Data Science (Predicting type of car to Purchase)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

submitted by
GUNTAKA DEEPTHI (221710316012)

under the guidance of

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT(GITAM)

(Declared as Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS MARCH-2020

DECLARATION

I submit this industrial training work entitled “**Predicting type of car to Purchase**” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “Bachelor of Technology” in “Electronics and Communication Engineering”. I declare that it was carried out independently by me under the guidance of ___________, ____________, GITAM (Deemed To Be University), Hyderabad, India. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree

Place: HYDERABAD **GUNTAKA DEEPTHI**

Date: 221710316012

GITAM CERTIFICATE

INTERNSHIP CERTIFICATE

ACKNOWLEDGEMENT

Our project would not have been successful without the help of several people. we would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are extremely thankful to our honorable Pro-Vice Chancellor, Prof. N. Siva Prasad for providing necessary infrastructure and resources for the accomplishment of our project.

We are highly indebted to Prof. N. Seetha Ramaiah, Principal, School of Technology, for his support during the tenure of the project.

We are very much obliged to our beloved Prof. S. Phani Kumar, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in completion of this project.

We hereby wish to express our deep sense of gratitude to ____________, ___________, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by him for the success of the Internship Project.

We are also thankful to all the staff members of the Computer Science and Engineering department who have cooperated in making our Internship Project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our internship Project.

Sincerely

GUNTAKA DEEPTHI

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on car Evaluation Dataset. My perception of understanding the given data set has been in the view of undertaking a client's requirement of overcoming the stagnant point of sales of the products being manufactured by client.

To get a better understanding and work on a strategical approach for solution of the client, I have adapted the view point of looking at the dataset features like buying and maintenance cost because these features are the most important features for any kind of customer.

1. INFORMATION ABOUT DATA SCIENCE:

1.1 What is Data Science ………………………………………………. 10

1.2 Need of Data Science ………………………………………………. 10

1.3 Uses of Data Science ……………………………………………….. 10

2. INFORMATION ABOUT MACHINE LEARNING :

2.1 Introduction ……………………………………………………….. 11

2.2 Importance of Machine Learning ……………………………………. 11

2.3 Uses of Machine Learning …………………………………………… 12

2.4 Types of Machine Learning ………………………………………….. 12

2.4.1 Supervised Learning

2.4.2 Unsupervised Learning

2.4.3 Semi Supervised Learning

2.5 Relation between Data Mining, Machine Learning and Deep Learning . 14

3. INFORMATION ABOUT PYTHON :

3.1 Introduction………………………………………………………… 15

3.2 History of Python ………………………………………………….. 15

3.3 Features ……………………………………………………………. 15

3.4 Setup of Python…………………………………………………….. 16

3.4.1 Installation(Python)

3.4.2 Installation(Anaconda)

3.5 Variable Types ……………………………………………………..17

3.5.1 Numbers

3.5.2 Strings

3.5.3 Lists

3.5.4 tuples

3.5.5 Dictionaries

3.6 Functions ………………………………………………………… 20

3.6.1 Defining a Function

3.6.2 Calling a Function

3.7 OOPs Concepts ………………………………………………… 21

3.7.1 Class

3.7.2 __init__ method in class

4. Predicting type of car to Purchase

4.1 Project Requirements ……………………………………………22

4.1.1 Packages used

4.1.2 Versions of the packages

4.1.3 Algorithms used

4.2 Problem Statement ……………………………………………..22

4.3 Dataset Description …………………………………………….22

4.4 Objective of the Case Study …………………………………….23

5. DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

5.1 Preprocessing of Data…………………………………………..23

5.1.1 Getting the Dataset

5.1.2 Importing Libraries

5.1.3 Importing the Dataset

5.1.4 Handing Missing values

5.1.5 Categorical Data

5.2 Generating Plots………………………………………… 26

5.2.1 - Visualize the data between all the Features

5.2.2 -Visualize the data between Target and the Features

5.3 Data Cleaning …………………………………………… 34

5.4 Encoding Categorical Data ……………………………… 34

6. FEATURE SELECTION:

6.1 Select relevant features for the analysis …………………………. 35

6.2 Train-Test-Split …………………………………………………...35

7. MODEL BUILDING AND EVALUATION:

7.1 Decision Tree …………………………………………………… 36

7.1.1 Recursive Binary Splitting

7.1.2 Cost of Split

7.1.3 Pruning

7.1.4 Train the Model

7.1.5 Metrics

7.2 Random Forest Classifier ……………………………………….. 41

7.2.1 Ensemble algorithm

7.2.2 Model Creation

7.2.3 Model Fitting

7.2.4 Metrics

8. Conclusion ……………………………………………. 44

1. Information About Data Science

1.1 What is Data Science

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Data science is related to data mining, deep learning and big data.

1.2 Need of Data Science

Companies need to use data to run and grow their everyday business. The fundamental goal of data science is to help companies make quicker and better decisions, which can take them to the top of their market, or at least – especially in the toughest red oceans – be a matter of long-term survival

Industries require data scientists to assist them in making a smarter decision. To predict the information everyone requires data scientists. Big Data and Data Science hold the key to the future. Data Science is important for better marketing.

1.3 Uses of Data Science

1. Internet Search
2. Digital Advertisements(Targeted Advertising and re-targeting)
3. Website Recommender Systems
4. Advanced Image Recognition
5. Speech Recognition
6. Gaming
7. Price Comparison Websites
8. Airline Route Planning
9. Fraud and Risk Detection
10. Delivery Logistics

Apart from the applications mentioned above, data science is also used in Marketing,Finance, Human Resources, Health Care, Government,Augmented Reality,Robots, Self Driving Cars

2. Information About Machine Learning

2.1 Introduction:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

2.2 Importance of Machine Learning:

Consider some of the instances where machine learning is applied: the self-driving Google car, cybersex fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works.

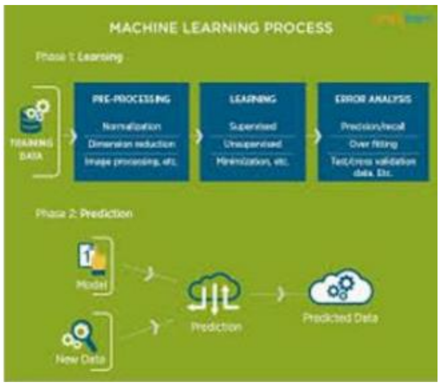


Fig 2.2.1 : The Process Flow

2.3 Uses of Machine Learning

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

2.4 Types of Machine Learning Algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

2.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labeled training data set – that is, a data set that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multi class classification.

2.4.2 Unsupervised Learning :

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

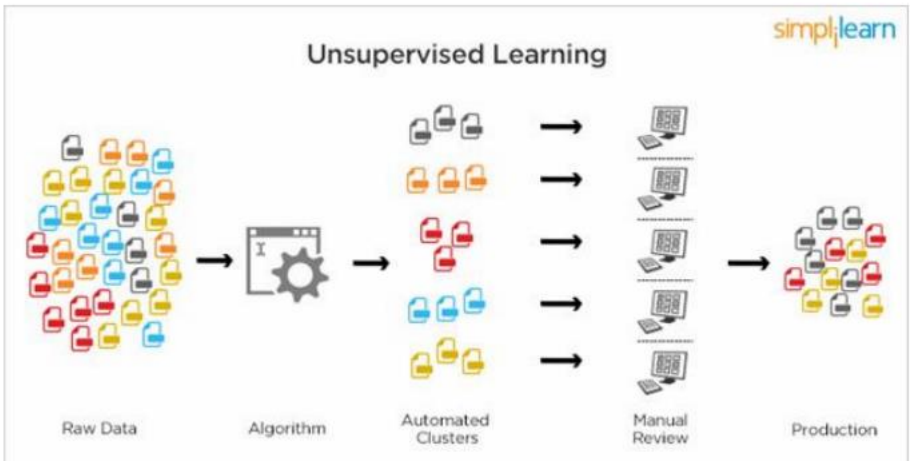


Fig 2.4.2.1: Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

2.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

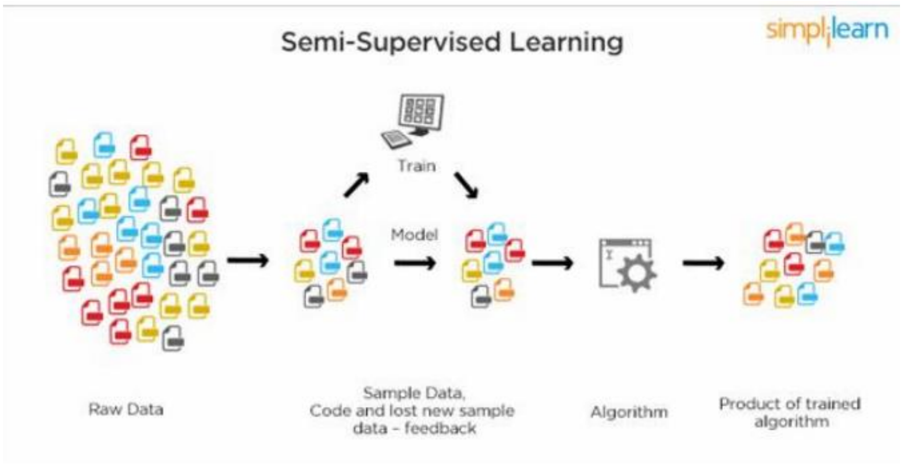


Fig 2.4.3.1: Semi Supervised Learning

2.5 Relation between Data Mining, Machine Learning and Deep Learning

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

3. INFORMATION ABOUT PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 Introduction

● Python is a high-level, interpreted, interactive and object-oriented scripting language.

● Python is a general purpose programming language that is often applied in scripting roles.

● Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

● Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.

● Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

3.2 History of Python

● Python was developed by GUIDO VAN ROSSUM in early 1990's

● Its latest version is 3.7 , it is generally called as python3

3.3 Features of Python

● Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.

● Easy-to-read: Python code is more clearly defined and visible to the eyes.

● Easy-to-maintain: Python's source code is fairly easy-to-maintaining.

● A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

● Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

● Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

● Databases: Python provides interfaces to all major commercial databases.

● GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.4 Python Setup

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

3.4.1 Installation(using python IDLE):

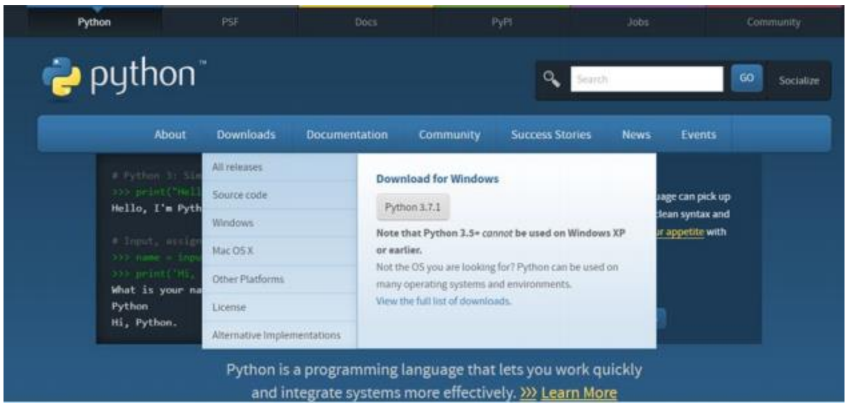
● Installing python is generally easy, and nowadays many Linux and Mac OS

distributions include a recent python.

● Download python from www.python.org

● When the download is completed, double click the file and follow the instructions to install it.

● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



```
Hello, I'm Pyth
What is your na
Python
Hi, Python.
```

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Fig 3.4.1.1: Python download

3.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.
- In WINDOWS:
- In windows
  - Step 1: Open Anaconda.com/downloads in web browser.
  - Step 2: Download python 3.4 version for

(32-bitgraphic installer/64 -bit graphic installer)

  - Step 3: select installation type(all users)
  - Step 4: Select path

(i.e. add anaconda to path & register anaconda as default python 3.4)

next click install and next click finish.

  - Step 5: Open jupyter notebook (it opens in default browser)

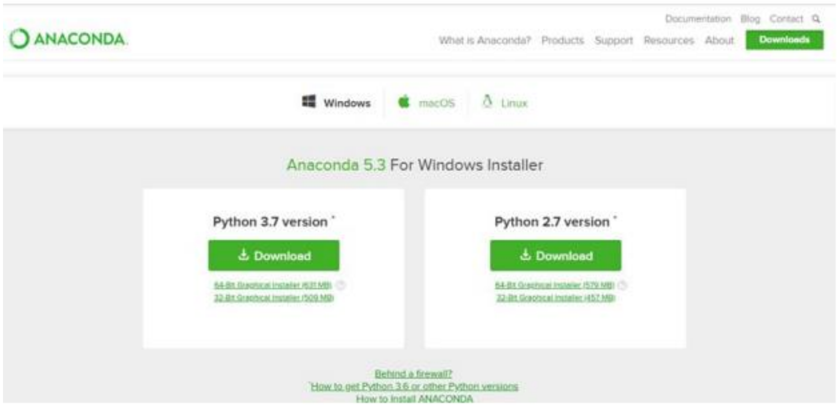

ANACONDA.
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version *
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
*How to get Python 3.6 or other Python versions
How to Install ANACONDA

Fig 3.4.2.1: Anaconda download

3.5 Python Variable Types

- Variables are nothing but reserved memory locations to store values.This means that when you create a variable you reserve some space in memory.

● Variables are nothing but reserved memory locations to store values.

● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.5.1 Python Numbers :

● Number data types store numeric values. Number objects are created when you assign a value to them.

● Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

3.5.2 Python Strings:

● Strings in Python are identified as a contiguous set of characters represented in the quotation marks.

● Python allows for either pairs of single or double quotes.

● Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

● The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

3.5.3 Python Lists:

● Lists are the most versatile of Python's compound data types.

● A list contains items separated by commas and enclosed within square brackets ([]).

● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.5.4 Python Tuples:

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

3.5.5 Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.6 Python Functions

3.6.1 Defining a Function :

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()). Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.6.2 Calling a Function :

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.7 Oops Concepts

3.7.1 Class :

● Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

● Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

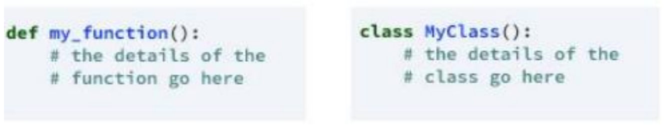
● Data member: A class variable or instance variable that holds data associated with a class and its objects.

● Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

● Defining a Class:

o We define a class in a very similar way how we define a function.

o Just like a function ,we use parentheses and a colon after the class name (i.e.():) when we define a class. Similarly, the body of our class is indented like a functions body is.



```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Fig 3.6.2.1: Defining a Class

3.7.2 __init__ method in Class:

● The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores : __init__().

4. Project Name : Classification of Cars

4.1 Project Requirements

4.1.1 Packages used.

- ✓ Pandas
- ✓ Matplotlib
- ✓ Numpy
- ✓ Seaborn
- ✓ Scikit-learn

4.1.2 Versions of the packages.

- ✓ Pandas : 0.25.1
- ✓ Matplotlib : 3.1.1
- ✓ Numpy : 1.16.5
- ✓ Seaborn:0.10.1
- ✓ Scikit-learn:0.23.1

4.1.3 Algorithms used.

- ✓ Decision Tree Algorithm
- ✓ Ensemble Algorithm: Random forest

4.2 Problem Statement

To predict the type of car based on the user 's requirement is called classification of cars.

4.3 Data set Description

- "Buying" : buying price
- "maint": price of the maintenance
- "doors": number of doors
- "persons": number of persons

- "lug_boot": size of luggage boot
- "safety": estimated safety of the car
- "value": overall price

Note: Source (Kaggle)

4.4 Objective of the Case Study

To get a better understanding and chalking out a plan of action for the client's solution, we have adapted the viewpoint of looking at a car based on previous users' requirements.

Based on the type of buyers and the maintenance cost, number of doors, number of people who can fit safely into the car, and the luggage space, safety perspective a value is to be predicted.

5. Data Preprocessing/Feature Engineering and EDA

5.1 Preprocessing of The Data

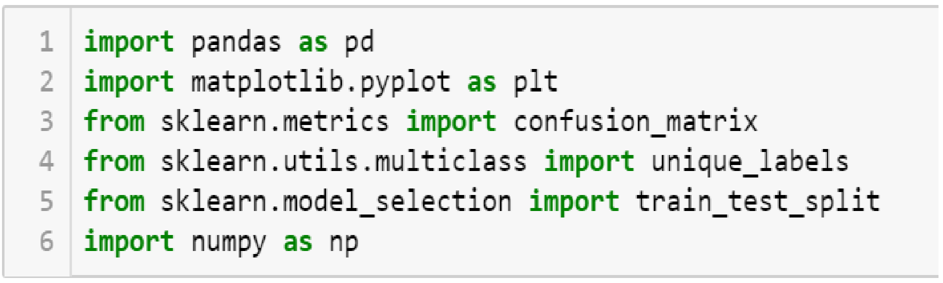
Prepossessing of the data actually involves the following steps:

5.1.1 Getting the Dataset

We can get the data set from the database or we can get the data from client.

5.1.2 Importing the Libraries

we have to import the libraries as per the requirement of the algorithm.

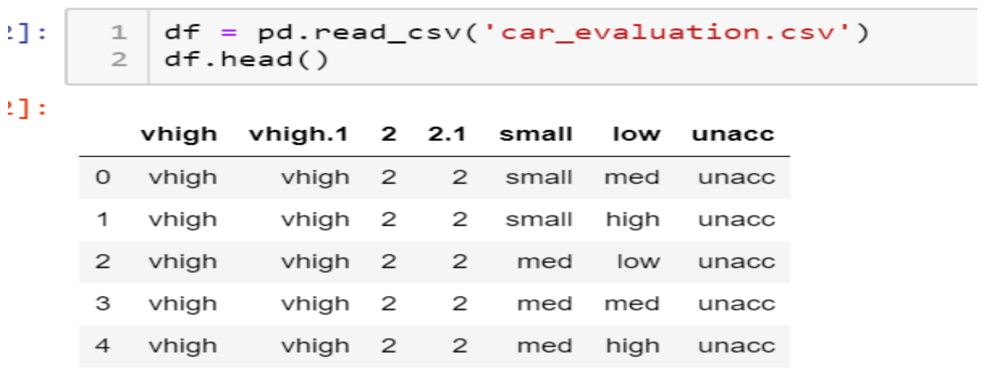


```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels
from sklearn.model_selection import train_test_split
import numpy as np
```

Fig 5.1.2.1: Importing Libraries

5.1.3 Importing The Data-Set

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire data set from a comma separated values file and we can assign it to a Data Frame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the data frame. Any missing value or Nan value have to be cleaned.



```
]:  df = pd.read_csv('car_evaluation.csv')
    df.head()
```

```
]:
       vhigh   vhigh.1  2  2.1  small   low   unacc
    0  vhigh   vhigh    2    2  small   med   unacc
    1  vhigh   vhigh    2    2  small   high  unacc
    2  vhigh   vhigh    2    2    med   low   unacc
    3  vhigh   vhigh    2    2    med   med   unacc
    4  vhigh   vhigh    2    2    med   high  unacc
```

Fig 5.1.3.1: Reading the data set

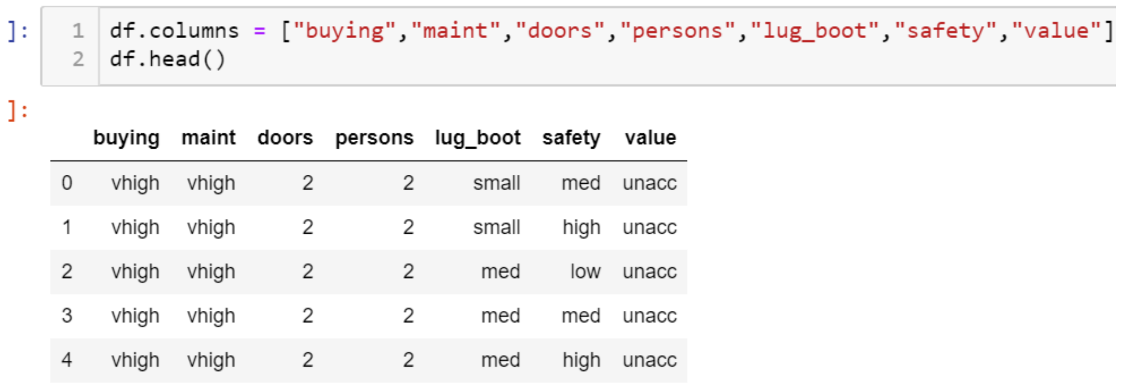
5.1.4 Handling Missing Values:

In my Data set there are no missing values.

Missing values can be handled in many ways using some inbuilt methods.

5.1.5 Categorical Data

- Machine Learning models are based on equations; we need to replace the text by numbers. So that we can include the numbers in the equations.



```
]:  df.columns = ["buying","maint","doors","persons","lug_boot","safety","value"]
    df.head()
```

```
]:
       buying  maint  doors  persons  lug_boot  safety  value
    0  vhigh   vhigh      2        2     small     med  unacc
    1  vhigh   vhigh      2        2     small    high  unacc
    2  vhigh   vhigh      2        2       med     low  unacc
    3  vhigh   vhigh      2        2       med     med  unacc
    4  vhigh   vhigh      2        2       med    high  unacc
```

Fig 5.1.5.1: printing data

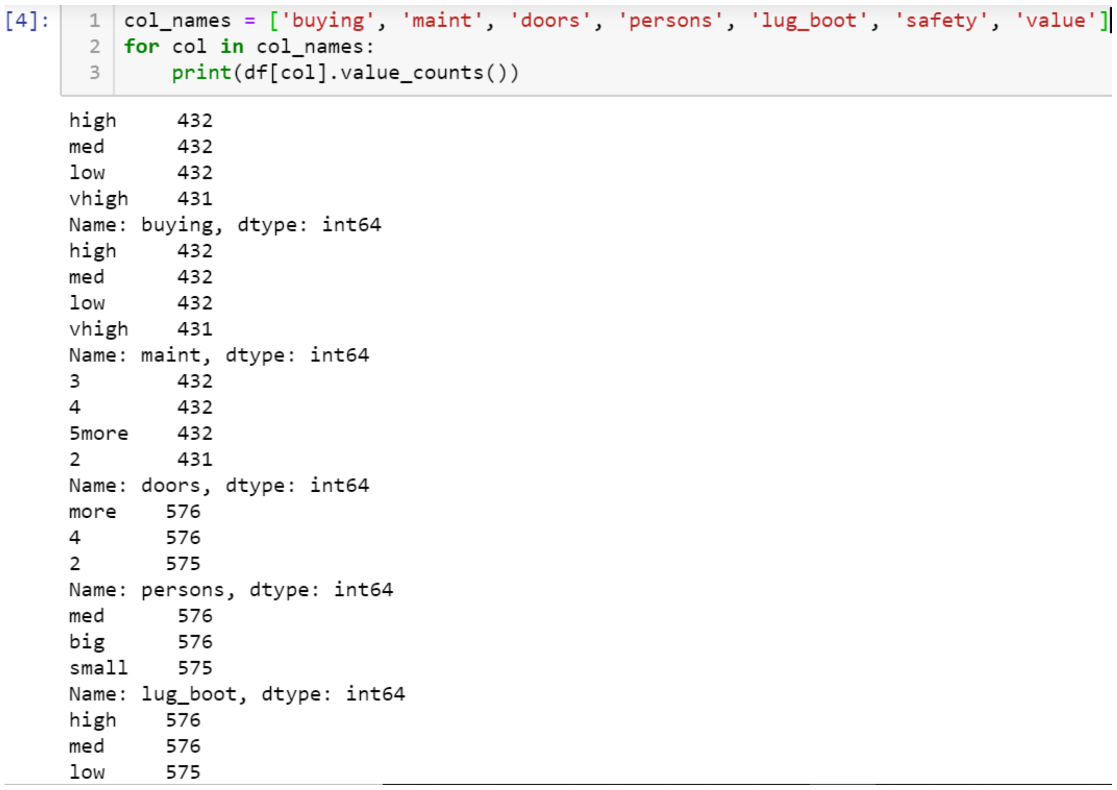
● Categorical Variables are of two types: Nominal and Ordinal

● **Nominal**: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any color

● **Ordinal**: The categories have a numerical ordering in between them. Example:

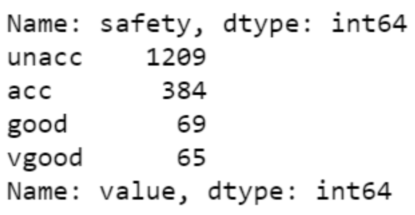
Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer

satisfaction survey, high low medium



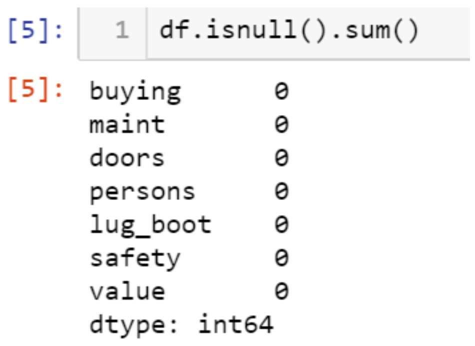
```
[4]:  col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'value']
      for col in col_names:
          print(df[col].value_counts())
```

```
high     432
med      432
low      432
vhigh    431
Name: buying, dtype: int64
high     432
med      432
low      432
vhigh    431
Name: maint, dtype: int64
3        432
4        432
5more    432
2        431
Name: doors, dtype: int64
more    576
4       576
2       575
Name: persons, dtype: int64
med      576
big      576
small    575
Name: lug_boot, dtype: int64
high    576
med     576
low     575
```



```
Name: safety, dtype: int64
unacc    1209
acc       384
good       69
vgood      65
Name: value, dtype: int64
```

Fig 5.1.5.2: Categorical Data



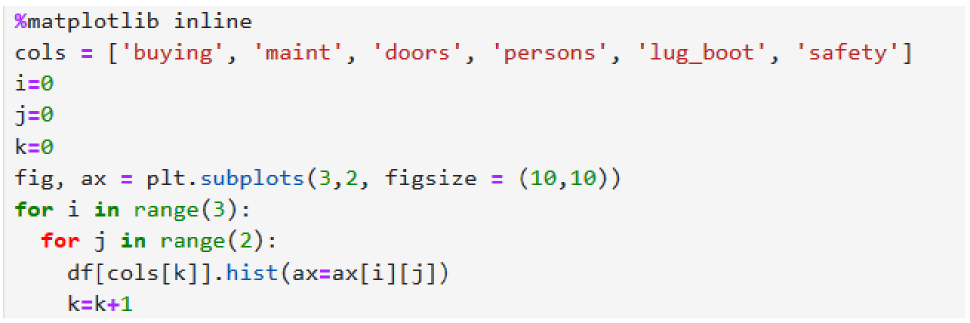
```
[5]:  df.isnull().sum()

[5]:  buying      0
      maint       0
      doors       0
      persons     0
      lug_boot    0
      safety      0
      value       0
      dtype: int64
```

Fig : 5.1.5.3: Getting count of Null values

5.2 Generating Plots

5.2.1 Visualize the data between all the Features

Using inline matplotlib I generated the graphs (Histograms) to get count of each label inside categorical feature:



```
%matplotlib inline
cols = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety']
i=0
j=0
k=0
fig, ax = plt.subplots(3,2, figsize = (10,10))
for i in range(3):
  for j in range(2):
    df[cols[k]].hist(ax=ax[i][j])
    k=k+1
```

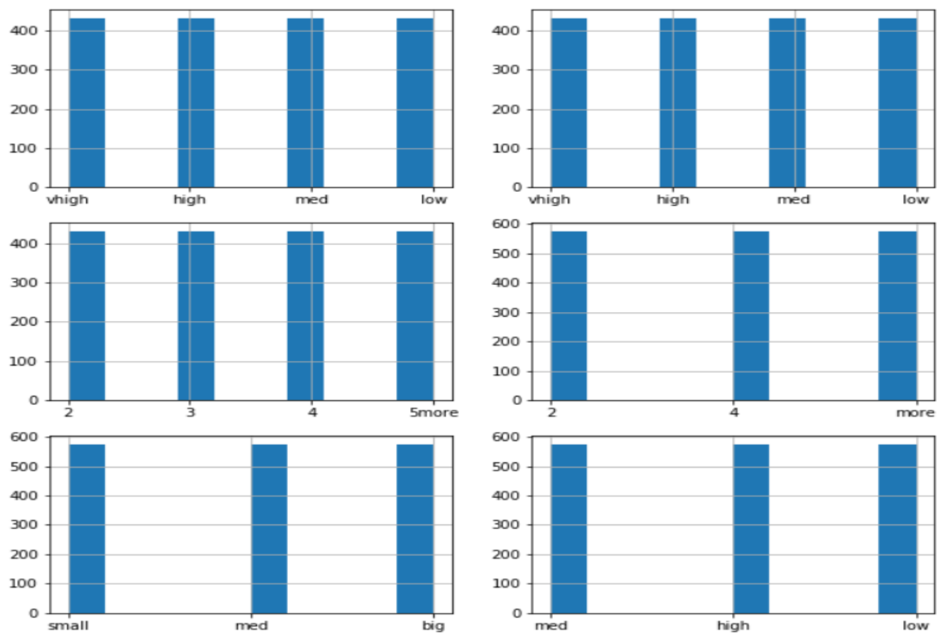


Fig: 5.2.1.1: visualization of data in between all the features

5.2.2 Visualize the data between Target and the Features

Now for visualization of data against the target value I used seaborn

- The below figure depicts the relation between buying and target value

```
import seaborn as sns
sns.countplot(df['buying'],hue = df['value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2719c203848>
```

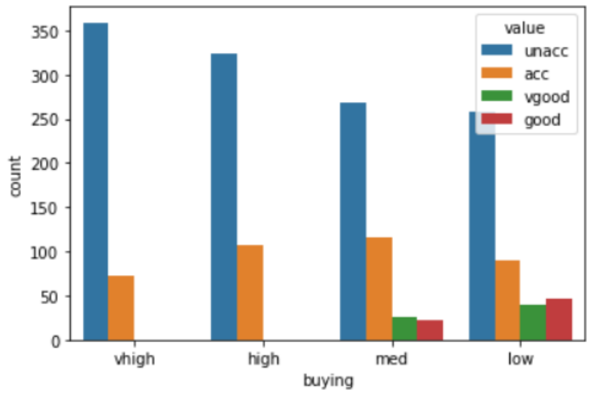


Fig 5.2.2.1

- The below figure depicts the relation between maintenance and target value

```
sns.countplot(df['maint'],hue = df['value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2719c2be988>
```

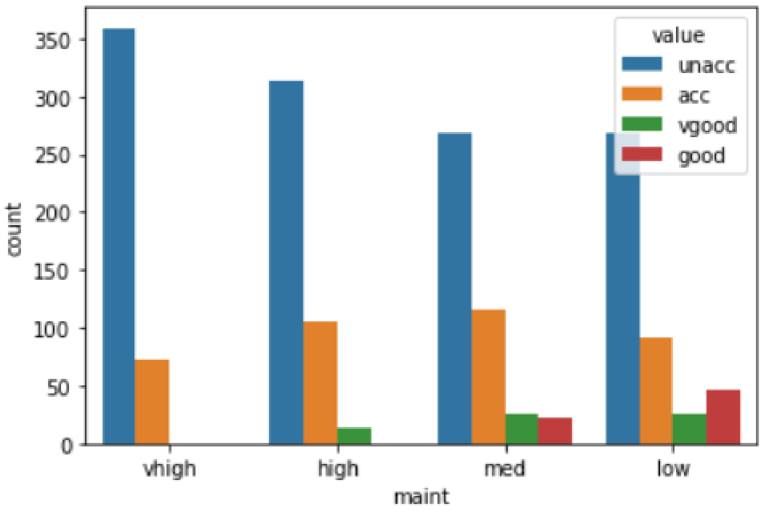


Fig 5.2.2.2

- The below figure depicts the relation between no of doors and target value

```
sns.countplot(df['doors'],hue = df['value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2719c360308>
```

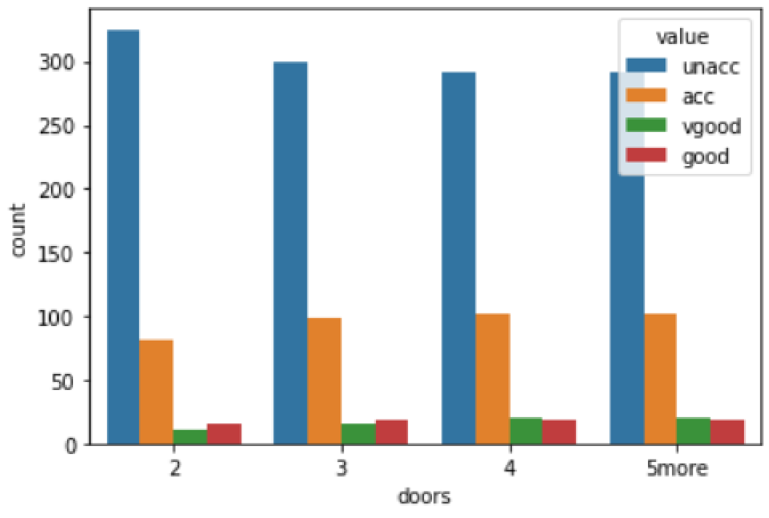


Fig 5.2.2.3

- The below figure depicts the relation in between no of persons and target value

```
sns.countplot(df['persons'],hue = df['value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2719c3fb288>
```

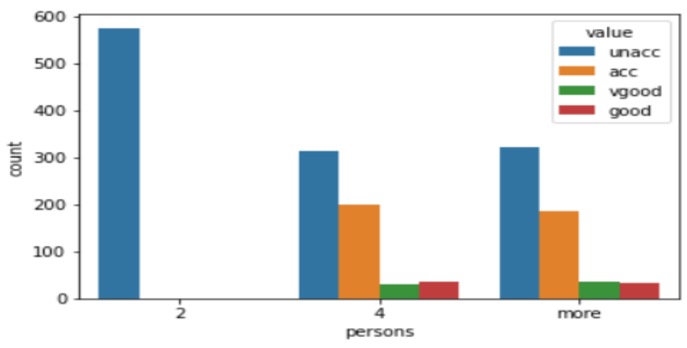


Fig 5.2.2.4

- The below figure depicts the relation in between no of persons and target value

```
sns.countplot(df['lug_boot'],hue = df['value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2719c462408>
```

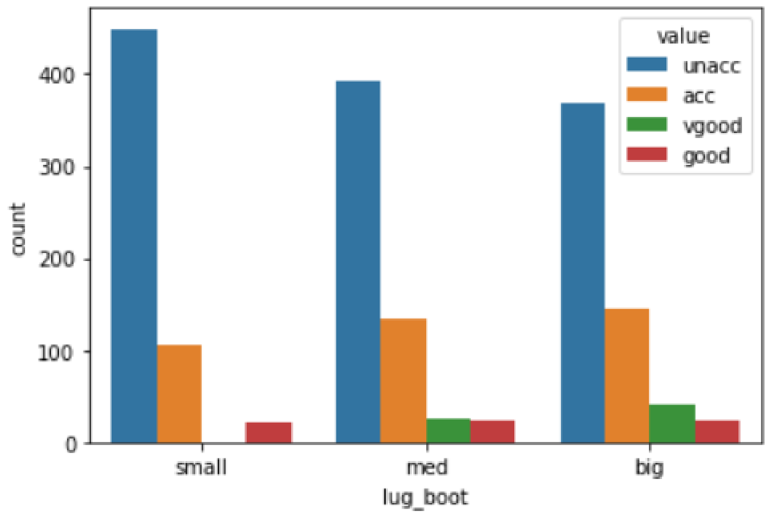


Fig 5.2.2.5

- The below figure depicts the relation in between no of persons and target value

```
sns.countplot(df['safety'],hue = df['value'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2719c4f43c8>
```

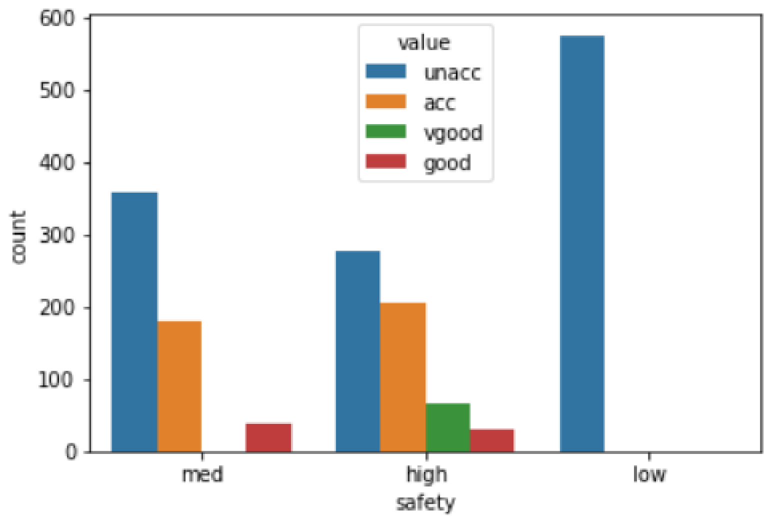


Fig 5.2.2.6

Now in visualizing the target value i found that data set is imbalanced.

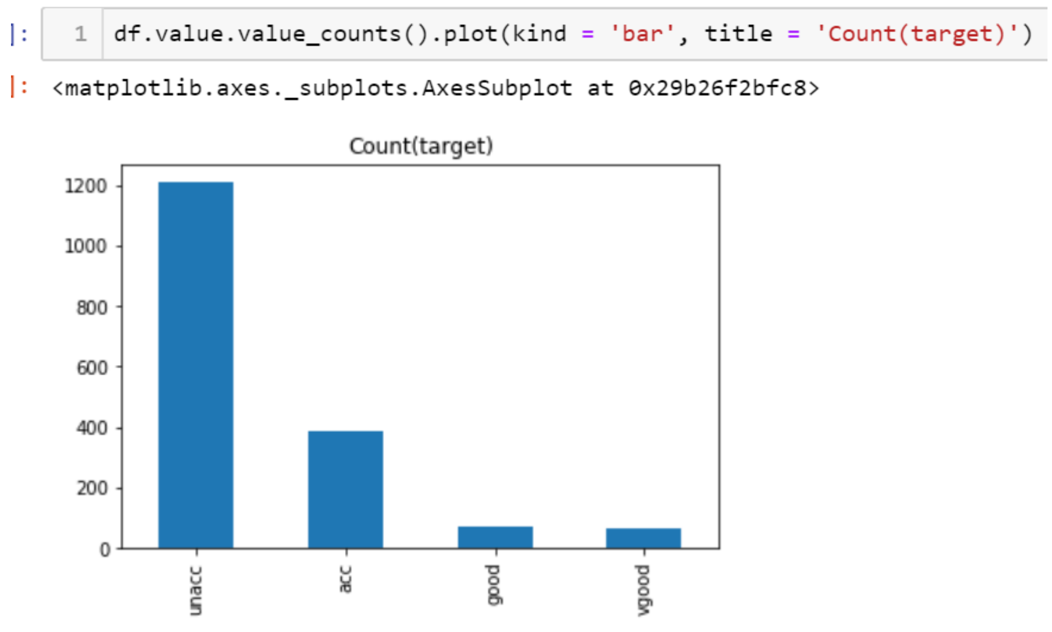
What is Imbalanced Data?

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. Imbalanced data typically refers to a classification problem where the number of observations per class is not equally distributed; often you'll have a large amount of data/observations for one class (referred to as the majority class), and much fewer observations for one or more other classes (referred to as the minority classes)

You can have a class imbalance problem on two-class classification problems as well as multi-class classification problems. Most techniques can be used on either.

The Problem with Imbalanced Classes?

Most machine learning algorithms work best when the number of samples in each class are about equal. This is because most algorithms are designed to maximize accuracy and reduce error.



```
]:  df.value.value_counts().plot(kind = 'bar', title = 'Count(target)')
]:  <matplotlib.axes._subplots.AxesSubplot at 0x29b26f2bfc8>
```

Fig 5.2.2.7: imbalanced data

To overcome this I am over sampling/ Up sampling the data

Techniques to handle Imbalanced Dataset:

1) **Use the right evaluation metrics:**

Applying inappropriate evaluation metrics for models generated using imbalanced data can be dangerous. Imagine our training data is the one illustrated in the graph above. If accuracy is used to measure the goodness of a model, a model which classifies all testing samples into "0" will have an excellent accuracy (99.8%), but obviously, this model won't provide any valuable information for us. In this case, other alternative evaluation metrics can be applied such as:

- Precision/Specificity: how many selected instances are relevant.
- recall/Sensitivity: how many relevant instances are selected.
- F1 score: harmonic mean of precision and recall.
- AUC: relation between true-positive rate and false positive rate.

2) Under Sampling (Down Sampling Majority Class):

Under sampling can be defined as removing some observations of the majority class. Under sampling can be a good choice when you have a ton of data - think millions of rows. But a drawback is that we are removing information that may be valuable. Under-sampling balances the dataset by reducing the size of the abundant class. This method is used when quantity of data is sufficient. By keeping all samples in the minority class and randomly selecting an equal number of samples in the majority class, a balanced new dataset can be retrieved for further modelling.

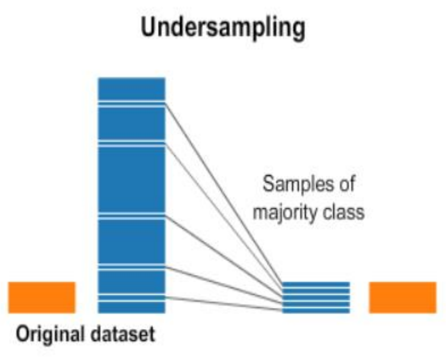


Fig 5.2.2.8: under sampling

3) **Oversampling (Up Sampling of Minority Class):**

Up Sampling can be defined as adding more copies of the minority class. Up-sampling is the process of randomly duplicating observations from the minority class in order to reinforce its signal.

There are several heuristics for doing so, but the most common way is to simply resample with replacement.

Oversampling can be a good choice when you don't have a ton of data to work with. We will use the resampling module from Scikit-Learn to randomly replicate samples from the minority class.

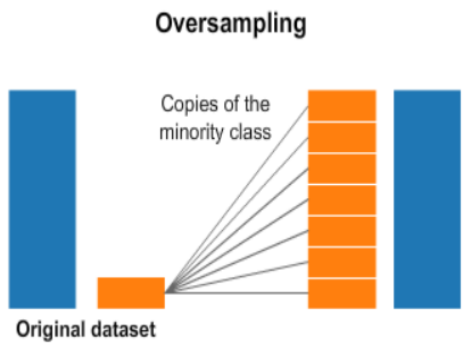
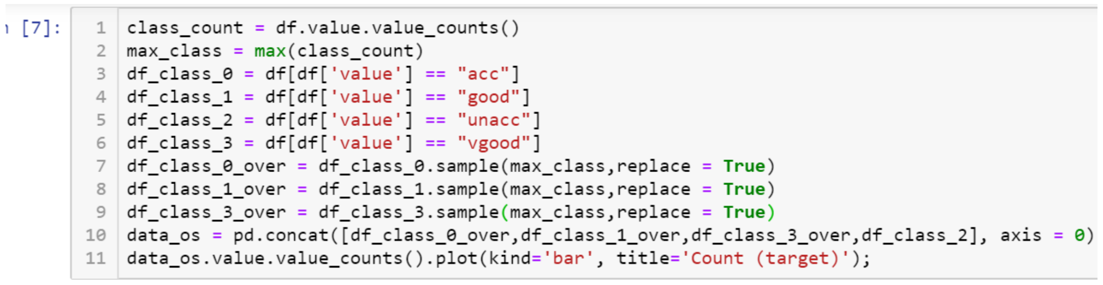


Fig 5.2.2.9: Over Sampling



```
class_count = df.value.value_counts()
max_class = max(class_count)
df_class_0 = df[df['value'] == "acc"]
df_class_1 = df[df['value'] == "good"]
df_class_2 = df[df['value'] == "unacc"]
df_class_3 = df[df['value'] == "vgood"]
df_class_0_over = df_class_0.sample(max_class,replace = True)
df_class_1_over = df_class_1.sample(max_class,replace = True)
df_class_3_over = df_class_3.sample(max_class,replace = True)
data_os = pd.concat([df_class_0_over,df_class_1_over,df_class_3_over,df_class_2], axis = 0)
data_os.value.value_counts().plot(kind='bar', title='Count (target)');
```

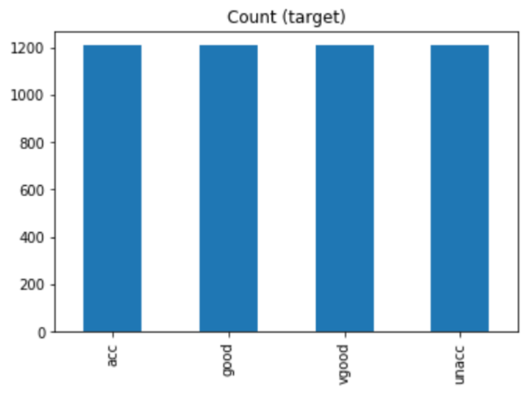
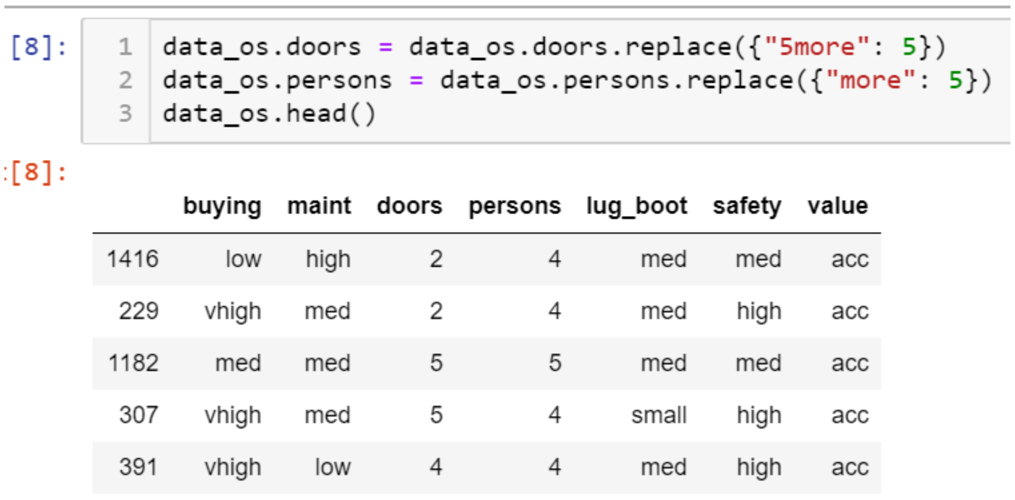


Fig 5.2.2.10: over sampling/ Up sampling the data

5.3 Data Cleaning:

I am Standardizing the features (No of Doors, and No of persons) to value 5.



```
[8]:  data_os.doors = data_os.doors.replace({"5more": 5})
      data_os.persons = data_os.persons.replace({"more": 5})
      data_os.head()
```

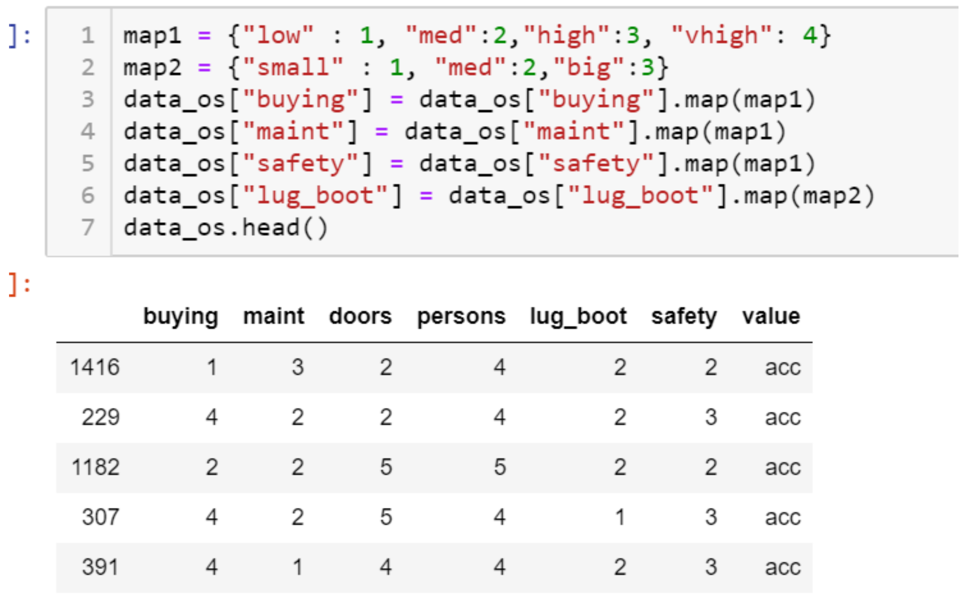
:[8]:

	buying	maint	doors	persons	lug_boot	safety	value
1416	low	high	2	4	med	med	acc
229	vhigh	med	2	4	med	high	acc
1182	med	med	5	5	med	med	acc
307	vhigh	med	5	4	small	high	acc
391	vhigh	low	4	4	med	high	acc

Fig 5.3.1: data cleaning

5.4 Encoding Categorical Data:

As Machine Learning can understand only numbers it's time to encode the categorical data to its respective values in a traditional way without using label encoder or pandas.



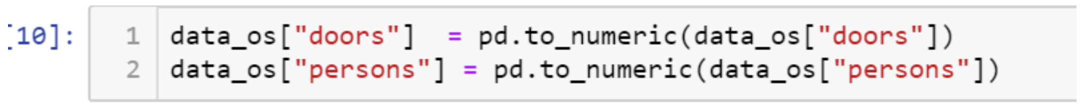
```
]:  map1 = {"low" : 1, "med":2,"high":3, "vhigh": 4}
    map2 = {"small" : 1, "med":2,"big":3}
    data_os["buying"] = data_os["buying"].map(map1)
    data_os["maint"] = data_os["maint"].map(map1)
    data_os["safety"] = data_os["safety"].map(map1)
    data_os["lug_boot"] = data_os["lug_boot"].map(map2)
    data_os.head()
```

]:

	buying	maint	doors	persons	lug_boot	safety	value
1416	1	3	2	4	2	2	acc
229	4	2	2	4	2	3	acc
1182	2	2	5	5	2	2	acc
307	4	2	5	4	1	3	acc
391	4	1	4	4	2	3	acc

Fig 5.4.1: mapping data

For getting further more information in order to train the model well i am introducing a new type of column that is car type.



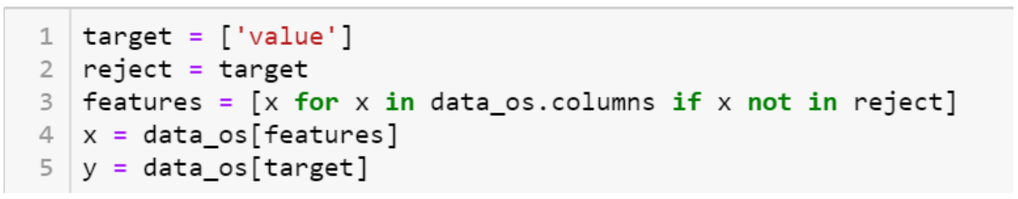
```
[10]:  data_os["doors"]   = pd.to_numeric(data_os["doors"])
       data_os["persons"] = pd.to_numeric(data_os["persons"])
```

Fig 5.4.2

6. Feature Selection

6.1 Select relevant features for the analysis

Splitting the data into independent and Dependent features in my scenario independent values are all features other than the Target value

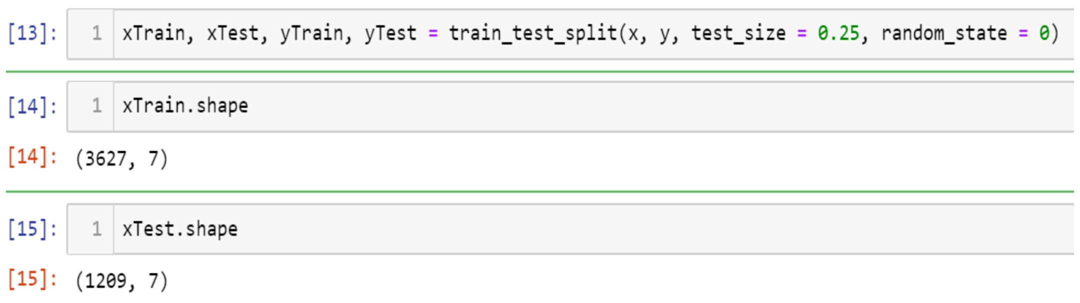


```
target = ['value']
reject = target
features = [x for x in data_os.columns if x not in reject]
x = data_os[features]
y = data_os[target]
```

Fig 6.1.1: splitting data

6.2 Train and Test Split:

Splitting data into Train and Test



```
[13]:  xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size = 0.25, random_state = 0)

[14]:  xTrain.shape
[14]:  (3627, 7)

[15]:  xTest.shape
[15]:  (1209, 7)
```

Fig 6.2.1: train & test split data

7. MODEL BUILDING AND EVALUATION

I Used 2 classifiers for predicting the output

7.1 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning, which will be the main focus of this article.

How can an algorithm be represented as a tree?

For this let's consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

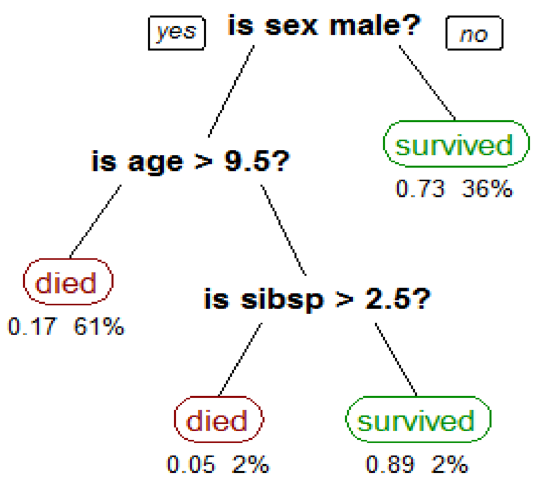


Fig 7.1.1: decision tree algorithm structure

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real data set will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

So, what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful. Let's start with a common technique used for splitting.

7.1.1 RECURSIVE BINARY SPLITTING:

Fig 7.1.1.1: Recursive Binary Splitting

In this procedure all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.

Consider the earlier example of tree learned from titanic data set. In the first split or the root, all attributes/features are considered and the training data is divided into groups based on this split. We have 3 features, so will have 3 candidate splits. Now we will calculate how much accuracy each split will cost us, using a function. The split that costs least is chosen, which in our example is sex of the passenger. This algorithm is recursive in nature as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the greedy algorithm, as we have an excessive desire of lowering the cost. This makes the root node as best predictor/classifier.

7.1.2 COST OF A SPLIT:

Lets take a closer look at cost functions used for classification and regression. In both cases the cost functions try to find most homogeneous branches, or branches having groups with similar responses. This makes sense we can be surer that a test data input will follow a certain path.

$$\text{Regression: sum } (y — \text{prediction})^2$$

Let’s say, we are predicting the price of houses. Now the decision tree will start splitting by considering each feature in training data. The mean of responses of the training data inputs of particular group is considered as prediction for that group. The above function is applied to all data points and cost is calculated for all candidate splits. Again, the split with lowest cost is chosen. Another cost function involves reduction of standard deviation, more about it can be found here.

$$\text{Classification: } G = \text{sum } (pk * (1 — pk))$$

A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split. Here, pk is proportion of same class inputs present in a particular group. A perfect class purity occurs when a group contains all inputs from the same class, in which case pk is either 1 or 0 and G = 0, where as a node having a 50–50 split of classes in a group has the worst purity, so for a binary classification it will have pk = 0.5 and G = 0.5.

When to stop splitting?

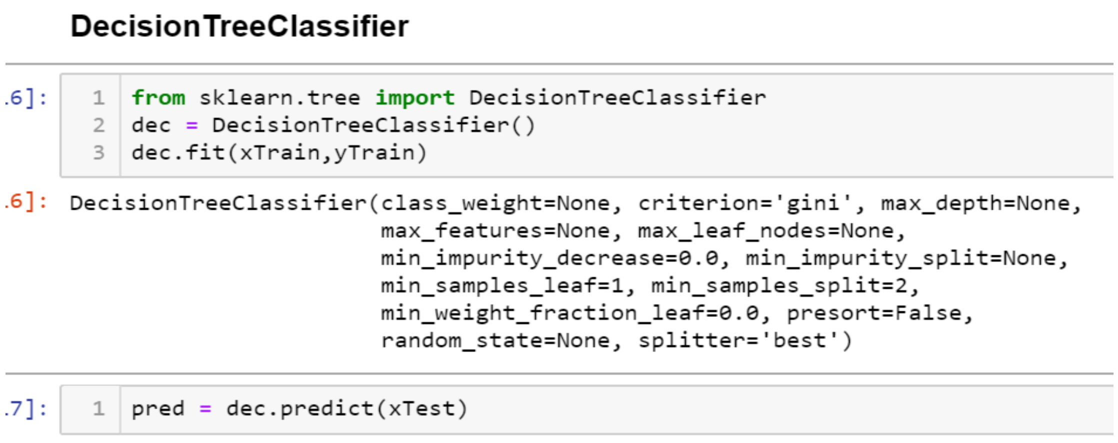
As a problem usually has a large set of features, it results in large number of splits, which in turn gives a huge tree. Such trees are complex and can lead to over fitting. So, we need to know when to stop? One way of doing this is to set a minimum number of training inputs to use on each leaf. For example, we can use a minimum of 10 passengers to reach a decision (died or survived), and ignore any leaf that takes less than 10 passengers. Another way is to set maximum depth of your model. Maximum depth refers to the the length of the longest path from a root to a leaf.

7.1.3 PRUNING:

The performance of a tree can be further increased by pruning. It involves removing the branches that make use of features having low importance. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing over fitting.

Pruning can start at either root or the leaves. The simplest method of pruning starts at leaves and removes each node with most popular class in that leaf, this change is kept if it doesn't deteriorate accuracy. It’s also called reduced error pruning. More sophisticated pruning methods can be used such as cost complexity pruning where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree. This is also known as weakest link pruning.

7.1.4 Train the Models



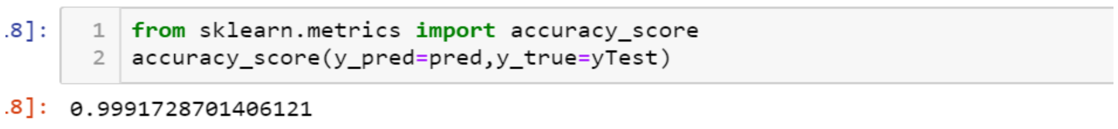
DecisionTreeClassifier

```
.6]: from sklearn.tree import DecisionTreeClassifier
     dec = DecisionTreeClassifier()
     dec.fit(xTrain,yTrain)

.6]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                            max_features=None, 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, presort=False,
                            random_state=None, splitter='best')

.7]: pred = dec.predict(xTest)
```

Fig 7.1.4.1: Decision Tree Classifier

7.1.5 METRICS

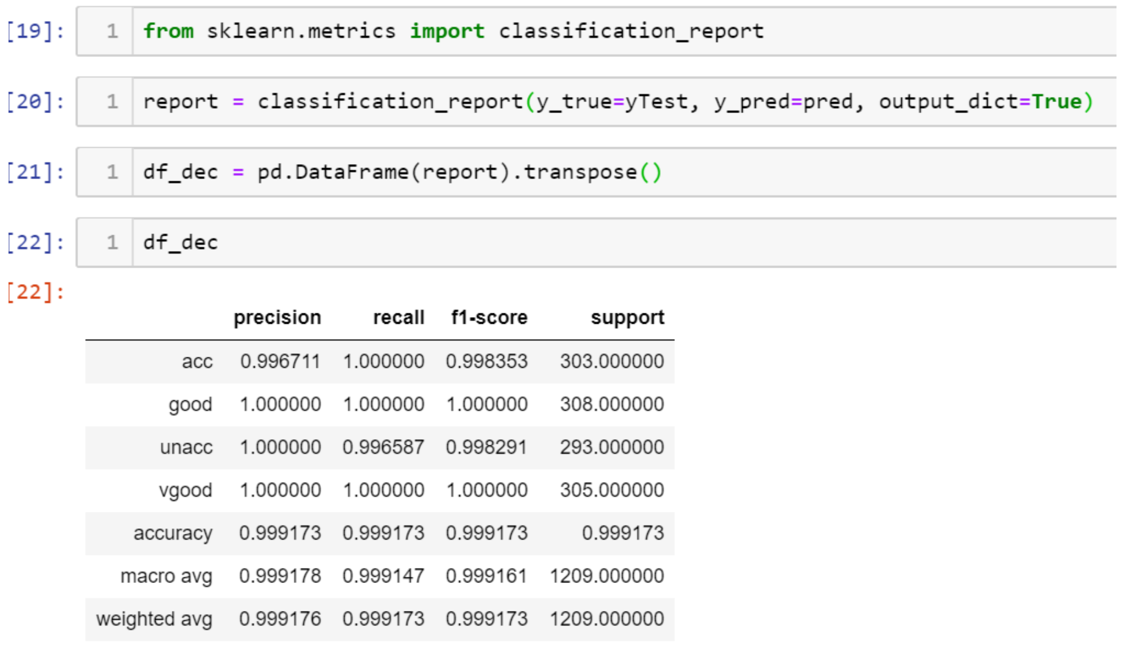
Accuracy Score:



```
.8]: from sklearn.metrics import accuracy_score
     accuracy_score(y_pred=pred,y_true=yTest)

.8]: 0.9991728701406121
```

Fig 7.1.5.1: Decision Tree Classifier (accuracy score)

Classification report:

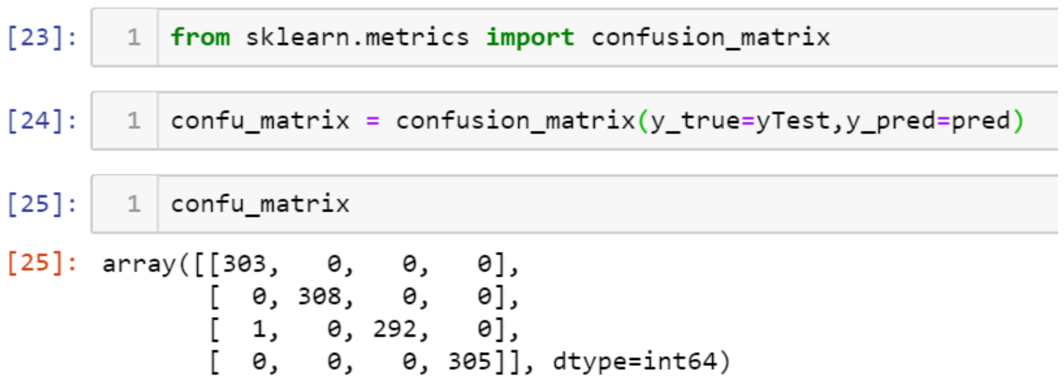


```
[19]:  from sklearn.metrics import classification_report

[20]:  report = classification_report(y_true=yTest, y_pred=pred, output_dict=True)

[21]:  df_dec = pd.DataFrame(report).transpose()

[22]:  df_dec
[22]:
```

	precision	recall	f1-score	support
acc	0.996711	1.000000	0.998353	303.000000
good	1.000000	1.000000	1.000000	308.000000
unacc	1.000000	0.996587	0.998291	293.000000
vgood	1.000000	1.000000	1.000000	305.000000
accuracy	0.999173	0.999173	0.999173	0.999173
macro avg	0.999178	0.999147	0.999161	1209.000000
weighted avg	0.999176	0.999173	0.999173	1209.000000

Fig 7.1.5.2: Decision Tree Classifier (classification report)

Confusion Matrix:



```
[23]:  from sklearn.metrics import confusion_matrix

[24]:  confu_matrix = confusion_matrix(y_true=yTest,y_pred=pred)

[25]:  confu_matrix

[25]:  array([[303,   0,   0,   0],
              [  0, 308,   0,   0],
              [  1,   0, 292,   0],
              [  0,   0,   0, 305]], dtype=int64)
```

Fig 7.1.5.3: Decision Tree Classifier (confusion matrix)

7.2 RANDOM FOREST CLASSIFIER

Random Forest Classifier

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

7.2.1 Ensemble Algorithm:

Ensemble algorithms are those which combines more than one algorithm of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object.

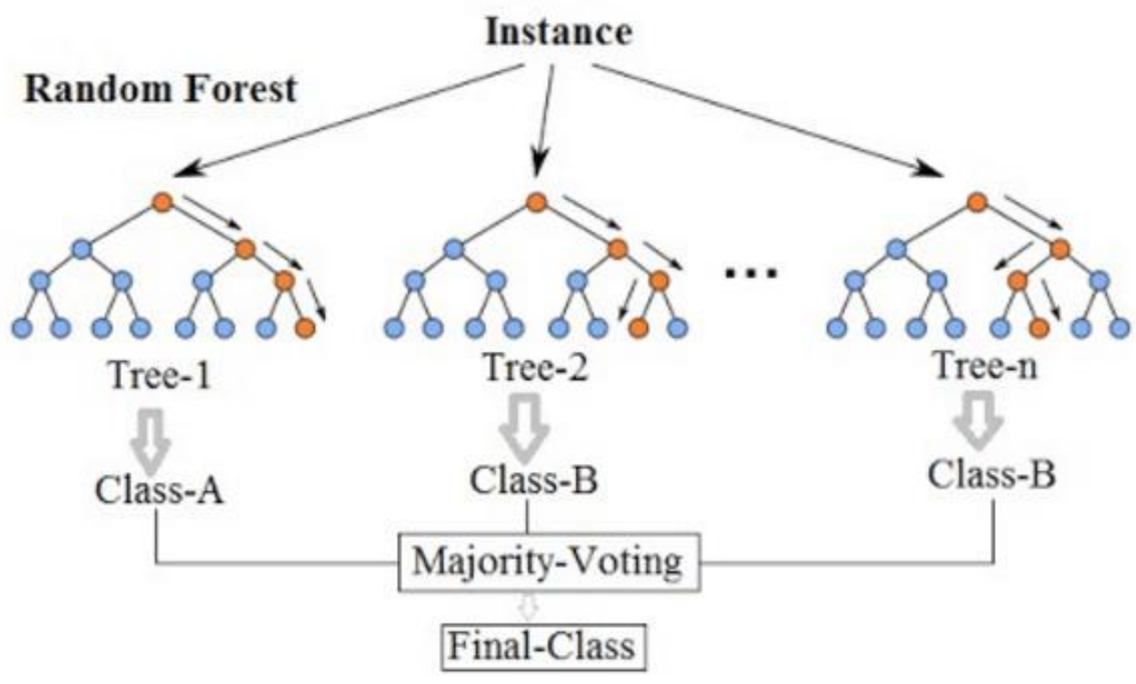
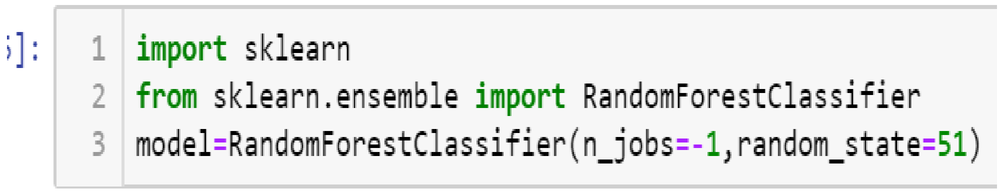


Fig 7.2.1.1: Structure of Random Forest Classification

Features and Advantages of Random Forest:

1. It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
2. It runs efficiently on large databases.
3. It can handle thousands of input variables without variable deletion.
4. It gives estimates of what variables that are important in the classification.
5. It generates an internal unbiased estimate of the generalization error as the forest building progresses.
6. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

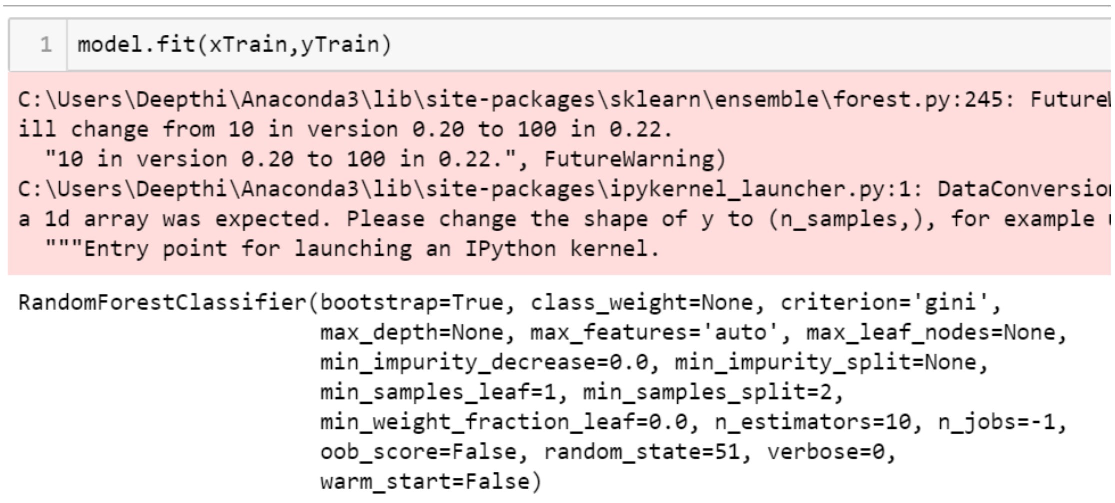
Model Creation:



```
import sklearn
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_jobs=-1,random_state=51)
```

Fig 7.2.1.4: random forest (model creation)

Model Fitting:



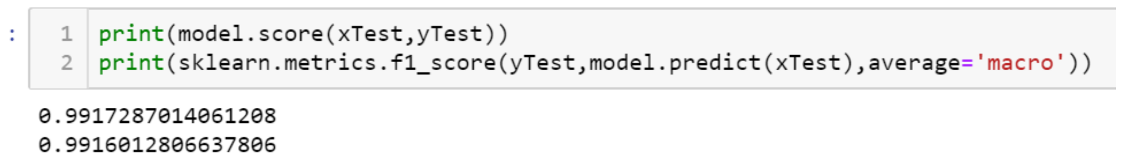
```
model.fit(xTrain,yTrain)
```

```
C:\Users\Deepthi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureW
ill change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Deepthi\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConversio
a 1d array was expected. Please change the shape of y to (n_samples,), for example u
  """Entry point for launching an IPython kernel.
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       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=-1,
                       oob_score=False, random_state=51, verbose=0,
                       warm_start=False)
```

Fig 7.2.1.5: random forest (model fitting)

Evaluating the model:



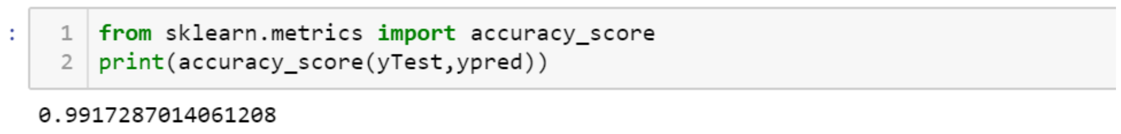
```
print(model.score(xTest,yTest))
print(sklearn.metrics.f1_score(yTest,model.predict(xTest),average='macro'))
```

```
0.9917287014061208
0.9916012806637806
```

Fig 7.2.1.6: random forest (evaluating the model)

7.2.2 METRICS

Accuracy Score:

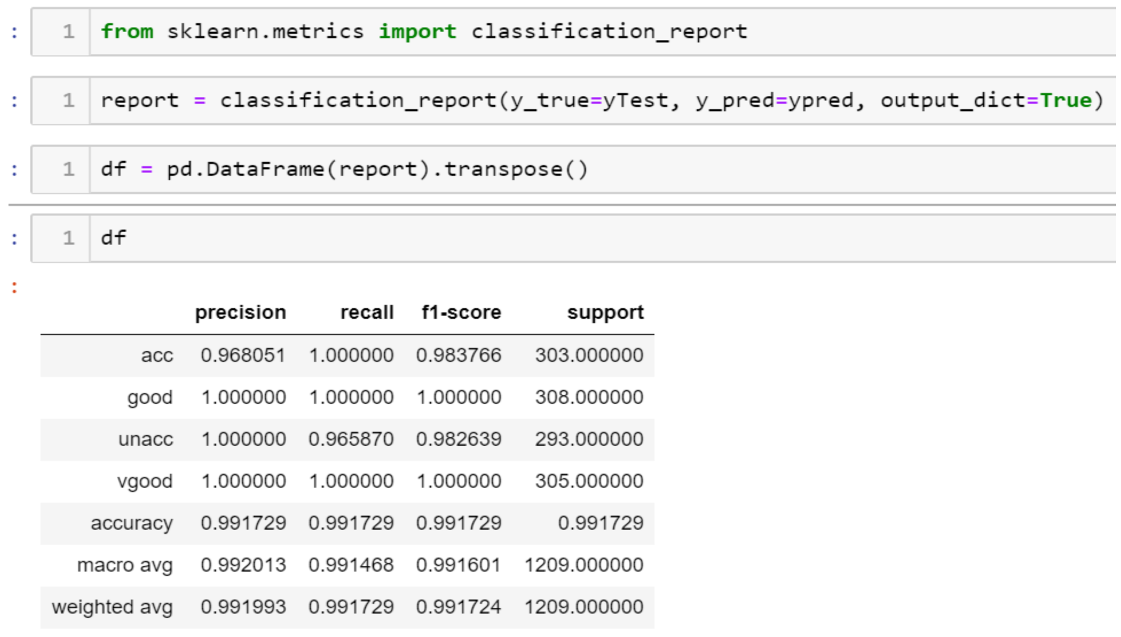


```
from sklearn.metrics import accuracy_score
print(accuracy_score(yTest,ypred))
```

```
0.9917287014061208
```

Fig 7.2.1.7: random forest (accuracy score)

Classification report:



```
from sklearn.metrics import classification_report
```

```
report = classification_report(y_true=yTest, y_pred=ypred, output_dict=True)
```

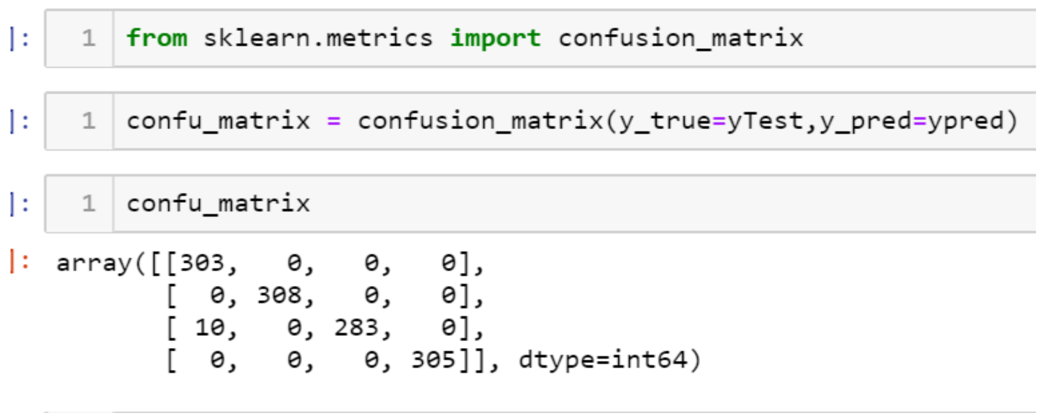
```
df = pd.DataFrame(report).transpose()
```

```
df
```

	precision	recall	f1-score	support
acc	0.968051	1.000000	0.983766	303.000000
good	1.000000	1.000000	1.000000	308.000000
unacc	1.000000	0.965870	0.982639	293.000000
vgood	1.000000	1.000000	1.000000	305.000000
accuracy	0.991729	0.991729	0.991729	0.991729
macro avg	0.992013	0.991468	0.991601	1209.000000
weighted avg	0.991993	0.991729	0.991724	1209.000000

Fig 7.2.1.8: random forest (classification report)

Confusion matrix:



```
]:  from sklearn.metrics import confusion_matrix
```

```
]:  confu_matrix = confusion_matrix(y_true=yTest,y_pred=ypred)
```

```
]:  confu_matrix
```

```
]: array([[303,   0,   0,   0],
          [  0, 308,   0,   0],
          [ 10,   0, 283,   0],
          [  0,   0,   0, 305]], dtype=int64)
```

Fig 7.2.1.9: random forest(confusion matrix*)*

8. Conclusion

Through this project, the classification of cars over evaluation is made up of a machine learning model that can be used to infer the price of a car based on the user 's requirement so that he / she can pick the appropriate vehicle, all classification evaluation metrics are evaluated and this model can be used in any edge devices such as (e.g.: mobile application) as the customer / future prospectus is used.