A Project Report

on

Automobile Price Prediction

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

Under The Supervision of
Mr. Hradesh Kumar
Assistant Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1010738 - Anushka Sharma
18SCSE1010362 - Ishika Gupta

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project entitled **" Automobile Price Prediction "** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Mr. Hradesh Kumar, Assistant Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010738 - Anushka Sharma
18SCSE1010362 - Ishika Gupta

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name

(Mr. Hradesh Kumar, Assistant Professor)

CERTIFICATE

The Final Project Viva-Voce examination of **18SCSE1010738 – Anushka Sharma, 18SCSE1010362 - Ishika Gupta** has been held on _18/12/2021_ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Date: 13/12/2021

Place: Greater Noida

Abstract

India has one of the biggest automobile markets all over the globe every day many buyers usually sell their cars after using for the time to another buyer, we call them as 2nd /3rd owner etc. Many platforms such as cars24.com, cardekho.com and OLX.com provides these buyers with a platform where they can sell their used cars, but what should be the price of the car, this is the toughest question ever. Machine Learning algorithms can bring a solution to this problem. Using a history of previously used cars selling data and using machine learning techniques such as Supervised Learning can predict a fair price of the car, here I also used machine learning algorithms such as Random Forest and Extra Tree Regression along with powerful python library Scikit-Learn to predict the selling price of the used car. The result has shown that these both algorithms are highly accurate in prediction even the dataset is large or small, irrespective of the size of the dataset they give a precise result.

Keywords: Machine Learning, Supervised Learning, Random Forest, Extra Tress Regression, RandomSearchCV , Algorithm, Kaggle, Car dataset, cardekho.

Table of Contents

List of Figures

Acronyms

B.Tech.	Bachelor of Technology
ML	Machine Learning
SL	Supervised learning
SVM	Support Vector Machine

CHAPTER-1

Introduction

Car price prediction is somehow interesting and popular problem. As per information that was gotten from the Agency for Statistics of BiH, 921.456 vehicles were registered in 2014 from which 84% of them are cars for personal usage. This number is increased by 2.7% since 2013 and it is likely that this trend will continue, and the number of cars will increase in future. This adds additional significance to the problem of the car price prediction. Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors. Typically, most significant ones are brand and model, age, horsepower and mileage. The fuel type used in the car as well as fuel consumption per mile highly affect price of a car due to a frequent change in the price of a fuel. Different features like exterior color, door number, type of transmission, dimensions, safety, air condition, interior, whether it has navigation or not will also influence the car price. In this paper, we applied different methods and techniques in order to achieve higher precision of the used car price prediction.

Predicting the resale value of a car is not a simple task. It is trite knowledge that the value of used cars depends on a number of factors. The most important ones are usually the age of the car, its make (and model), the origin of the car (the original country of the manufacturer), its mileage (the number of kilometers it has run) and its horsepower. Due to rising fuel prices, fuel economy is also of prime importance. Unfortunately, in practice, most people do not know exactly how much fuel their car consumes for each km driven. Other factors such as the type of fuel it uses, the interior style, the braking system, acceleration, the volume of its cylinders (measured in cc), safety index, its size, number of doors, paint colour, weight of the car, consumer

reviews, prestigious awards won by the car manufacturer, its physical state, whether it is a sports car, whether it has cruise control, whether it is automatic or manual transmission, whether it belonged to an individual or a company and other options such as air conditioner, sound system, power steering, cosmic wheels, GPS navigator all may influence the price as well. Some special factors which buyers attach importance in Mauritius is the local of previous owners, whether the car had been involved in serious accidents and whether it is a lady-driven car. The look and feel of the car certainly contribute a lot to the price. As we can see, the price depends on a large number of factors. Unfortunately, information about all these factors are not always available and the buyer must make the decision to purchase at a certain price based on few factors only.

CHAPTER-2

Literature Survey

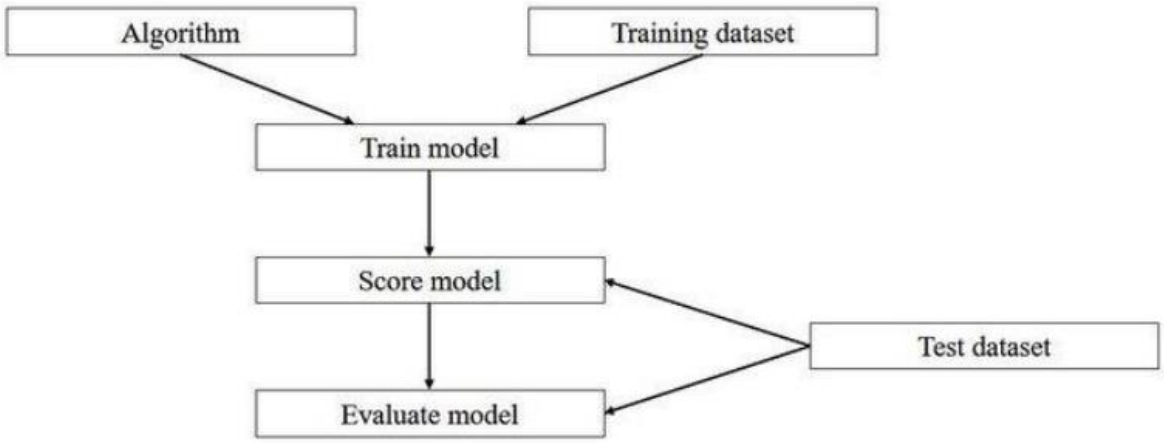
In this project, we investigate the application of supervised machine learning techniques to predict the price of used cars. The predictions are based on historical data collected from kaggle. Different techniques like Random Forest, multiple linear regression analysis, k-nearest neighbours, naïve bayes and decision trees have been used to make the predictions. The Considerable number of distinct attributes are examined for the reliable and accurate prediction, we have applied three machine learning techniques (Artificial Neural Network, Support Vector Machine and Random Forest). Model in second hand car system based on neural networks. In this paper, the price evaluation model based on big data analysis is proposed, which takes advantage of widely circulated vehicle data and a large number of vehicle transaction data to analyze the price data for each type of vehicles by using the optimized neural network algorithm. It aims to establish a second-hand car price evaluation model to get the price that best matches the car.

The Problem

The prices of new cars 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 price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase .There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best. Besides, different models and systems may

contribute on predicting power for a used car's actual market value. It is important to know their actual market value while both buying and selling.

DFD diagram



CHAPTER 3

Working of Project

The Client

To be able to predict used cars market value can help both buyers and sellers.

Used car sellers (dealers): They are one of the biggest target group that can be interested in results of this study. If used car sellers better understand what makes a car desirable, what the important features are for a used car, then they may consider this knowledge and offer a better service.

The Data

The data used in this project was downloaded from Kaggle. This dataset contains information about used cars.This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning. The columns in the given dataset are as follows:

1. name
2. year
3. selling_price
4. km_driven
5. fuel
6. seller_type
7. transmission
8. Owner

Data Wrangling

In this section, it will be discussed about how data cleaning and wrangling methods are applied on the used cars data file.Before making data cleaning, some explorations and data visualizations were applied on data set. This gave some idea and guide about how to deal with missing values and extreme values. After data cleaning, data exploration was applied again in order to understand cleaned version of the data.

Data cleaning: First step for data cleaning was to remove unnecessary features. For this purpose, 'url','image_url', 'lat', 'long', 'city_url', 'desc', 'city', 'VIN' features were dropped totally. As a next step, it was investigated number of null points and percentage of null data points for that feature . As the second step, some missing values were filled with appropriate values. For the missing 'condition' values, it was paid attention to fill depending on category. Average odometer of all 'condition' sub-categories were calculated. Then, missing values were filled by considering this average odometer values for each condition sub-category. In addition, cars that have model value higher than 2019 were filled as 'new', and between 2017–2019 were filled as 'like new'. At the end of this process, all missing values in 'condition' feature were cleaned.

The Exploratory Data Analysis (EDA)

While exploring the data, we will look at the different combinations of features with the help of visuals. This will help us to understand our data better and give us some clue about pattern in data.

An examination of price trend

Price is the feature that we are predicting in this study. Before applying any models, taking a look at price data may give us some ideas.

SQL

Structured Query Language (SQL) is an indispensable skill in the data science industry and generally speaking, learning this skill is relatively straightforward. However, most forget that SQL isn't just about writing queries, which is just the first step down the road. Ensuring that queries are performant or that they fit the context that you're working in is a whole other thing.

That's why this SQL tutorial will provide you with a small peek at some steps that you can go through to evaluate your query:

• First off, you'll start with a short overview of the importance of learning SQL for jobs in data science;

• Next, you'll first learn more about how SQL query processing and execution so that you can adequately understand the importance of writing qualitative queries: more specifically, you'll see that the query is parsed, rewritten, optimized and finally evaluated;

With that in mind, you'll not only go over some query anti-patterns that beginners make when writing queries, but you'll also learn more about alternatives and solutions to those possible mistakes; You'll also learn more about the set-based versus the procedural approach to querying.

• You'll also see that these anti-patterns stem from performance concerns and that, besides the "manual" approach to improving SQL queries, you can analyze your queries also in a more

structured, in-depth way by making use of some other tools that help you to see the query plan; And,

• You'll briefly go more into time complexity and the big O notation to get an idea about the time complexity of an execution plan before you execute your query; Lastly,

• You'll briefly get some pointers on how you can tune your query further.

SQL is far from dead: it's one of the most in-demand skills that you find in job descriptions from the data science industry, whether you're applying for a data analyst, a data engineer, a data scientist or any other roles. This is confirmed by 70% of the respondents of the 2016 O'Reilly Data Science Salary Survey, who indicate that they use SQL in their professional context. What's more, in this survey, SQL stands out way above the R (57%) and Python (54%) programming languages.

You get the picture: SQL is a must-have skill when you're working towards getting a job in the data science industry.

Not bad for a language that was developed in the early 1970s, right?

But why exactly is it that it is so frequently used? And why isn't it dead even though it has been around for such a long time?

There are several reasons: one of the first reasons would be that companies mostly store data in Relational Database Management Systems (RDBMS) or in Relational Data Stream Management Systems (RDSMS) and you need SQL to access that data. SQL is the lingua franca of data: it gives you the ability to interact with almost any database or even to build your own locally!

As if this wasn't enough yet, keep in mind that there are quite a few SQL implementations that are incompatible between vendors and do not necessarily follow standards. Knowing the

standard SQL is thus a requirement for you to find your way around in the (data science) industry.

On top of that, it's safe to say that SQL has also been embraced by newer technologies, such as Hive, a SQL-like query language interface to query and manage large datasets, or Spark SQL, which you can use to execute SQL queries. Once again, the SQL that you find there will differ from the standard that you might have learned, but the learning curve will be considerably easier.

If you do want to make a comparison, consider it as learning linear algebra: by putting all that effort into this one subject, you know that you will be able to use it to master machine learning as well!

In short, this is why you should learn this query language:

• It's is fairly easy to learn, even for total newbies. The learning curve is quite easy and gradual, so you'll be writing queries in no time.

• It follows the "learn once, use anywhere" principle, so it's a great investment of your time!

• It's an excellent addition to programming languages; In some cases, writing a query is even preferred over writing code because it's more performant!

SQL Processing & Query Execution

To improve the performance of your SQL query, you first have to know what happens internally when you press the shortcut to run the query.

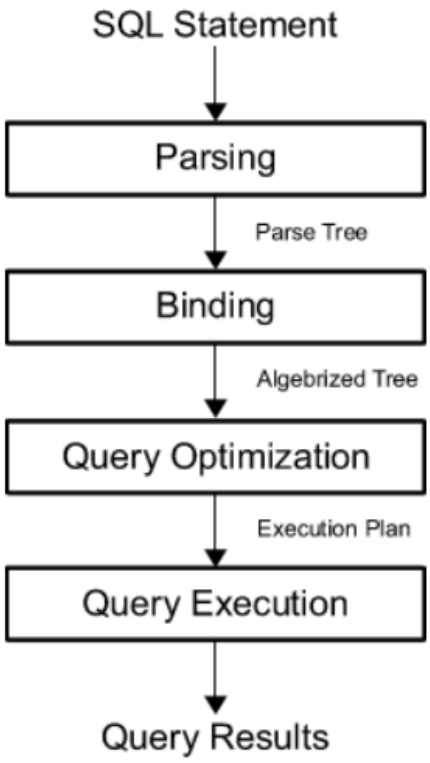
First, the query is parsed into a "parse tree"; The query is analyzed to see if it satisfies the syntactical and semantical requirements. The parser creates an internal representation of the input query. This output is then passed on to the rewrite engine.

It is then the task of the optimizer to find the optimal execution or query plan for the given query. The execution plan defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated.

To indeed find the most optimal execution plan, the optimizer enumerates all possible execution plans, determines the quality or cost of each plan, takes information about the current database state and then chooses the best one as the final execution plan. Because query optimizers can be imperfect, database users and administrators sometimes need to manually examine and tune the plans produced by the optimizer to get better performance.

Now you probably wonder what is considered to be a “good query plan”. As you already read, the quality of the cost of a plan plays a considerable role. More specifically, things such as the number of disk I/Os that are required to evaluate the plan, the plan’s CPU cost and the overall response time that can be observed by the database client and the total execution time are essential. That is where the notion of time complexity will come in. You’ll read more about this later on.

Next, the chosen query plan is executed, evaluated by the system’s execution engine and the results of your query are returned.



Writing SQL Queries

What might not have become clear from the previous section is that the Garbage In, Garbage Out (GIGO) principle naturally surfaces within the query processing and execution: the one who formulates the query also holds the keys to the performance of your SQL queries. If the optimizer gets a poorly formulated query, it will only be able to do as much… That means that there are some things that you can do when you're writing a query. As you already saw in the introduction, the responsibility is two-fold: it's not only about writing queries that live up to a certain standard, but also about gathering an idea of where performance problems might be lurking within your query.

An ideal starting point is to think of “spots” within your queries where issues might sneak in. And, in general, there are four clauses and keywords where newbies can expect performance issues to occur:

- The WHERE clause;
- Any INNER JOIN or LEFT JOIN keywords; And,
- The HAVING clause;

Granted, this approach is simple and naive, but as a beginner, these clauses and statements are excellent pointers, and it’s safe to say that when you’re just starting out, these spots are the ones where mistakes happen and, ironically enough, where they’re also hard to spot.

However, you should also realize that performance is something that needs a context to become meaningful: merely saying that these clauses and keywords are bad isn’t the way to go when you’re thinking about SQL performance. Having a WHERE or HAVING clause in your query doesn’t necessarily mean that it’s a bad query…

Take a look at the following section to learn more about anti-patterns and alternative approaches to building up your query. These tips and tricks are meant as a guide. How and if you actually need to rewrite your query depends on the amount of data, the database and the number of times you need to execute the query, among other things. It entirely depends on the goal of your query and having some prior knowledge about the database that you want to query is crucial!

1. Only Retrieve The Data

You Need The mindset of “the more data, the better” isn’t one that you should necessarily live by when you’re writing SQL queries: not only do you risk obscuring your insights by getting

more than what you actually need, but also your performance might suffer from the fact that your query pulls up too much data.

That's why it's generally a good idea to look out for the SELECT statement, the DISTINCT clause and the LIKE operator.

The SELECT Statement

A first thing that you can already check when you have written your query is whether the SELECT statement is as compact as possible. Your aim here should be to remove unnecessary columns from SELECT. This way you force yourself only to pull up data that serves your query goal. In case you have correlated subqueries that have EXISTS, you should try to use a constant in the SELECT statement of that subquery instead of selecting the value of an actual column. This is especially handy when you're checking the existence only.

Remember that a correlated subquery is a subquery that uses values from the outer query. And note that, even though NULL can work in this context as a "constant", it's very confusing!

The DISTINCT Clause

The SELECT DISTINCT statement is used to return only distinct (different) values. DISTINCT is a clause that you should definitely try to avoid if you can. As you have read in other examples, the execution time only increases if you add this clause to your query. It's therefore always a good idea to consider whether you really need this DISTINCT operation to take place to get the results that you want to accomplish.

The LIKE Operator

When you use the LIKE operator in a query, the index isn't used if the pattern starts with % or _. It will prevent the database from using an index (if it exists). Of course, from another point of view, you could also argue that this type of query potentially leaves the door open to retrieve too many records that don't necessarily satisfy your query goal.

Once again, your knowledge of the data that is stored in the database can help you to formulate a pattern that will filter correctly through all the data to find only the rows that really matter for your query.

2. Limit Your Results

When you cannot avoid filtering down on your SELECT statement, you can consider limiting your results in other ways. Here's where approaches such as the LIMIT clause and data type conversions come in.

You should always use the most efficient, that is, smallest, data types possible. There's always a risk when you provide a huge data type when a smaller one will be more sufficient.

However, when you add data type conversion to your query, you only increase the execution time.

An alternative is to avoid data type conversion as much as possible. Note also here that it's not always possible to remove or omit the data type conversion from your queries, but that you should definitely aim to be careful in including them and that when you do, you test the effect of the addition before you run the query.

3. Don’t Make Queries More Complex Than They Need To Be

The data type conversions bring you to the next point: you should not over-engineer your queries. Try to keep them simple and efficient. This might seem too simple or stupid even to be a tip, mainly because queries can get complex.

However, you’ll see in the examples that are mentioned in the next sections that you can easily start making simple queries more complex than they need to be.

The OR Operator

When you use the OR operator in your query, it’s likely that you’re not using an index.

Remember that an index is a data structure that improves the speed of the data retrieval in your database table, but it comes at a cost: there will be additional writes, and additional storage space is needed to maintain the index data structure. Indexes are used to quickly locate or look up data without having to search every row in a database every time the database table is accessed. Indexes can be created by using one or more columns in a database table.

If you don’t make use of the indexes that the database includes, your query will inevitably take longer to run. That’s why it’s best to look for alternatives to using the OR operator in your query;

The first query uses the WHERE clause to restrict the number of rows that need to be summed, whereas the second query sums up all the rows in the table and then uses HAVING to throw away the sums it calculated. In these types of cases, the alternative with the WHERE clause is obviously the better one, as you don’t waste any resources.

You see that this is not about limiting the result set, but instead about limiting the intermediate number of records within a query.

Note that the difference between these two clauses lies in the fact that the WHERE clause introduces a condition on individual rows, while the HAVING clause introduces a condition on aggregations or results of a selection where a single result, such as MIN, MAX, SUM,… has been produced from multiple rows.

You see, evaluating the quality, writing and rewriting of queries is not an easy job when you take into account that they need to be as performant as possible; Avoiding anti-patterns and considering alternatives will also be a part of responsibility when you write queries that you want to run on databases in a professional environment.

This list was just a small overview of some anti-patterns and tips that will hopefully help beginners; If you'd like to get an insight into what more senior developers consider the most frequent anti-patterns, check out this discussion.

Set-based versus Procedural Approaches to Querying

What was implicit in the above anti-patterns is the fact that they actually boil down to the difference in set-based versus a procedural approach to building up your queries.

The procedural approach of querying is an approach that is much like programming: you tell the system what to do and how to do it.

An example of this is the redundant conditions in joins or cases where you abuse the HAVING clause, like in the above examples, in which you query the database by performing a function

and then calling another function, or you use logic that contains loops, conditions, User Defined Functions (UDFs), cursors, … to get the final result. In this approach, you'll often find yourself asking a subset of the data, then requesting another subset from the data and so on.

It's no surprise that this approach is often called "step-by-step" or "row-by-row" querying.

The other approach is the set-based approach, where you just specify what to do. Your role consists of specifying the conditions or requirements for the result set that you want to obtain from the query.

How your data is retrieved, you leave to the internal mechanisms that determine the implementation of the query: you let the database engine determine the best algorithms or processing logic to execute your query.

Since SQL is set-based, it's hardly a surprise that this approach will be quite more effective than the procedural one and it also explains why, in some cases, SQL can work faster than code. Tip the set-based approach of querying is also the one that most top employers in the data science industry will ask of you to master!

You'll often need to switch between these two types of approaches. Note that if you ever find yourself with a procedural query, you should consider rewriting or refactoring it.

From Query to Execution Plans

Knowing that anti-patterns aren't static and evolve as you grow as an SQL developer and the fact that there's a lot to consider when you're thinking about alternatives also means that avoiding query anti-patterns and rewriting queries can be quite a difficult task. Any help can come in

handy, and that's why a more structured approach to optimize your query with some tools might be the way to go. Note also that some of the anti-patterns mentioned in the last section had roots in performance concerns, such as the AND, OR and NOT operators and their lack of index usage.

Thinking about performance doesn't only require a more structured approach but also a more in-depth one.

Be however it may, this structured and in-depth approach will mostly be based on the query plan, which, as you remember, is the result of the query that's first parsed into a "parse tree" and defines precisely what algorithm is used for each operation and how the execution of operations is coordinated.

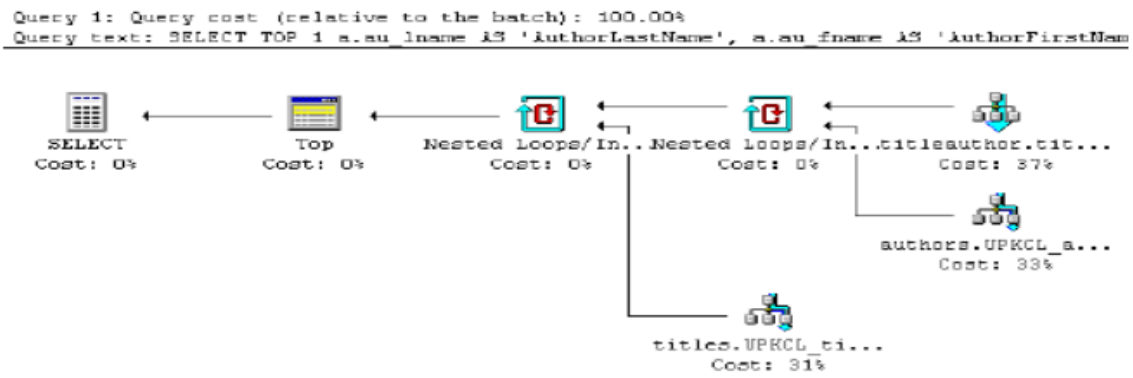
Query Optimization

As you have read in the introduction, it could be that you need to examine and tune the plans that are produced by the optimizer manually. In such cases, you will need to analyze your query again by looking at the query plan.

To get a hold of this plan, you will need to use the tools that your database management system provides to you.

Some tools that you might have at your disposal are the following:

• Some packages feature tools which will generate a graphical representation of a query plan. Take a look at this example:



- Other tools will be able to provide you with a textual description of the query plan. One example is the EXPLAIN PLAN statement in Oracle, but the name of the instruction varies according to the RDBMS that you're working with. Elsewhere, you might find EXPLAIN (MySQL, PostgreSQL) or EXPLAIN QUERY PLAN (SQLite).

Note that if you're working with PostgreSQL, you make the difference between EXPLAIN, where you just get a description that states the idea of how the planner intends to execute the query without running it, while EXPLAIN ANALYZE actually executes the query and returns to you an analysis of the expected versus actual query plans. Generally speaking, a real execution plan is one where you actually run the query, whereas an estimated execution plan works out what it would do without executing the query. Although logically equivalent, an actual execution plan is much more useful as it contains additional details and statistics about what actually happened when executing the query.

In the remainder of this section, you'll learn more about EXPLAIN and ANALYZE and how you can use these two to learn more about your query plan and the possible performance of your query.

To do this, you'll get started with some examples in which you'll work with two tables: one_million and half_million.

You can retrieve the current information of the table one_million with the help of EXPLAIN; Make sure to put it right on top of your query, and when it's run, it'll give you back the query plan:

Time Complexity & The Big O

Now that you have examined the query plan briefly, you can start digging deeper and think about the performance in more formal terms with the help of the computational complexity theory. This area in theoretical computer science that focuses on classifying computational problems according to their difficulty, among other things; These computational problems can be algorithms, but also queries.

For queries, however, you're not necessarily classifying them according to their difficulty, but rather to the time it takes to run it and get some results back. This is explicitly referred to as time complexity and to articulate or measure this type of complexity, you can use the big O notation.

With the big O notation, you express the runtime in terms of how quickly it grows relative to the input, as the input gets arbitrarily large. The big O notation excludes coefficients and lower order terms so that you can focus on the important part of your query's running time: its rate of growth. When expressed this way, dropping coefficients and lower order terms, the time complexity is said to be described asymptotically. That means that the input size goes to infinity.

In database language, the complexity measures how much longer it takes a query to run as the size of the data tables, and therefore the database, increase.

Note that the size of your database doesn't only increase as more data is stored in tables, but also the mere fact that indexes are present in the database also plays a role in the size.

Estimating Time Complexity of Your Query Plan

As you have seen before, the execution plan defines, among other things, what algorithm is used for each operation, which makes that every query execution time can be logically expressed as a function of the table size involved in the query plan, which is referred to as a complexity function. In other words, you can use the big O notation and your execution plan to estimate the query complexity and the performance.

In the following subsections, you'll get a general idea about the four types of time complexities, and you'll see some examples of how queries' time complexity can vary according to the context in which you run it.

Hint: the indexes are part of the story here!

Note, though, that there are different types of indexes, different execution plans and different implementations for various databases to consider, so that the time complexities listed below are very general and can vary according to your specific setting.

O(1): Constant Time

An algorithm is said to run in constant time if it requires the same amount of time regardless of the input size. When you're talking about a query, it will run in constant time if it requires the same amount of time irrespective of the table size.

Creating a View

We will start by generating a simple chart. In this section, we will get to know our data and will begin to ask questions about the data to gain insights. There are some important terms that we will encounter in this section.

Dimension

Measures

Aggregation

Dimensions are qualitative data, such as a name or date. By default, Tableau automatically classifies data that contains qualitative or categorical information as a dimension, for example, any field with text or date values. These fields generally appear as column headers for rows of data, such as Customer Name or Order Date, and also define the level of granularity that shows in the view. Measures are quantitative numerical data.

By default, Tableau treats any field containing this kind of data as a measure, for example, sales transactions or profit. Data that is classified as a measure can be aggregated based on a given dimension, for example, total sales (Measure) by region (Dimension).

Aggregation is the row-level data rolled up to a higher category, such as the sum of sales or total profit. Tableau automatically sorts the fields in Measures and Dimensions. However, for any anomaly, one can change it manually too.

Steps

1. Go to the worksheet. Click on the tab Sheet 1 at the bottom left of the tableau workspace.

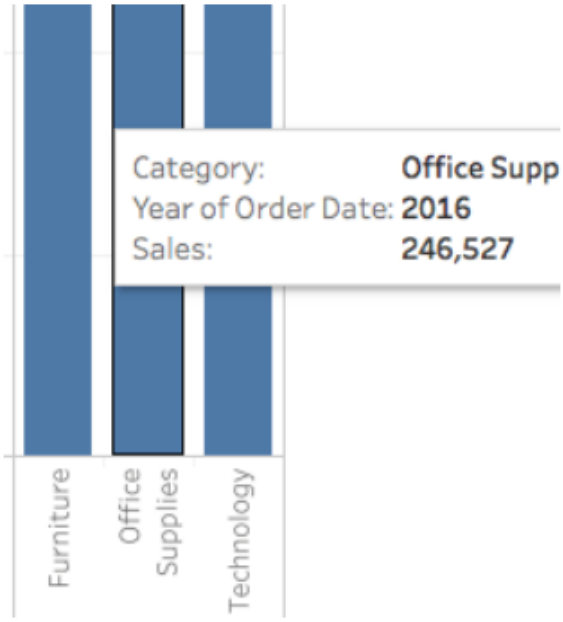
2. Once, you are in the worksheet, from Dimensions under the Data pane, drag the Order Date to the Column shelf. On dragging the Order Date to the columns shelf, a column for each year of Orders is created in the dataset. An 'Abc' indicator is visible under each column which implies that text or numerical or text data can be dragged here. On the other hand, if we pulled Sales here, a cross-tab would be created which would show the total Sales for each year.

3. Similarly, from the Measures tab, drag the Sales field onto the Rows shelf. Tableau populates a chart with sales aggregated as a sum. Total aggregated sales for each year by order date is displayed. Tableau always populates a line chart for a view that includes time-field which in this example is Order Date. Refining the View Let us delve deeper and try to find out more insights regarding which products drive more sales.

Let's start by adding the product categories to look at sales totals in a different way. Steps 1. Category is present under the Dimensions pane.

Drag it to the columns shelf and place it next to YEAR(Order Date). The Category should be placed to the right of Year. In doing so, the view immediately changes to a bar chart type from a line. The chart shows the overall Sales for every Product by year.

Learn More To view information about each data point (that is, mark) in the view, hover over one of the bars to reveal a tooltip. The tooltip displays total sales for that category. Here is the tooltip for the Office Supplies category for 2016:



2. The view above nicely shows sales by category, i.e., furniture, office supplies, and technology. We can also infer that furniture sales are growing faster than sales of office supplies except for 2016. Hence it will be wise to focus sales efforts on furniture instead of office supplies.

But furniture is a vast category and consists of many different items. How can we identify which furniture item is contributing towards maximum sales? To help us answer that question, we decide to look at products by Sub-category to see which items are the big sellers. Let's say for the Furniture category; we want to look at details about only bookcases, chairs, furnishings, and tables.

We will Double-click or drag the SubCategory dimension to the Columns shelf. The sub-category is another discrete field. It further dissects the Category and displays a bar for every sub-category broken down by category and year. However, it is a humongous amount of data to make sense of visually. In the next section, we will learn about filters, color and other ways to

make the view more comprehensible. Hands On 3.Emphasizing the Results In this section, we will try to focus on specific results. Filters and colors are ways to add more focus to the details that interest us.

Adding filters to the view Filters can be used to include or exclude values in the view. Here we try to add two simple filters to the worksheet to make it easier to look at product sales by sub-category for a specific year.

Steps In the Data pane, under Dimensions, right-click Order Date and select Show Filter.Repeat for Sub->category field also.

Filters are the type of cards and can be moved around on the worksheet by simple drag and drop Adding colors to the view Colors can be helpful in the visual identification of a pattern. Steps In the Data pane, under Measures, drag Profit to Color on the Marks card. It can be seen that Bookcases, Tables and even machine contribute to negative profit, i.e., loss. A powerful insight.

CHAPTER 4

Results and Discussion

Let's take a closer look at the filters to find out more about the unprofitable products.

Steps

1. In the view, in the Sub-Category filter card, uncheck all boxes except Bookcases, Tables, and Machines. This brings to light an interesting fact. While in some years, Bookcases and Machines were actually profitable. However, in 2016, Machines became unprofitable.

2. Select All in the Sub-Category filter card to show all the subcategories again.

3. From the Dimensions, drag Region to the Rows shelf and place it to the left of the Sum(Sales) tab. We notice that machines in the South are reporting a higher negative profit overall than in your other regions.

4. Let us now give a name to the sheet. At the bottom-left of the workspace, doubleclick Sheet 1 and type Sales by Product and Region.

5. In order to preserve the view, Tableau allows us to duplicate our worksheet so that we can continue in another sheet from where we left off.

6. In your workbook, right-click the Sales by Product and Region sheet and select Duplicate and rename the duplicated sheet to Sales-South.

7. In the new worksheet, from Dimensions, drag Region to the Filters shelf to add it as a filter in the view.

8. In the Filter Region dialogue box, clear all check boxes except South and then click OK. Now we can focus on sales and profit in the South. We find that machine sales had a negative profit in 2014 and again in 2016. We will investigate this in the next section

9. Lastly, do not forget to save the results by selecting File > Save As. Let us name our workbook as Regional Sales and Profits.

This dataset consists information about used car listed on cardekho.com. It has 9 columns each columns consists information about specific features like Car_Name gives information about car company .which Year the brand new car has been purchased.selling_price the price at which car is being sold this will be target label for further prediction of price.km_driven number of kilometre car has been driven.fuel this feature the fuel type of car (CNG , petrol,diesel etc).seller_type tells whether the seller is individual or a dealer. transmission gives information about the whether the car is automatic and manual.owner number of previous owner of the car. Present_price what is the current showroom price of the car.

Step 1 : Setting a virtual environment This should be initial step when you are building an end to end project. We need new virtual environment because each project required different set and different version of libraries so by making an individual environment for a specific project we can feed all the essential library to that environment. Follow these step to do so….

```
conda create -n carfare python=3.6
#some essential package for the environment will be installed #automatically then you will get option
[y/n] ---> y #click y
```

After that we will activate the environment by using

```
>> activate carfare
>> jupyter notebook #run jupyter notebook on newly created env
```

It might be possible that we will get issue about absence of jupyter notebook for that we have to do one more step.

```
>> pip install jupyter notebook # installing jupyter notebook on env
>> jupyter notebook
```

our environment is created, now we will do our complete project on this environment.

Step 2 :- Acquiring data set and importing all the essential library I have taken data set from here & The data set is in csv format. Now I will import all the essential library that will be needed for this project.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

Step 3 :- Data Pre Processing

```
df=pd.read_csv("car data.csv")
df.head()  #Printing first five rows
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

```
len(df["Car_Name"].unique())[out]>> 98
```

Here we have cars of 98 different companies and name of companies won't affect car's price ,price depends upon how many year it's been used ,fuel type etc.,.so I will drop the column car_name from original data frame.

Now we will check the data type of each column if the data type is numerical then we have no such issue but if datatype is categorical then we need to convert all those categorical features into numerical values.

If we will observe above output we can say that there are some features which are having object data type now in my next step i will create a cat_df and will store all the categorical feature into cat_df.

Now we will make the categorical dataframe with all the features having categorical variables, and will drop all the categorical features from original dataframe.

number of year car has been used = current year — previous year

So as we are in 2020 and car is of 2014 then number of years it's been used will be :- number of year car has been used=2020–2014=6 to print number of year car has been used we need to add a column which represent current year.

We have successfully added the current_ year now we will add number_of_years column and drop Year and Current _year column.

On an average car has been driven 36947 kilometres and max distance the car has been traveled is 5,00,000 kilometres. The car with highest ex-showroom selling price present in data set is 92.6 lakh. Maximum number of years car has been used and then come for sell is 17 years.maximum number of owner that has used a single car is 3.Maximum selling price for used car is 35 lakh rupees. This is how we make conclusion with statistical description of dataset.

Step 4 :- Data Visualization

This is most important step of data science life cycle, here we understand the behavior of data and try to make certain meaningful insight out of it.

let's understand it by doing… More number of Years you will use your car lesser the amount you will get. lesser the car would be driven higher will be the cost as we see the graph at max distance i e:- 500000 kilometres the car's cost is near to Zero or we can say nobody is willing to pay any amount to those cars. plotting pair plot We cannot visualize multi dimensional scatter plot hence by using pair plot we can visualize each and every dimension of (Dimension with numerical variable)multidimensional data precisely.

As we see there are very less overlapping in dataset is seen so we cannot use knn ,linear regression,svm and because of the dynamic nature of dataset we even cannot use decision tree so we will go with random forest and xgboost. Uni variate analysis :- when analysis involve single variable most predominantly it is used to find the pattern in dataset.

Most number of car has been sold within a price range of 1–10 lakh and for a price range of 25 - 35 lakh there are negligible amount of customer

Demand for those car that has been traveled less distance are in more demand especially if car has traveled distance within a range of 0–5000 kilometre people are more attracted towards them.

C.D.F Plot

It defines how many percentage of variable has value less than and equal to corresponding xaxis. lets' take above example how many percentage of vehicle has selling price less than 15 lakh then we can find this kind of answer by using C.D.F.

As we can see 94.7% of cars that are on cardekho has price ≤15 lakh . So one thing is clear that if we want to purchase used car with a price range of 20–25 we won't prefer to go to cardekho.com because there we won't get so many options.

multivariate Analysis :- When we are analyzing two and more variables.

If we see above graph we can understand that for those vehicle whose original price lie within range of 0–20 lakh they are getting approximately 50% of their money when they sell their car after using certain period of time.

How Machine Learning and Artificial Intelligence Changing the Face of eCommerce? | Data Driven…

The eCommerce development company, nowadays, integrating advancement to take customer experience to the next level…

www.datadriveninvestor.com

Step 5:- Feature Engineering

As we have already made the cat_df with all the categorical features so we will drop all those feature consisting categorical variable from original dataframe and at the end after using feature engineering in cat_df we will concatenate cat_df with original dataframe and by doing this we will be able to convert all the variables into numerical form.

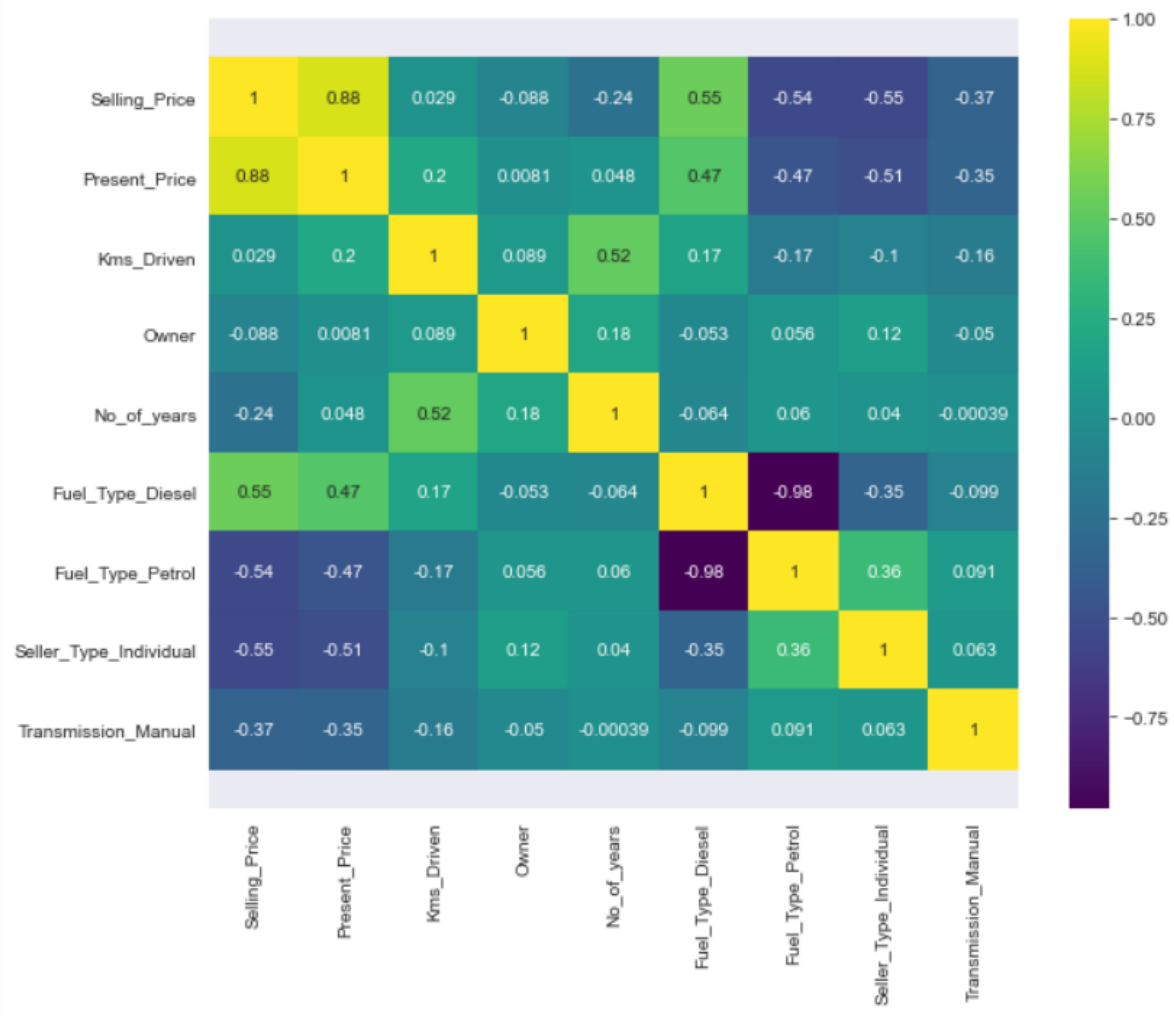
Now we will do feature engineering on cat_df to convert the categorical variable into numerical variable.But before that we will check how many unique categorical variable each feature consists.

As Fuel_Type,Seller_Type,Transmission all are nominal feature and having small number of categorical variable we will

	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	0	1	0	1
1	1	0	0	1
2	0	1	0	1
3	0	1	0	1
4	1	0	0	1

Now we have converted all the features into numerical variable.Here we will check correlation between feature but for this dataset we won't do feature selection,

Because Feature selection is used when we have large features in a dataset but in this dataset we only check how features in dataset are correlated with each other.



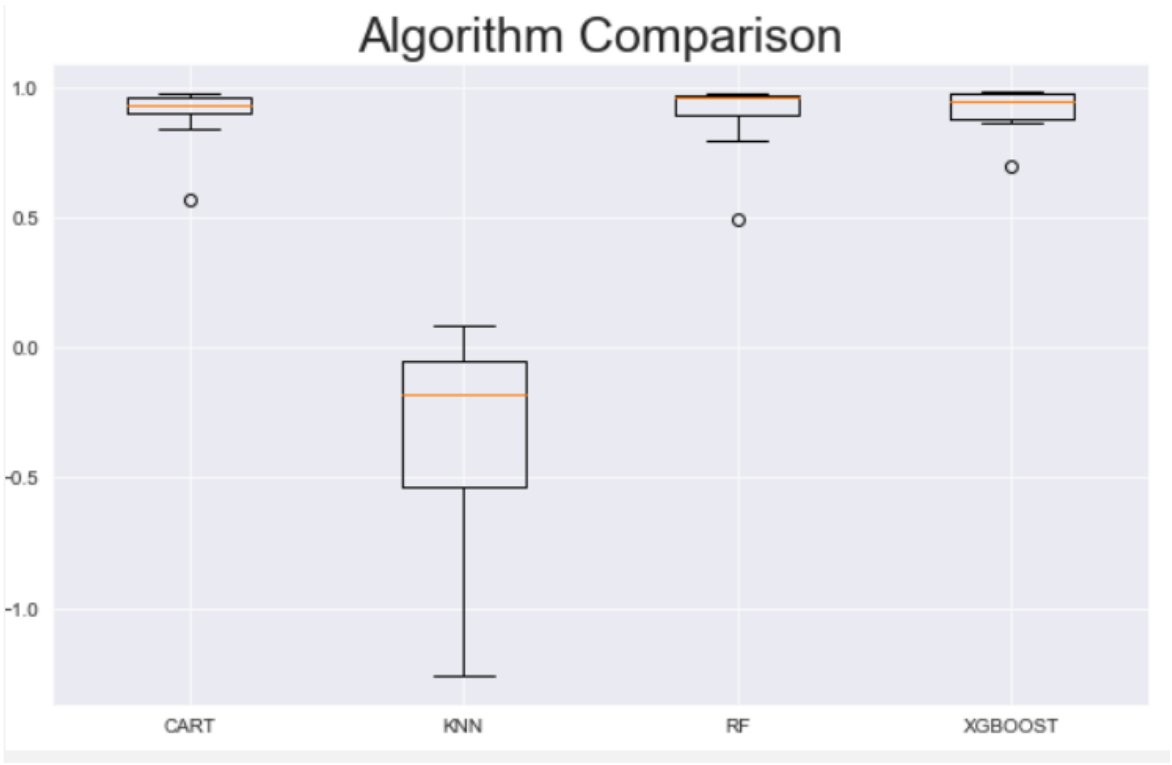
Step 6 : Pre Modeling Steps

Splitting dataset into dependent and independent variable.

```
X=df.iloc[:,1:]
Y=df.iloc[:,0]
print(X.shape)
print(Y.shape)[out]          >>          (301,          11)
      (301,)
```

Choosing best fit model for our dataset :

checking which model will be best for our dataset as from pair plot it is clear that we have to take model which do prediction on non linear and combination of categorical and numerical data that are decision tree,random forest and xgboost then we will check which model will have high accuracy based on that we select the model. We can choose our best fit model using cross validation score.



As we see above accuracy score result we can say that XGboost gives better accuracy with very low standard deviation hence we should go with XGboost.

Feature Importance :- checking which all features are important for output features out of all the given features.

Transmission_manual,seller_Type_individual,Fuel_Type_petrol,Fuel_Type_Diesel,No_of_Years make most impact to output prediction.

Saving a model in serialized format : Save model in serialized format and when we need to do prediction we will just load the pickle file and make prediction using that serialized file we don't need to again make a new model for prediction on new test data.

Working of Random Forest Algorithm

Before understanding the working of the random forest we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

Ensemble uses two types of methods:

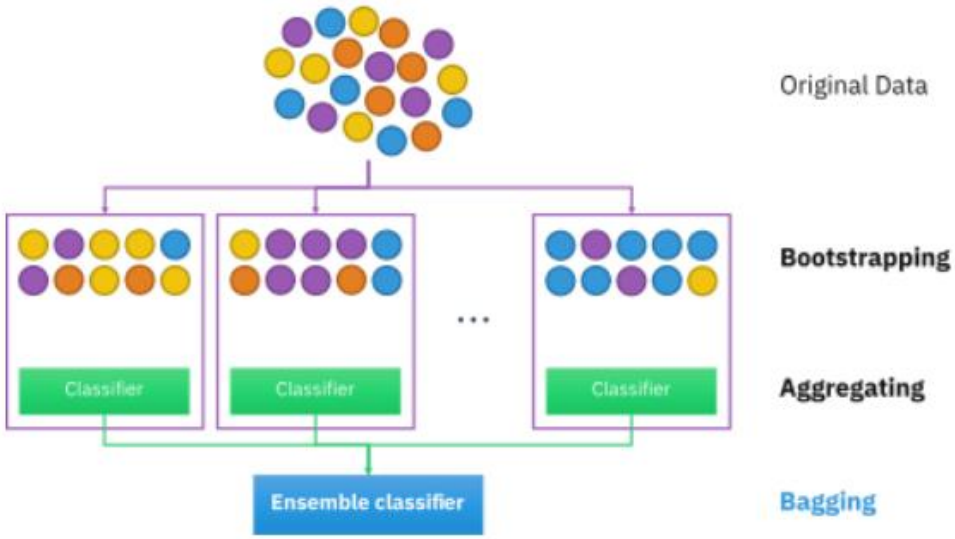
1. Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.
2. Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST

As mentioned earlier, Random forest works on the Bagging principle. Now let's dive in and understand bagging in detail.

Bagging

Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement

known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.



Now let's look at an example by breaking it down with the help of the following figure. Here the bootstrap sample is taken from actual data (Bootstrap sample 01, Bootstrap sample 02, and Bootstrap sample 03) with a replacement which means there is a high possibility that each sample won't contain unique data. Now the model (Model 01, Model 02, and Model 03) obtained from this bootstrap sample is trained independently. Each model generates results as shown.

Now Happy emoji is having a majority when compared to sad emoji. Thus based on majority voting final output is obtained as Happy emoji.

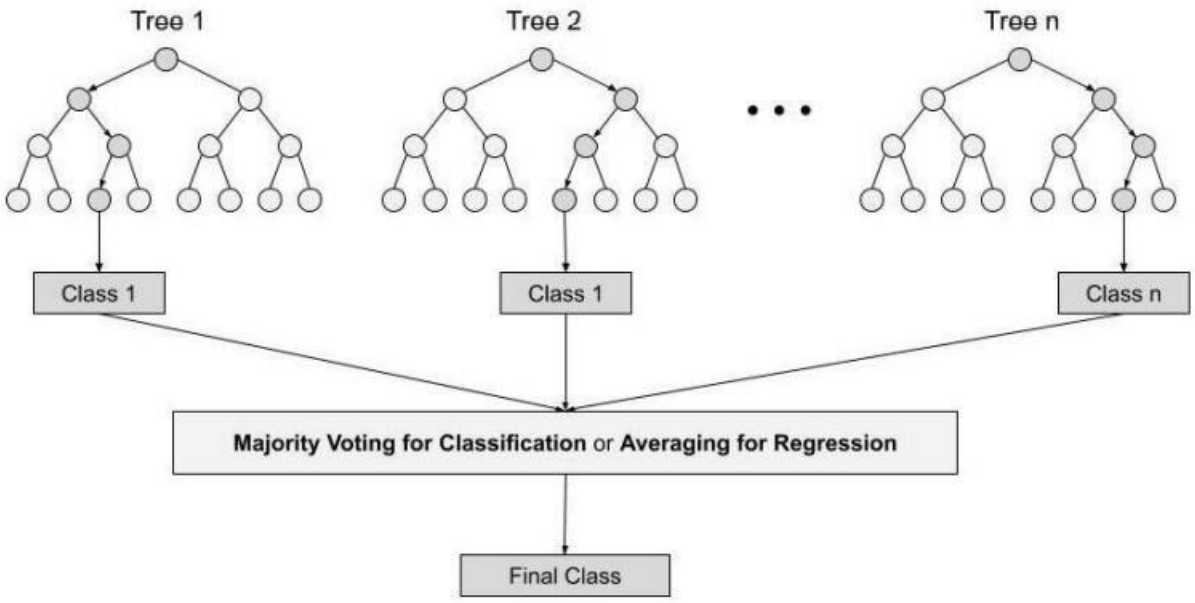
Steps involved in random forest algorithm:

Step 1: In Random forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.



For example: consider the fruit basket as the data as shown in the figure below. Now n number of samples are taken from the fruit basket and an individual decision tree is constructed for each sample. Each decision tree will generate an output as shown in the figure. The final output is considered based on majority voting. In the below figure you can see that the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.

Important Features of Random Forest

1. Diversity- Not all attributes/variables/features are considered while making an individual tree, each tree is different.

2. Immune to the curse of dimensionality- Since each tree does not consider all the features, the feature space is reduced.

3. Parallelization-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

4. Train-Test split- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

5. Stability- Stability arises because the result is based on majority voting/ averaging. Difference Between Decision Tree & Random Forest

Random forest is a collection of decision trees; still, there are a lot of differences in their behavior.

Decision trees	Random Forest
1. Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	1. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2. A single decision tree is faster in computation.	2. It is comparatively slower.
3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	3. Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

Thus random forests are much more successful than decision trees only if the trees are diverse and acceptable.

Important Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

Following hyperparameters increases the predictive power:

1. n_estimators– number of trees the algorithm builds before averaging the predictions.
2. max_features– maximum number of features random forest considers splitting a node.
3. mini_sample_leaf– determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. n_jobs– it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.
2. random_state– controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
3. oob_score – OOB means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.

In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv('car data.csv')
```

In [3]:
```
df.shape
```

Out[3]:
```
(301, 9)
```

In [4]:
```
print(df['Seller_Type'].unique())
print(df['Fuel_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

```
['Dealer' 'Individual']
['Petrol' 'Diesel' 'CNG']
['Manual' 'Automatic']
[0 1 3]
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [6]:
```
df.describe()
```

Out[6]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [18]:
```
final_dataset=final_dataset.drop(['Current Year'],axis=1)
```

In [19]:
```
final_dataset.head()
```

Out[19]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	6	0	1	0	1
1	4.75	9.54	43000	0	7	1	0	0	1
2	7.25	9.85	6900	0	3	0	1	0	1
3	2.85	4.15	5200	0	9	0	1	0	1
4	4.60	6.87	42450	0	6	1	0	0	1

In [20]:
```
final_dataset.corr()
```

Out[20]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_M
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.550724	-0.3
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.512030	-0.3
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.101419	-0.1
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.124269	-0.0
no_year	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.039896	-0.0
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.350467	-0.0
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.358321	0.0
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321	1.000000	0.0
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013	0.063240	1.0

Out[19]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	6	0	1	0	1
1	4.75	9.54	43000	0	7	1	0	0	1
2	7.25	9.85	6900	0	3	0	1	0	1
3	2.85	4.15	5200	0	9	0	1	0	1
4	4.60	6.87	42450	0	6	1	0	0	1

In [20]:
```
final_dataset.corr()
```

Out[20]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_M
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.550724	-0.3
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.512030	-0.3
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.101419	-0.1
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.124269	-0.0
no_year	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.039896	-0.0
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.350467	-0.0
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.358321	0.0
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321	1.000000	0.0
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013	0.063240	1.0

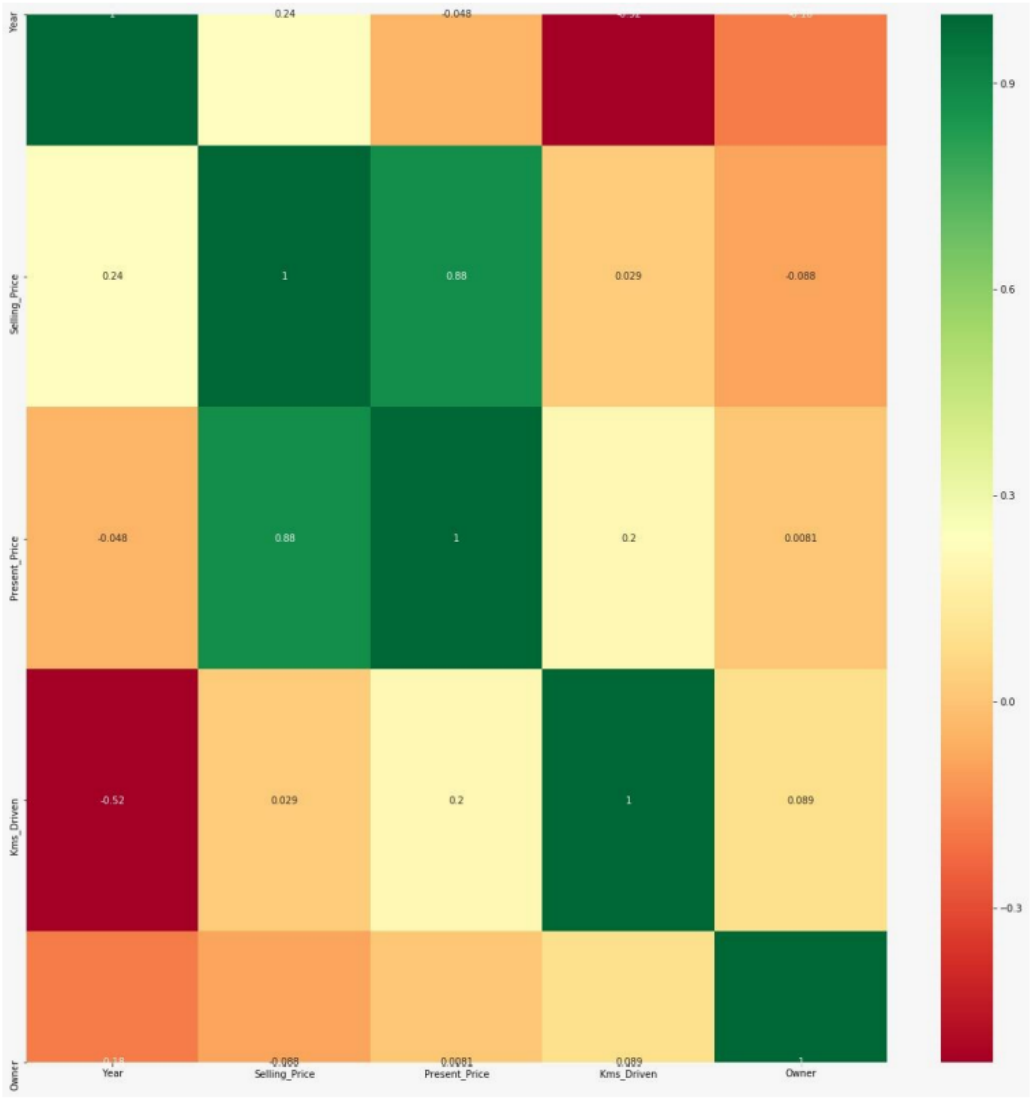
In [21]:
```
import seaborn as sns
```

In [22]:
```
sns.pairplot(final_dataset)
```

Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x173ae18ae08>
```

Popular features of used cars

When buying a used car, people pay serious attention to the odometer value on the car. We can see that odometer changes the price of a car significantly . On the other hand, this does not mean that only low odometer cars are sold. Depending on the price, high odometer cars also have buyers.


Year
Selling_Price
Present_Price
Kms_Driven
Owner
0.24
-0.048
0.24
1
0.88
0.029
-0.088
-0.048
0.88
1
0.2
0.0081
-0.52
0.029
0.2
1
0.089
0.9
0.6
0.3
0.0
−0.3

```
y.head()
```

```
Out[31]: 0    3.35
         1    4.75
         2    7.25
         3    2.85
         4    4.60
         Name: Selling_Price, dtype: float64
```

```
In [ ]:
```

```
In [32]: from sklearn.ensemble import ExtraTreesRegressor
         import matplotlib.pyplot as plt
         model = ExtraTreesRegressor()
         model.fit(X,y)
```

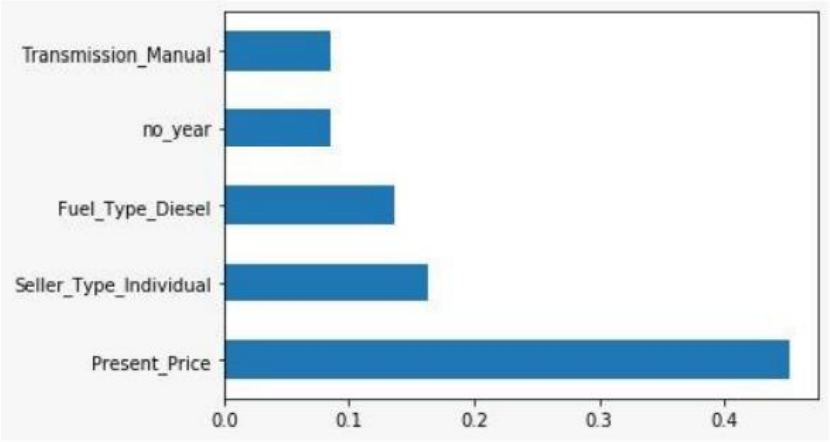
```
C:\Users\AKASH\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators wil
l change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
Out[32]: ExtraTreesRegressor(bootstrap=False, criterion='mse', max_depth=None,
                             max_features='auto', max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                             oob_score=False, random_state=None, verbose=0,
                             warm_start=False)
```

```
In [33]: print(model.feature_importances_)
```

```
[4.51451164e-01 4.11396450e-02 2.50443844e-04 8.57866923e-02
 1.35966360e-01 3.83675083e-02 1.62431757e-01 8.46064289e-02]
```

```
In [34]: feat_importances = pd.Series(model.feature_importances_, index=X.columns)
         feat_importances.nlargest(5).plot(kind='barh')
```



Machine Learning Model

This section used applied machine learning models as a framework for the data analysis. The data set is a supervised data which refers to fitting a model of dependent variables to the independent variables, with the goal of accurately predicting the dependent variable for future observations or understanding the relationship between the variables.

Pre-processing the data

Label Encoding. In the dataset, there are 13 predictors. 2 of them are numerical variables while rest of them are categorical. In order to apply machine learning models, we need numeric representation of the features. Therefore, all non-numeric features were transformed into numerical form.

Train the data. In this process, 20% of the data was split for the test data and 80% of the data was taken as train data.

Scaling the Data. While exploring the data in the previous sections, it was seen that the data is not normally distributed. Without scaling, the machine learning models will try to disregard coefficients of features that has low values because their impact will be so small compared to the big value features.

While scaling, it's also important to scale with correct method because inappropriate scaling causes inappropriate target quantification and inappropriate measures of performance (Hurwitz, E., & Marwala, 2012). Min-max scaler is appropriate especially when the data is not normal distribution and want outliers to have reduced influence. Besides, both ridge and lasso get influenced by magnitude of the features to perform regularization. Therefore, Min-Max Scaler was used on the dataset.

Random Forest

Random forest is a set of multiple decision trees. Deep decision trees may suffer from overfitting, but random forest prevents overfitting by creating trees on random subsets. That's why, it's a good model to in the analysis.

```
In [45]: rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.7s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.7s remaining:    0.0s
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.5s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.5s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   1.0s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
```

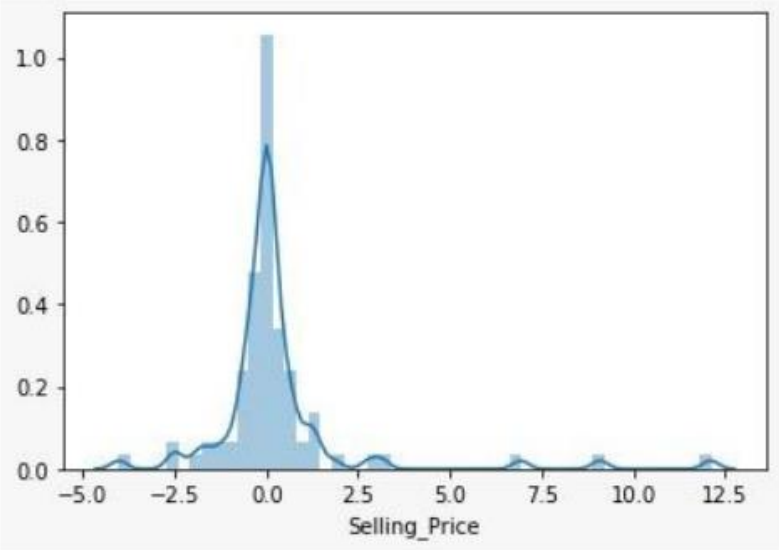
```
In [46]: rf_random.best_params_
```

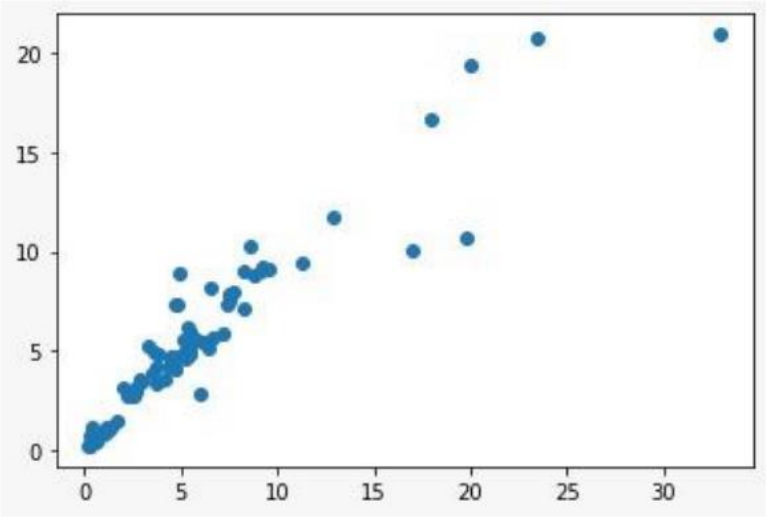
```
Out[46]: {'n_estimators': 1000,
          'min_samples_split': 2,
          'min_samples_leaf': 1,
          'max_features': 'sqrt',
          'max_depth': 25}
```

```
In [47]: rf_random.best_score_
```

```
Out[47]: -3.972709884205608
```

```
In [48]: predictions=rf_random.predict(X_test)
```





In [51]:
```
from sklearn import metrics
```

In [52]:
```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 0.8818702197802184
MSE: 3.944592801172526
RMSE: 1.9860998970778196
```

In [53]:
```
import pickle
# open a file, where you ant to store the data
file = open('random_forest_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file)
```

In [ ]:

CHAPTER 5

Conclusion

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. By performing different models, it was aimed to get different perspectives and eventually compared their performance. With this study, it purpose was to predict prices of used cars by using a dataset that has 13 predictors and 380962 observations. With the help of the data visualizations and exploratory data analysis, the dataset was uncovered and features were explored deeply. The relation between features were examined. At the last stage, predictive models were applied to predict price of cars.

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.

References

1. Agencija za statistiku BiH. (n.d.), retrieved from: http://www.bhas.ba . [accessed July 18, 2018.]
2. Listiani, M. (2009). Support vector regression analysis for price prediction in a car leasing application (Doctoral dissertation, Master thesis, TU Hamburg-Harburg).
3. Du, J., Xie, L., & Schroeder, S. (2009). Practice Prize Paper—PIN Optimal Distribution of Auction Vehicles System: Applying Price Forecasting, Elasticity Estimation, and Genetic Algorithms to Used-Vehicle Distribution. Marketing Science, 28(4), 637-644.
4. Ho, T. K. (1995, August). Random decision forests. In Document analysis and recognition, 1995., proceedings of the third international conference on (Vol. 1, pp. 278-282). IEEE
5. Auto pijaca BiH. (n.d.), Retrieved from: https://www.autopijaca.ba. [accessed August 10, 2018].
6. Gongqi, S., Yansong, W., & Qiang, Z. (2011, January). New Model for Residual Value Prediction of the Used Car Based on BP Neural Network and Nonlinear Curve Fit. In Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on (Vol. 2, pp. 682-685). IEEE.
7. Pudaruth, S. (2014). Predicting the price of used cars using machine learning techniques. Int. J. Inf. Comput. Technol, 4(7), 753-764.
8. Noor, K., & Jan, S. (2017). Vehicle Price Prediction System using Machine Learning Techniques. International Journal of Computer Applications, 167(9), 27-31.
9. Auto pijaca BiH. (n.d.), Retrieved from: https://www.autopijaca.ba. [accessed August 10, 2018]

10. Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (n.d.), Retrieved from: https://www.cs.waikato.ac.nz/ml/weka/. [August 04, 2018].
11. Ho, T. K. (1995, August). Random decision forests. In Document analysis and recognition, 1995., proceedings of the third international conference on (Vol. 1, pp. 278-282). IEEE.
12. .Russell, S. (2015). Artificial Intelligence: A Modern Approach (3rd edition). PE.
13. Ben-Hur, A., Horn, D., Siegelmann, H. T., & Vapnik, V. (2001). Support vector clustering. Journal of machine learning research, 2(Dec), 125-137
14. Aizerman, M. A. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control, 25, 821- 837