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn import preprocessing, svm
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
```



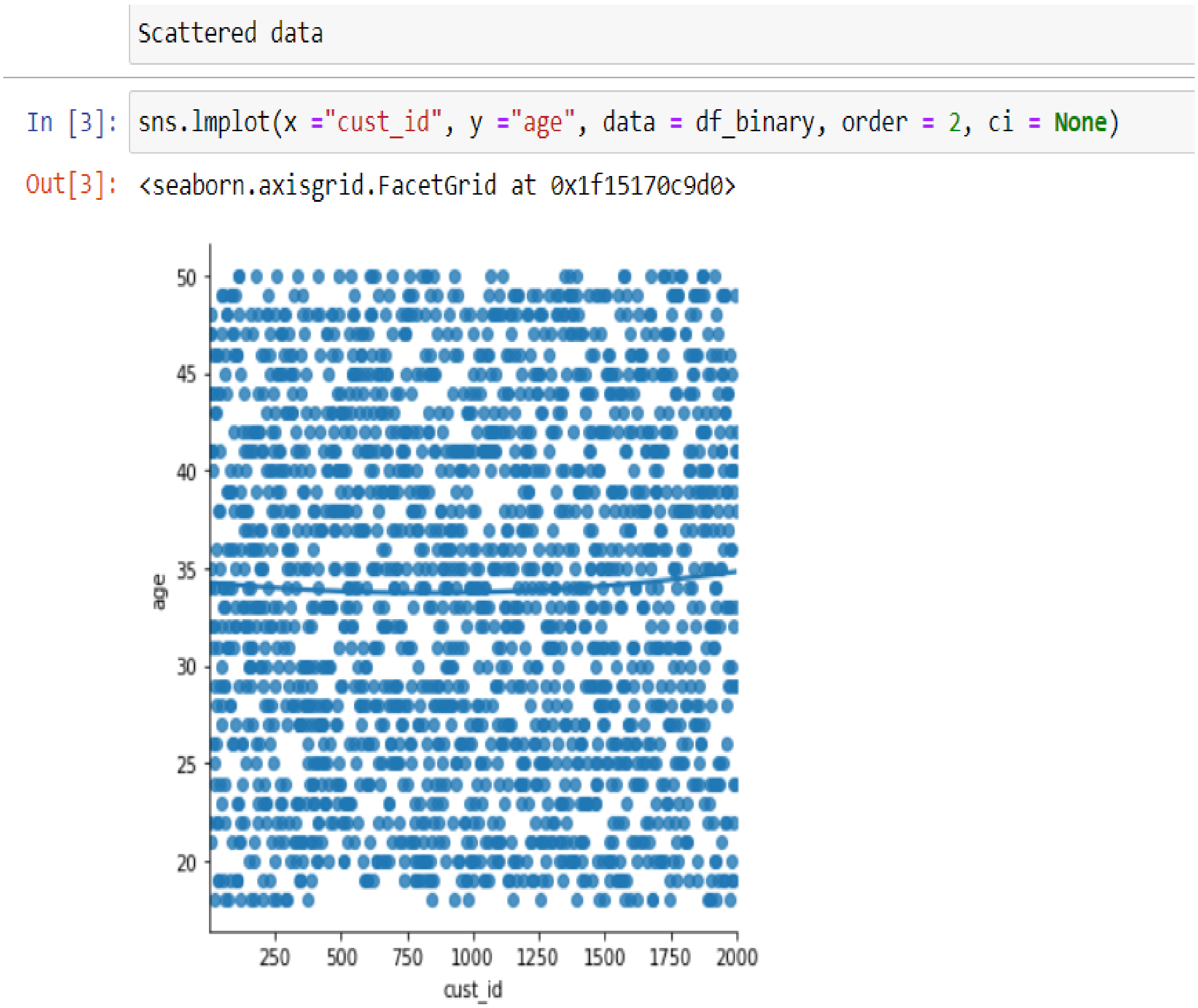
```
2.Reading customer.csv
```

```
[2]: df = pd.read_csv('customer.csv')
     df_binary = df[['cust_id', 'age']]


     df_binary.columns = ['cust_id', 'age']

     df_binary.head()
```



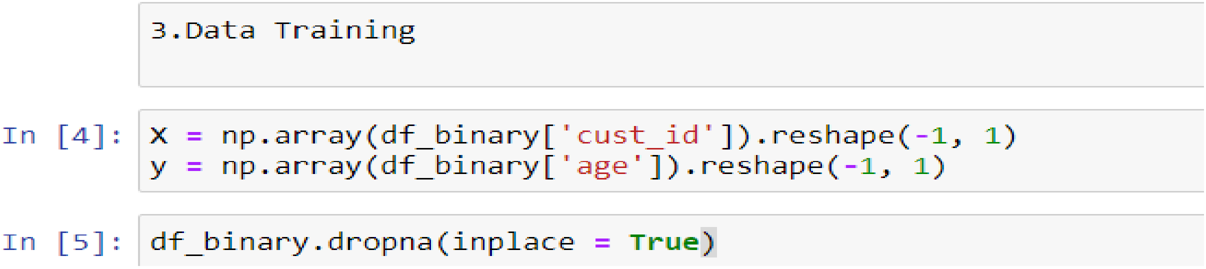
Out[2]:

	cust_id	age
0	1	34
1	2	22
2	3	47
3	4	41
4	5	35



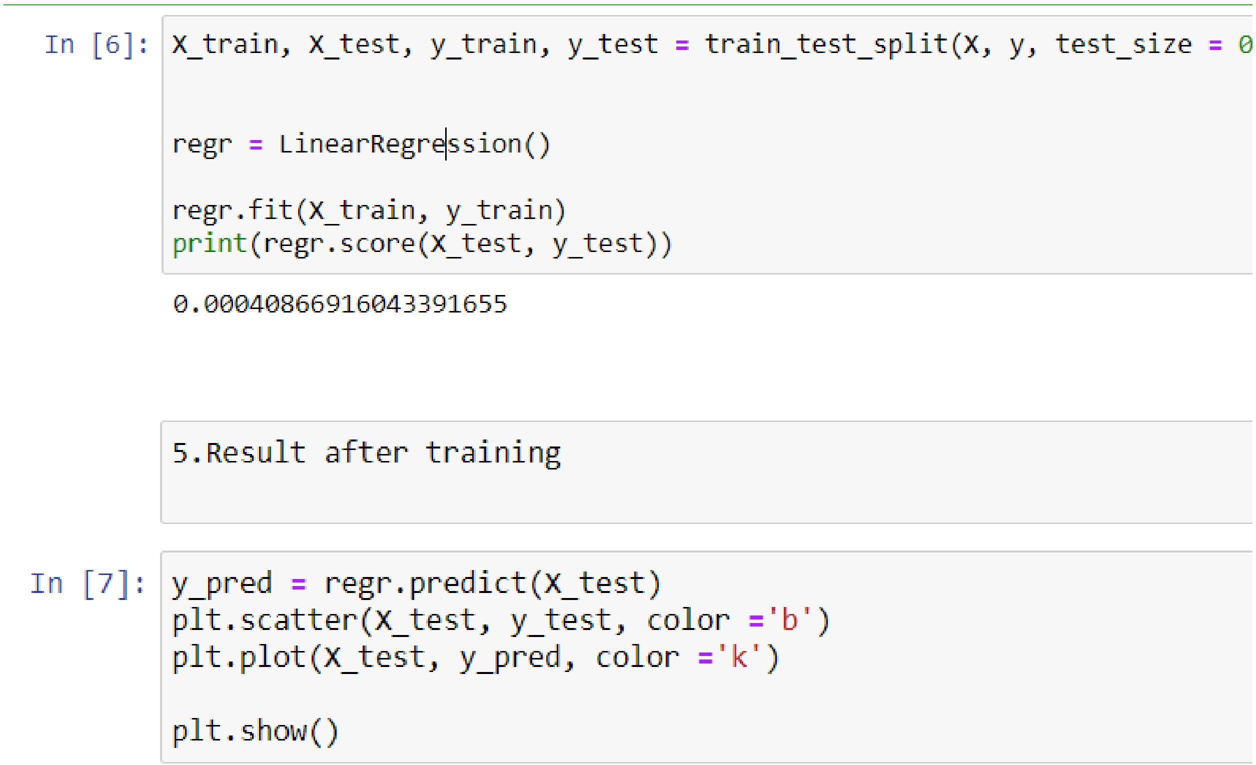
Scattered data

```
In [3]: sns.lmplot(x ="cust_id", y ="age", data = df_binary, order = 2, ci = None)

Out[3]: <seaborn.axisgrid.FacetGrid at 0x1f15170c9d0>
```



3.Data Training

```
In [4]: X = np.array(df_binary['cust_id']).reshape(-1, 1)
        y = np.array(df_binary['age']).reshape(-1, 1)

In [5]: df_binary.dropna(inplace = True)
```

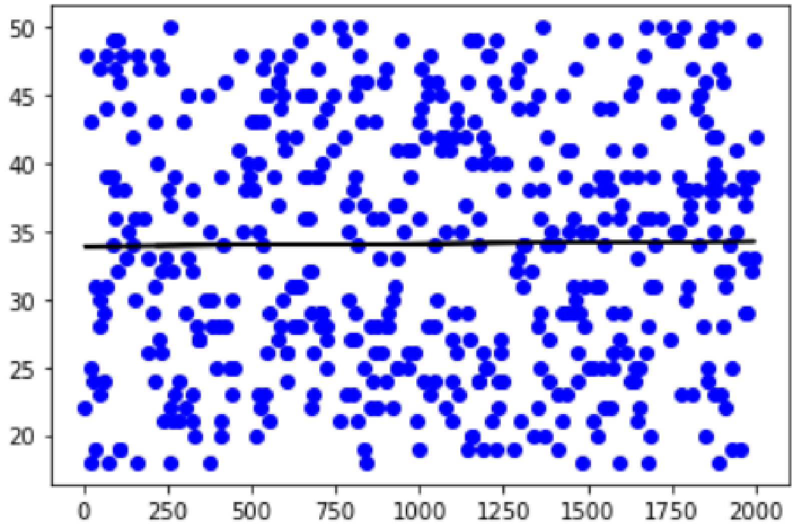


```
In [6]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0


        regr = LinearRegression()

        regr.fit(X_train, y_train)
        print(regr.score(X_test, y_test))
```

```
0.00040866916043391655
```

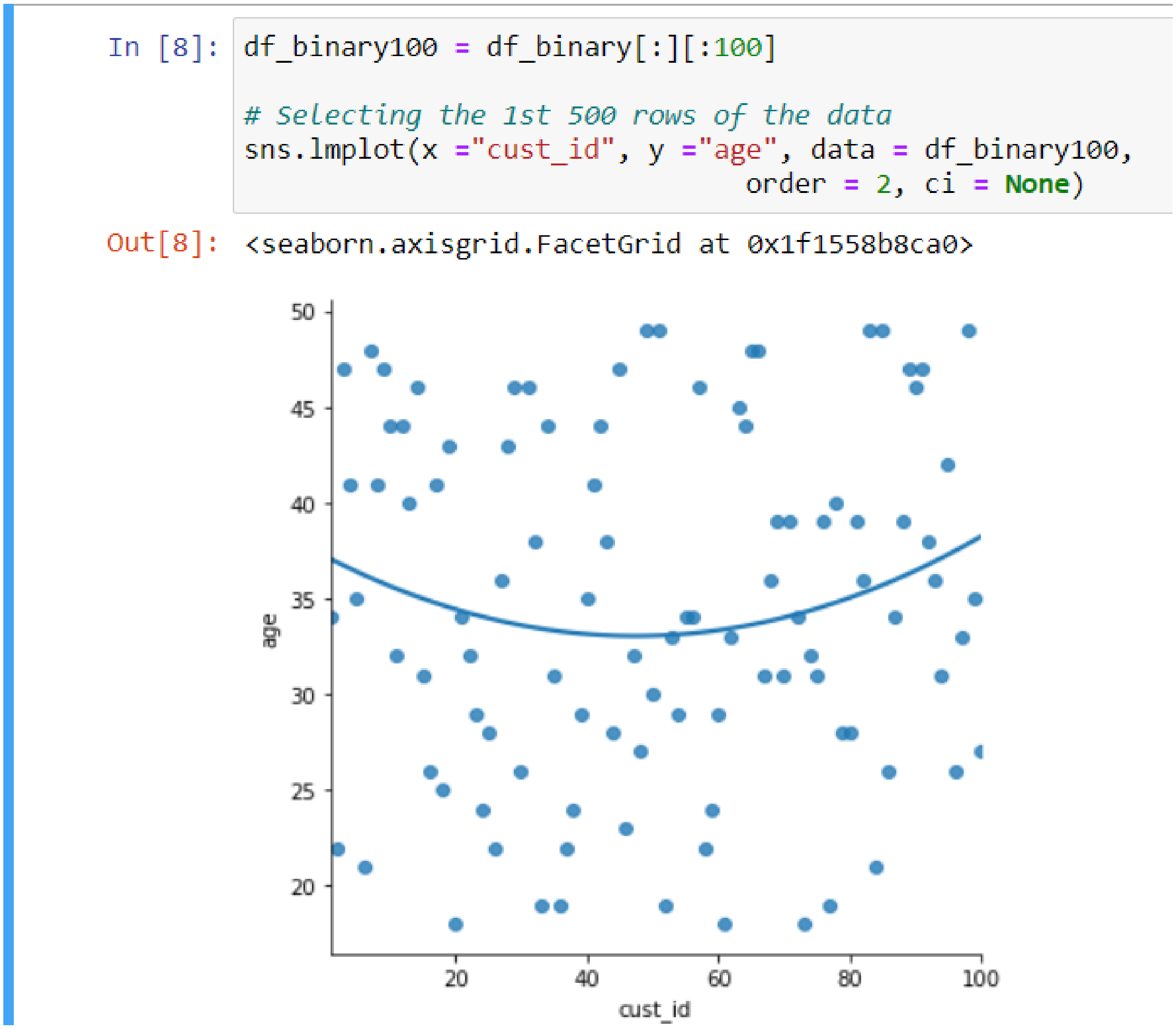
5.Result after training

```
In [7]: y_pred = regr.predict(X_test)
        plt.scatter(X_test, y_test, color ='b')
        plt.plot(X_test, y_pred, color ='k')

        plt.show()
```



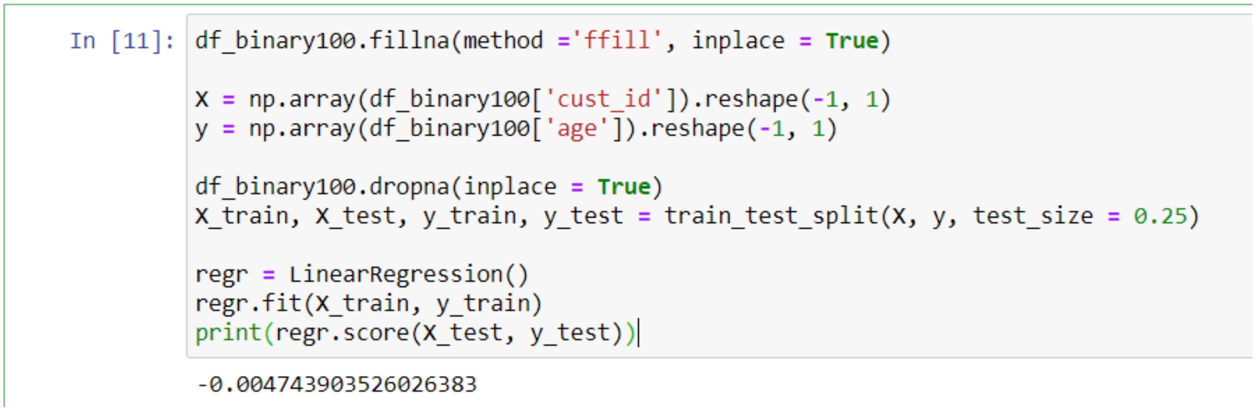
Working with a smaller dataset upto 100 and 500

1)100

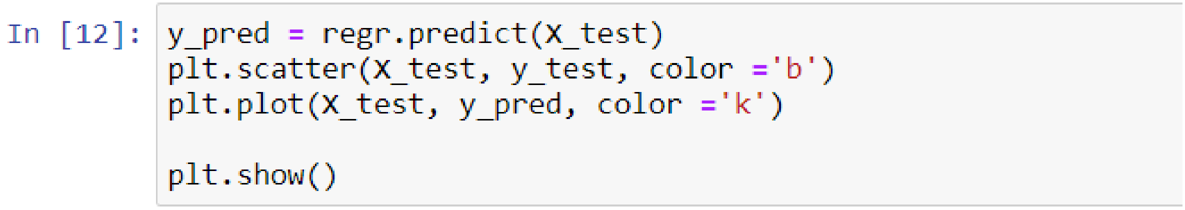


```
In [8]: df_binary100 = df_binary[:][:100]

        # Selecting the 1st 500 rows of the data
        sns.lmplot(x ="cust_id", y ="age", data = df_binary100,
                                        order = 2, ci = None)

Out[8]: <seaborn.axisgrid.FacetGrid at 0x1f1558b8ca0>
```

cust_id

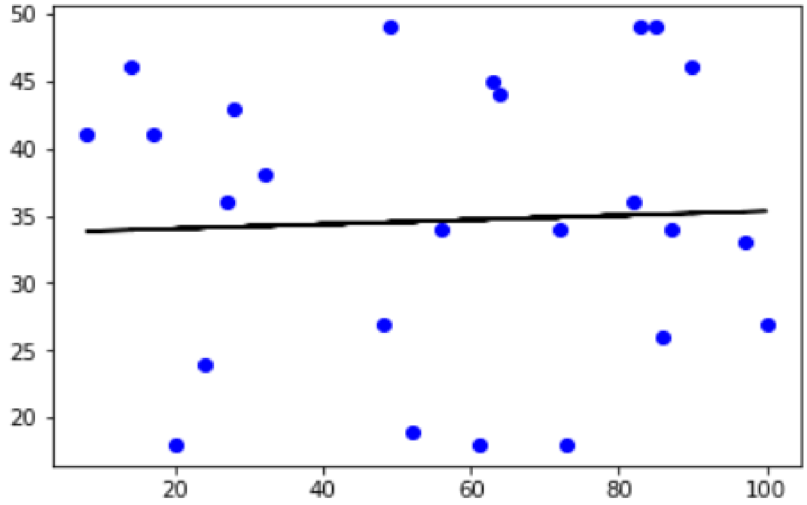


```
In [11]: df_binary100.fillna(method ='ffill', inplace = True)

         X = np.array(df_binary100['cust_id']).reshape(-1, 1)
         y = np.array(df_binary100['age']).reshape(-1, 1)

         df_binary100.dropna(inplace = True)
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

         regr = LinearRegression()
         regr.fit(X_train, y_train)
         print(regr.score(X_test, y_test))

         -0.004743903526026383
```

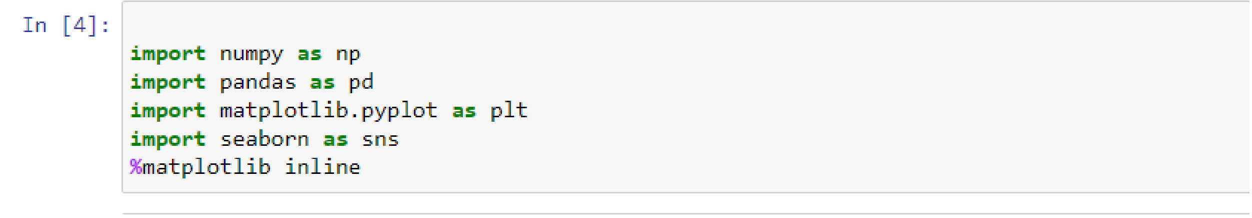


In [12]:

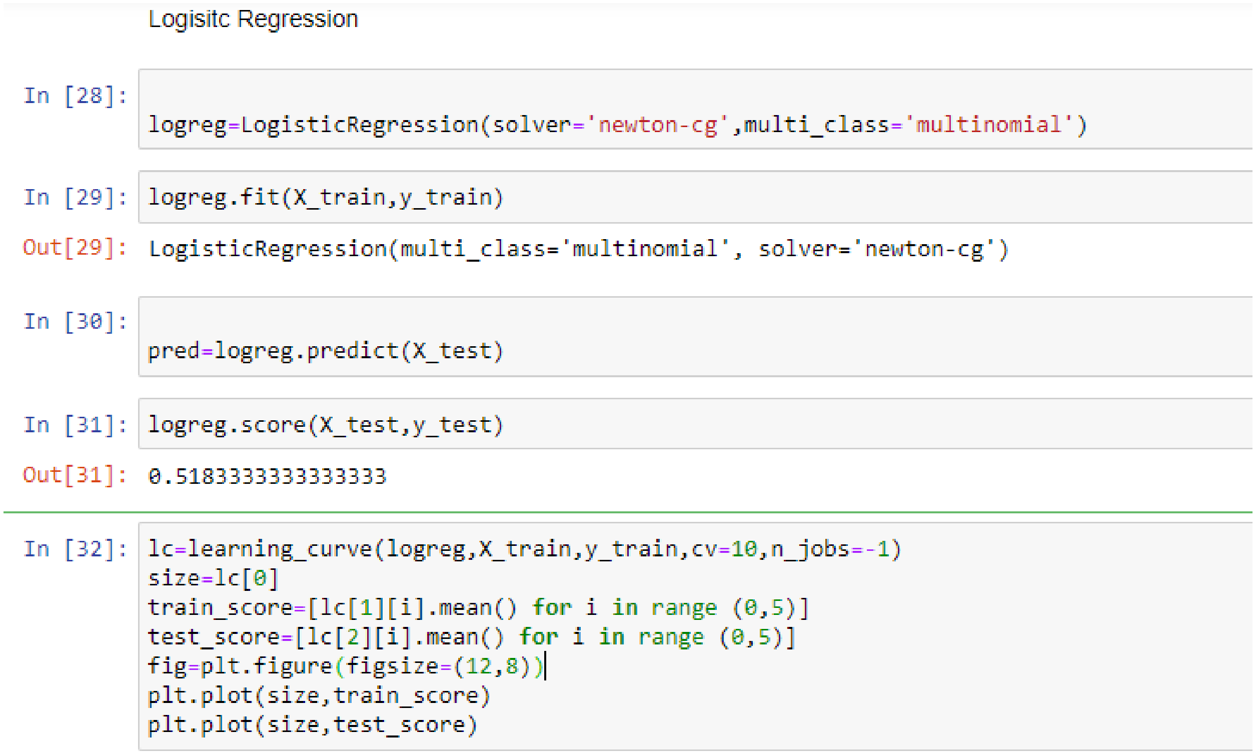
```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
```



X. Logistic Regression

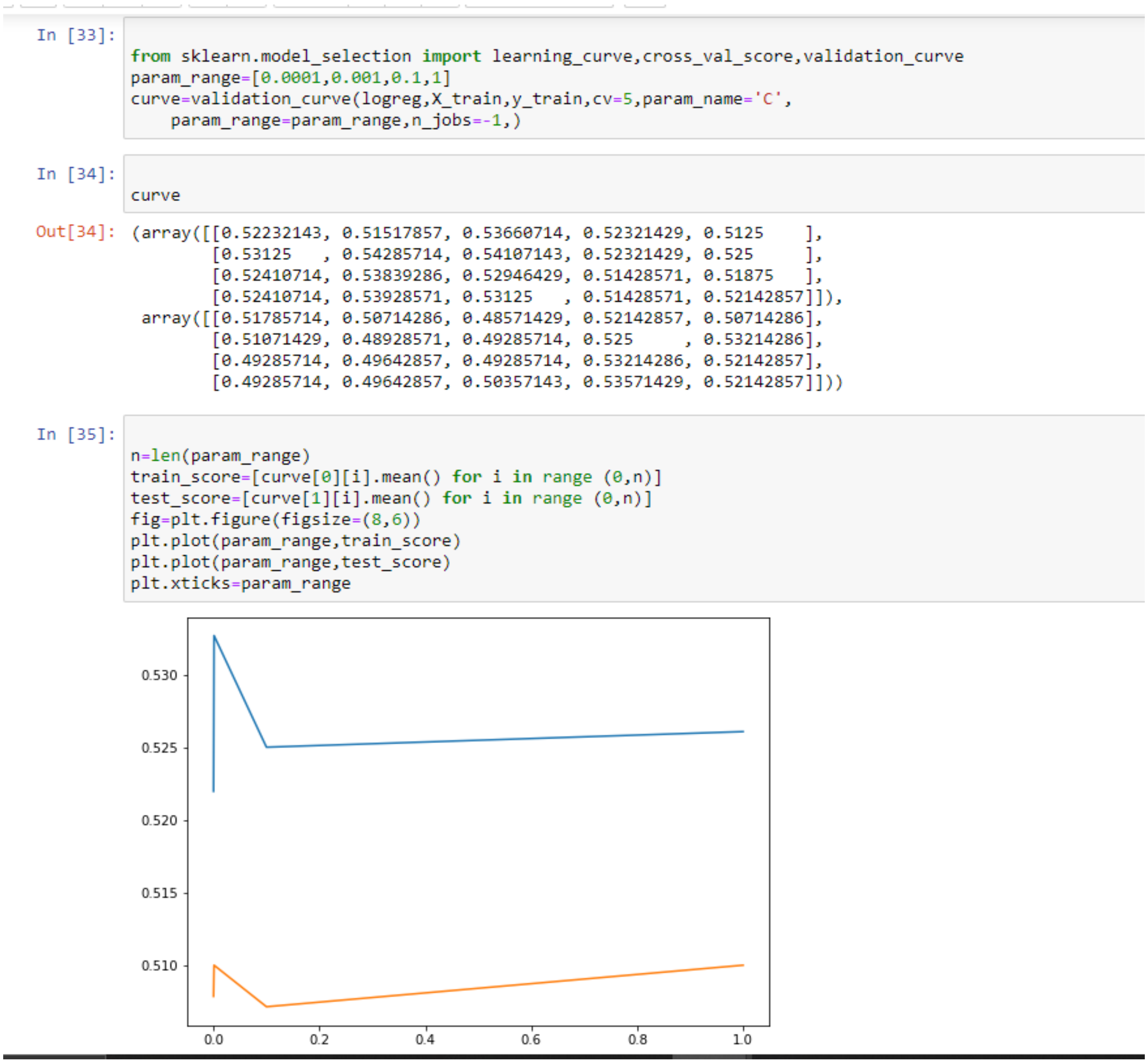


```
In [4]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```



Logisitc Regression

```
In [28]: logreg=LogisticRegression(solver='newton-cg',multi_class='multinomial')

In [29]: logreg.fit(X_train,y_train)

Out[29]: LogisticRegression(multi_class='multinomial', solver='newton-cg')

In [30]: pred=logreg.predict(X_test)

In [31]: logreg.score(X_test,y_test)

Out[31]: 0.5183333333333333

In [32]: lc=learning_curve(logreg,X_train,y_train,cv=10,n_jobs=-1)
         size=lc[0]
         train_score=[lc[1][i].mean() for i in range (0,5)]
         test_score=[lc[2][i].mean() for i in range (0,5)]
         fig=plt.figure(figsize=(12,8))
         plt.plot(size,train_score)
         plt.plot(size,test_score)
```

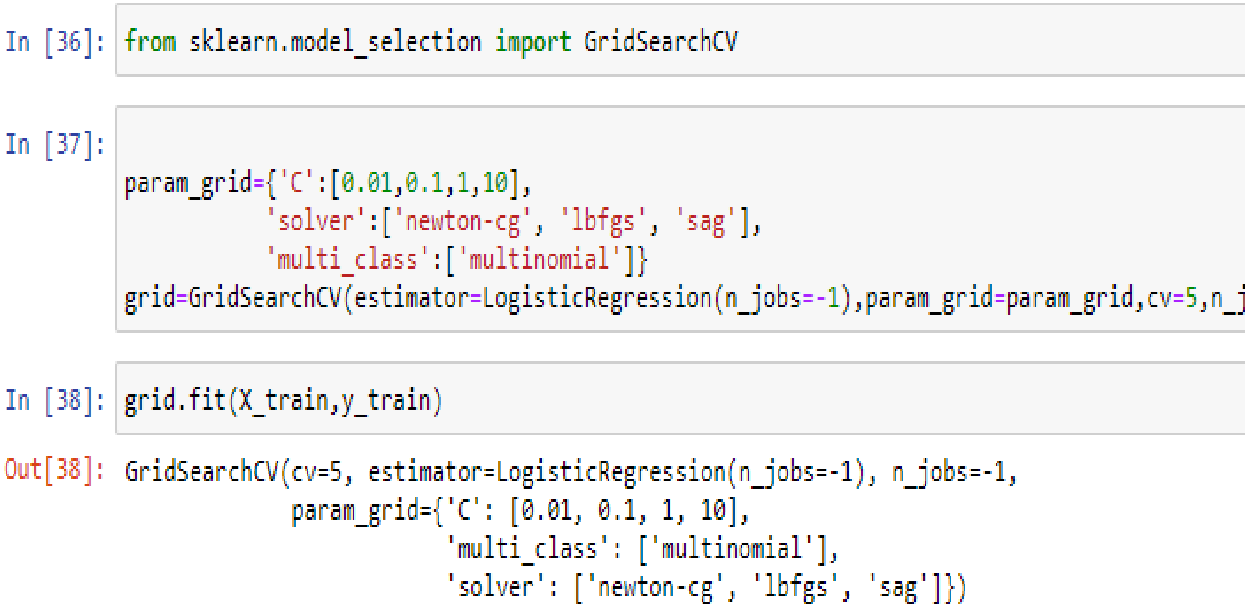


```
In [33]:
from sklearn.model_selection import learning_curve,cross_val_score,validation_curve
param_range=[0.0001,0.001,0.1,1]
curve=validation_curve(logreg,X_train,y_train,cv=5,param_name='C',
    param_range=param_range,n_jobs=-1,)
```

```
In [34]:
curve
```

```
Out[34]: (array([[0.52232143, 0.51517857, 0.53660714, 0.52321429, 0.5125    ],
        [0.53125   , 0.54285714, 0.54107143, 0.52321429, 0.525     ],
        [0.52410714, 0.53839286, 0.52946429, 0.51428571, 0.51875   ],
        [0.52410714, 0.53928571, 0.53125   , 0.51428571, 0.52142857]]),
 array([[0.51785714, 0.50714286, 0.48571429, 0.52142857, 0.50714286],
        [0.51071429, 0.48928571, 0.49285714, 0.525     , 0.53214286],
        [0.49285714, 0.49642857, 0.49285714, 0.53214286, 0.52142857],
        [0.49285714, 0.49642857, 0.50357143, 0.53571429, 0.52142857]]))
```

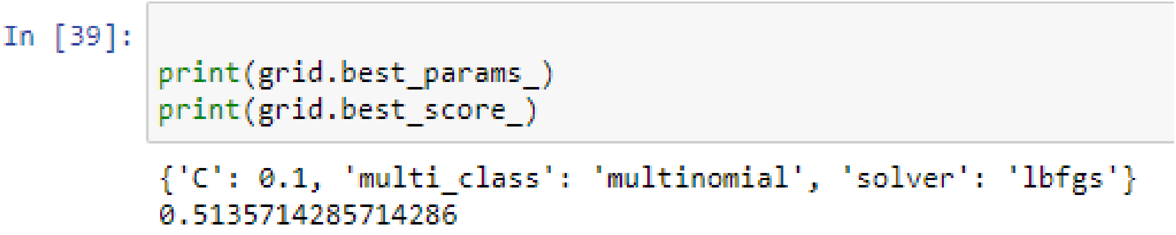
```
In [35]:
n=len(param_range)
train_score=[curve[0][i].mean() for i in range (0,n)]
test_score=[curve[1][i].mean() for i in range (0,n)]
fig=plt.figure(figsize=(8,6))
plt.plot(param_range,train_score)
plt.plot(param_range,test_score)
plt.xticks=param_range
```



```
In [36]: from sklearn.model_selection import GridSearchCV
```

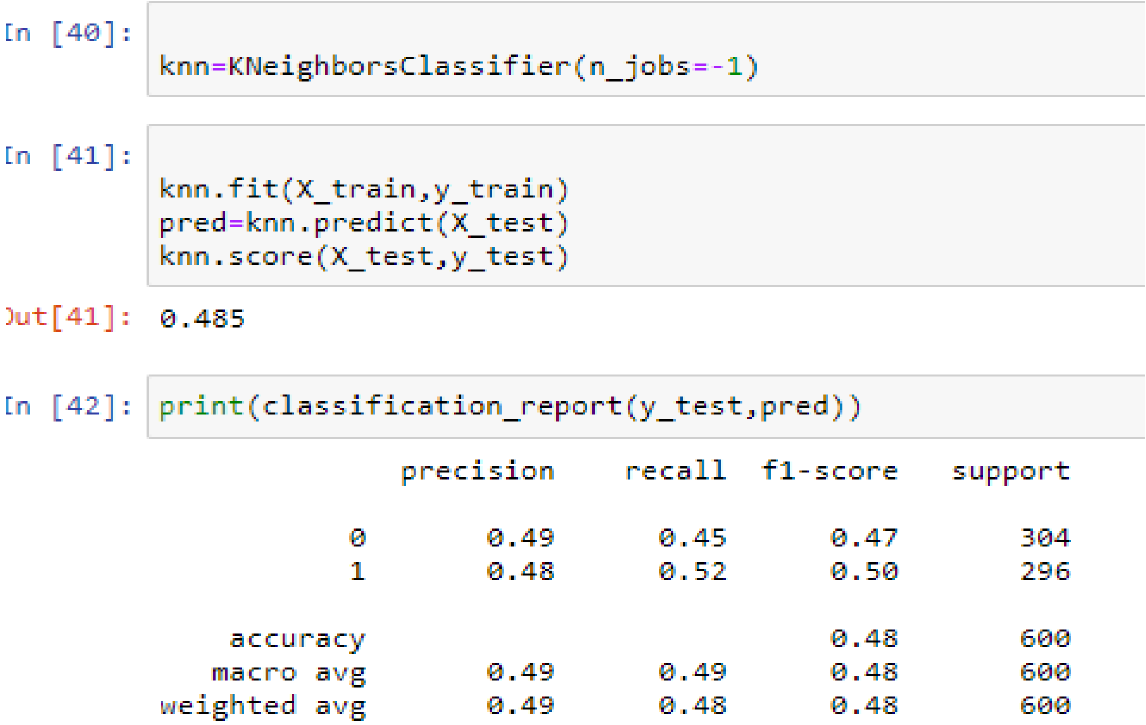
```
In [37]:
         param_grid={'C':[0.01,0.1,1,10],
                     'solver':['newton-cg', 'lbfgs', 'sag'],
                     'multi_class':['multinomial']}
         grid=GridSearchCV(estimator=LogisticRegression(n_jobs=-1),param_grid=param_grid,cv=5,n_j
```

```
In [38]: grid.fit(X_train,y_train)

Out[38]: GridSearchCV(cv=5, estimator=LogisticRegression(n_jobs=-1), n_jobs=-1,
                      param_grid={'C': [0.01, 0.1, 1, 10],
                                  'multi_class': ['multinomial'],
                                  'solver': ['newton-cg', 'lbfgs', 'sag']})
```



```
In [39]:
         print(grid.best_params_)
         print(grid.best_score_)

         {'C': 0.1, 'multi_class': 'multinomial', 'solver': 'lbfgs'}
         0.5135714285714286
```

XI. KNN Algorithm :



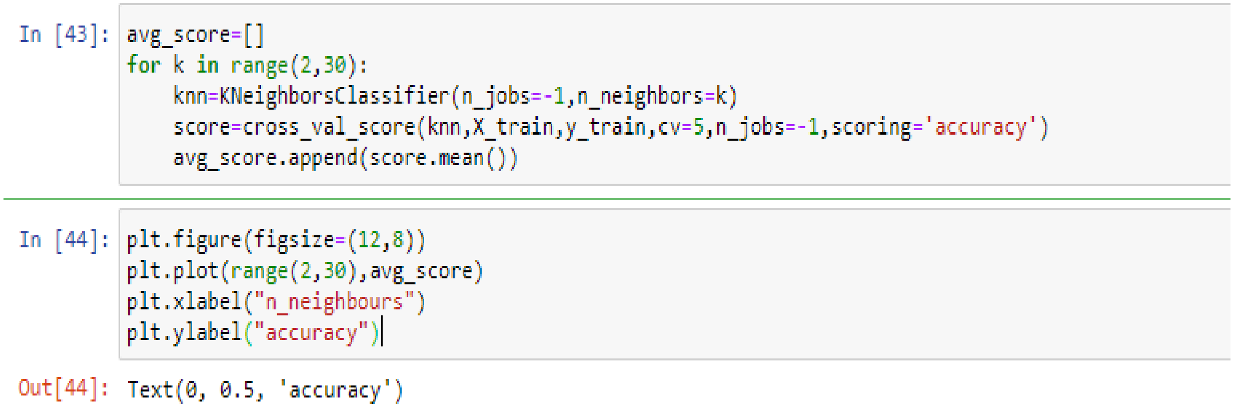
```
In [40]:
knn=KNeighborsClassifier(n_jobs=-1)
```

```
In [41]:
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
knn.score(X_test,y_test)
```

```
Out[41]: 0.485
```

```
In [42]: print(classification_report(y_test,pred))
```

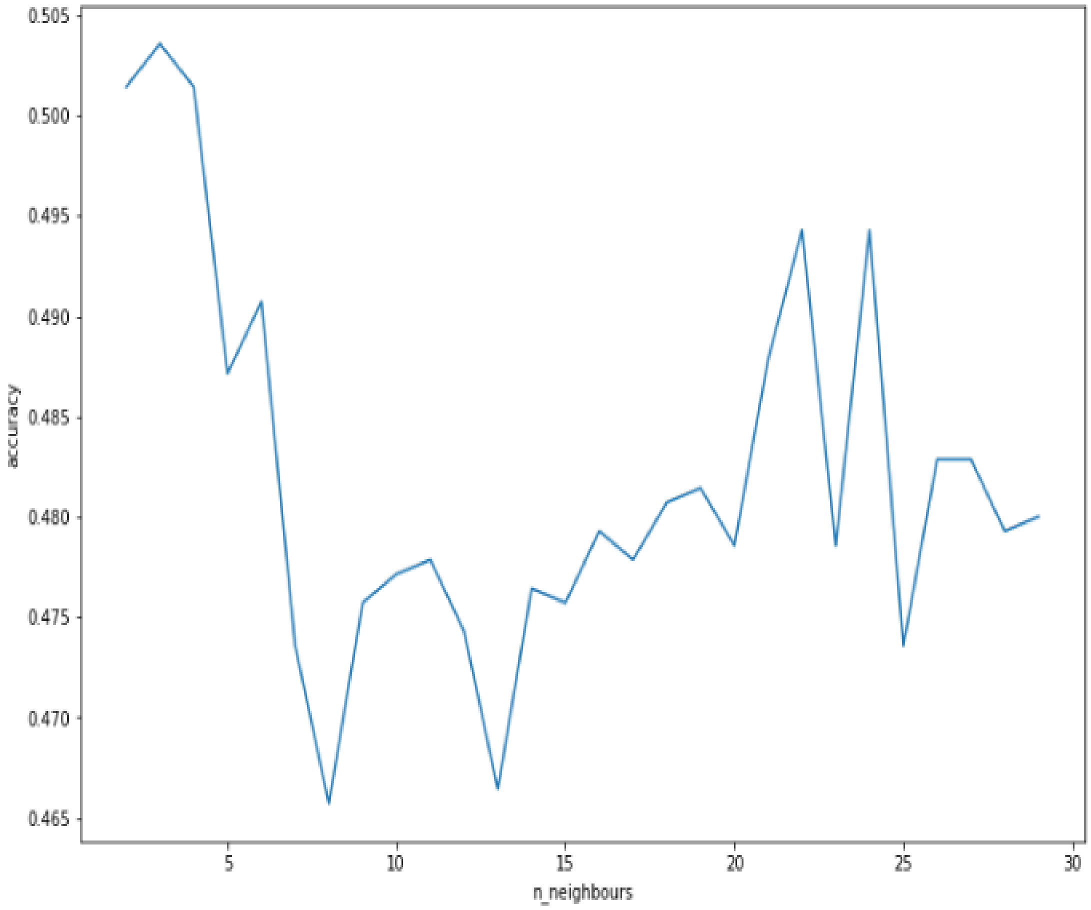
```
              precision    recall  f1-score   support

           0       0.49      0.45      0.47       304
           1       0.48      0.52      0.50       296

    accuracy                           0.48       600
   macro avg       0.49      0.49      0.48       600
weighted avg       0.49      0.48      0.48       600
```



```
In [43]: avg_score=[]
         for k in range(2,30):
             knn=KNeighborsClassifier(n_jobs=-1,n_neighbors=k)
             score=cross_val_score(knn,X_train,y_train,cv=5,n_jobs=-1,scoring='accuracy')
             avg_score.append(score.mean())
```

```
In [44]: plt.figure(figsize=(12,8))
         plt.plot(range(2,30),avg_score)
         plt.xlabel("n_neighbours")
         plt.ylabel("accuracy")
```

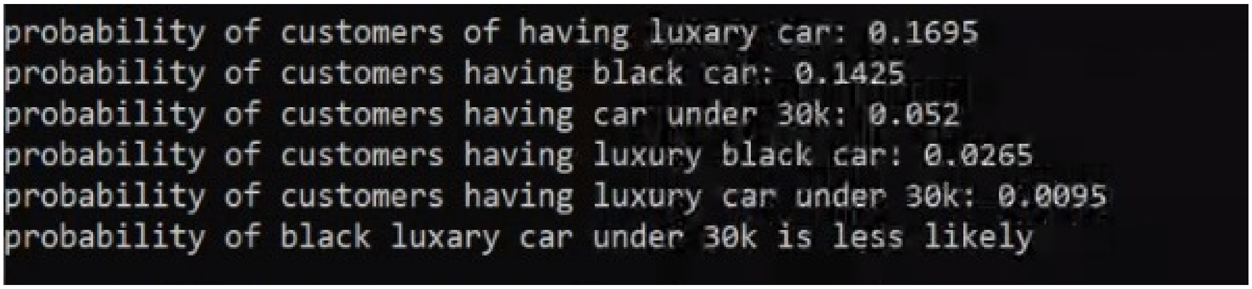
```
Out[44]: Text(0, 0.5, 'accuracy')
```



Results :

1) Naive Bayes :

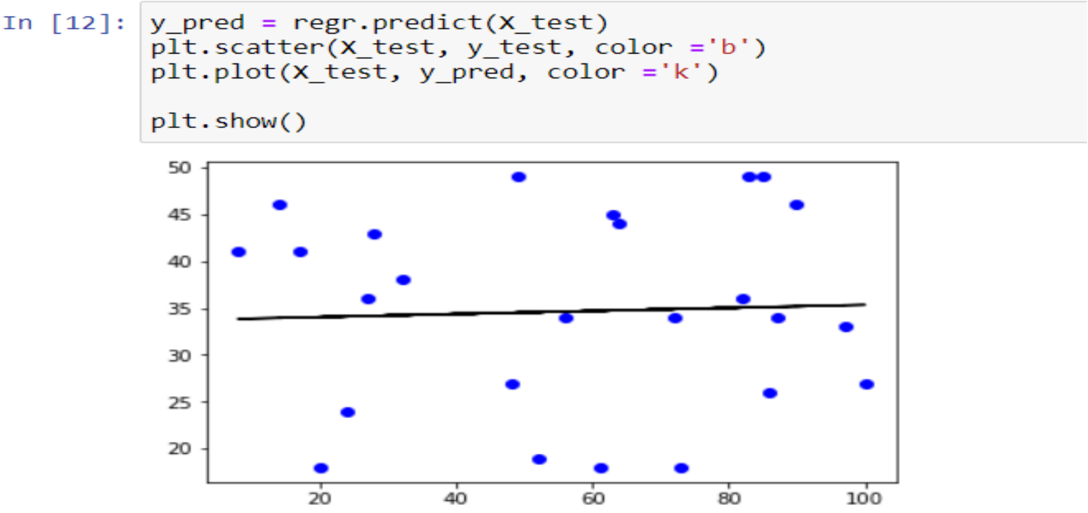
- Naïve Bayes Algorithm is a supervised learning algorithm, which is based on Bayes Theorem and is used for solving classification problems.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- We used this model to calculate the prediction of “Luxury car ”, using which is “Black” in color and under price “30k”.



```
probability of customers of having luxary car: 0.1695
probability of customers having black car: 0.1425
probability of customers having car under 30k: 0.052
probability of customers having luxury black car: 0.0265
probability of customers having luxury car under 30k: 0.0095
probability of black luxary car under 30k is less likely
```

2) Linear Regression :

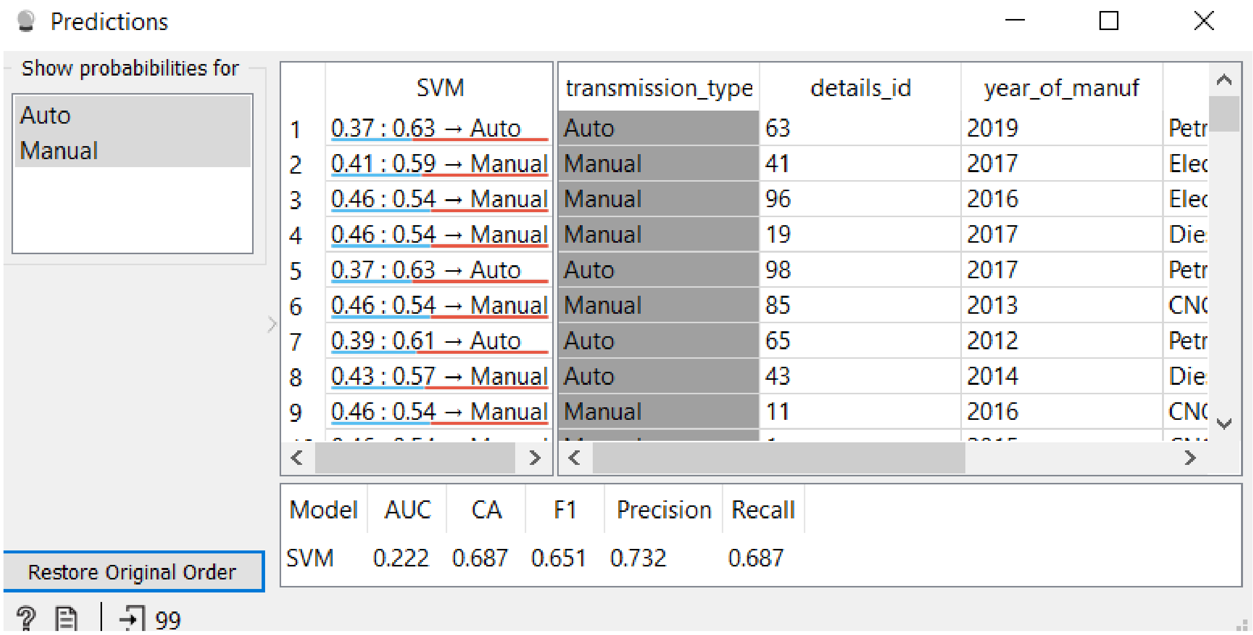
- Simple linear regression is useful for finding relationships between two continuous variables. One is predictor or independent variable and the other is response or dependent variable. It looks for statistical relationships but not deterministic relationships.
- Here we predicted the age of the customer who is willing to rent a car.



```
In [12]: y_pred = regr.predict(X_test)
         plt.scatter(X_test, y_test, color ='b')
         plt.plot(X_test, y_pred, color ='k')

         plt.show()
```

3) SVM :

- Support vector machine is a classifying algorithm and we implemented this
- algorithm to classify whether the mode of transmission (Auto/Manual) is based on the (year_of_manuf, fuel_type, no_of_seats, car_type) attributes



Predictions

Show probabilities for

Auto
Manual

Restore Original Order

99

	SVM	transmission_type	details_id	year_of_manuf	
1	0.37 : 0.63 → Auto	Auto	63	2019	Petr
2	0.41 : 0.59 → Manual	Manual	41	2017	Elec
3	0.46 : 0.54 → Manual	Manual	96	2016	Elec
4	0.46 : 0.54 → Manual	Manual	19	2017	Die
5	0.37 : 0.63 → Auto	Auto	98	2017	Petr
6	0.46 : 0.54 → Manual	Manual	85	2013	CN(
7	0.39 : 0.61 → Auto	Auto	65	2012	Petr
8	0.43 : 0.57 → Manual	Auto	43	2014	Die
9	0.46 : 0.54 → Manual	Manual	11	2016	CN(

Model	AUC	CA	F1	Precision	Recall
SVM	0.222	0.687	0.651	0.732	0.687

Conclusion

The web based car rental system is an application that allows customers to access and use a wide range of available cars for a particular period of time. It has offered an advantage to both customers as well as to the company to efficiently manage the business and satisfy customers' requirements. It will verify and store the information of the customers while booking a car. Customers can book a car online with the required car specification on a particular date and time.

Using the star and snowflake schemas, Olap queries and various algorithms for classification, clustering and prediction makes it easier to generate reports and analyse the process about the sales, stocks and availability of cars according to different categories, brand, locations and at a specific time period. This software provides an easy-to-use interface that allows simple access from browsing cars to booking requests.