Data Science (BIKE BUYERS PREDICTION)

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

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

submitted by

PRUTHVIRAJ CHINTAKINDI(221810302013)

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 SEPTEMBER-2021

DECLARATION

I submit this industrial training work entitled “**Predicting bike buyers**” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “Bachelor of Technology” in “Computer Science and 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 **PRUTHVIRAJ CHINTAKINDI**

Date: 221810302013

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

PRUTHVIRAJ

ABSTRACT

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. The concepts of data science extend but not limited to machine learning, deep learning and artificial intelligence. Through data science the unwanted data can be eliminated easily, which means clean data can obtained very swiftly. The relation among parameters can also found out easily with not much effort being applied. In the project of car classification evaluation, the concepts of data science play an important role as the main aim is to predict the “decision” parameter.

In the project every parameter is mapped with “decision” parameter in order to find the correlation among them. Also, the proportions of all parameters are distributed to get a clear understanding of density of values. This helps us approach a method to make predictions more accurate and suitable.

TABLE OF CONTENT

1. Information About Data Science

1.1 What is Data Science

Data science is an interdisciplinary 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, etc.

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.

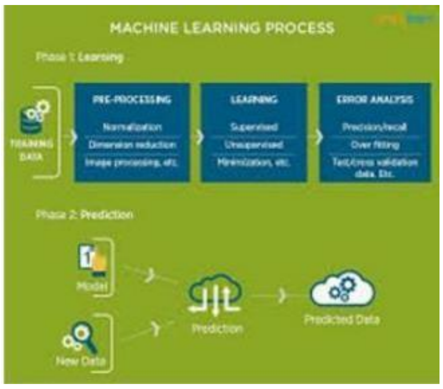


Fig 2.2.1 : The Process Flow

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.

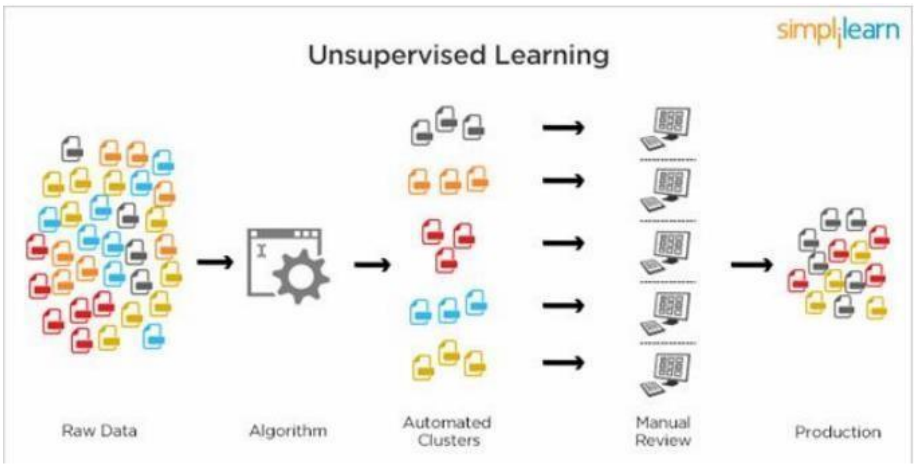


Fig 2.4.2.1: Unsupervised Learning

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.

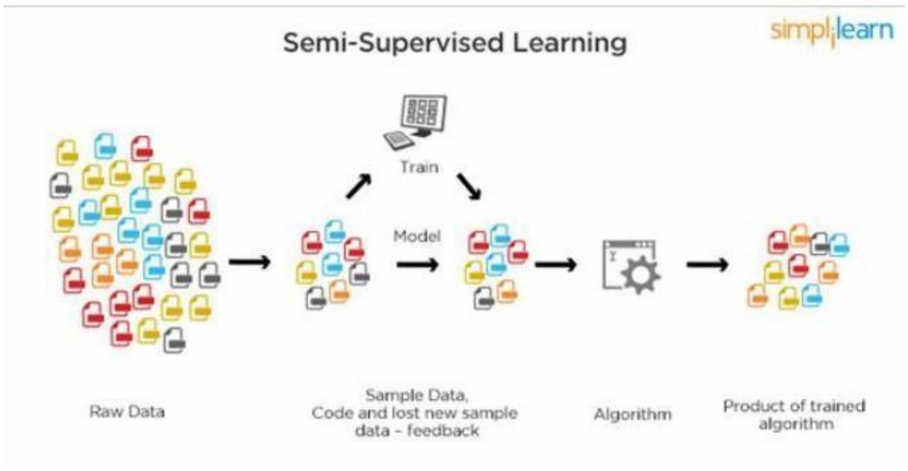

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Fig2.4.3.1: 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):

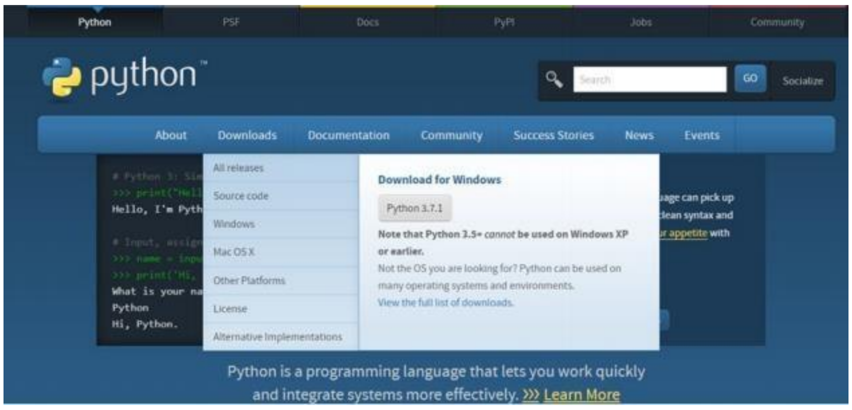
● Installing python is generally easy, and nowadays many Linux and Mac

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.



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)

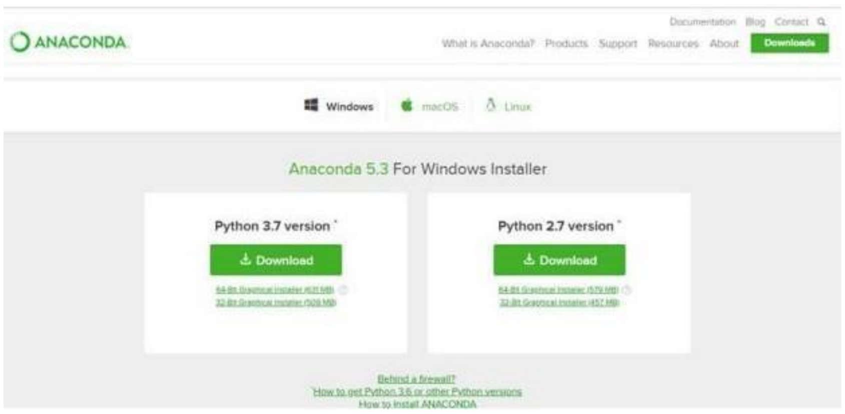

ANACONDA
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version
Download
Python 2.7 version
Download
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

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 tobe included in the function and structures the blocks of code. Once the basic structureof 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.

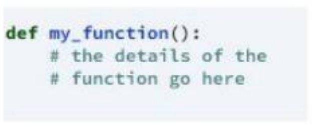
● Data member: A class variable or instance variable that holds data associated with a class and its objects.

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

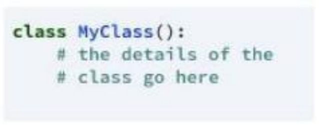
● Defining a Class:

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 : bike buyers prediction

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.

- ✔ Ensemble Algorithm: Random forest
- ✔ Logistic regression

4.2 Problem Statement

To predict the sale of a bike based on the customer’s data.

- "Id" : unique id of customer
- “Gender” : Male/Female
- “Married” : Applicant married or not
- "Income": Income of customer
- "Children": Number of children
- "Education": Type of education
- "Occupation": Type of occupation
- "Homeowner": has a home or not
- "Cars": Has car or not
- "Commute distance": Distance from home to workplace
- "Region": the region to which the customer lives in
- "Age": Age of the customer
- "Purchased bike ": whether the customer already has a bike or not.

  Note: Source (Kaggle)

4.3 The objective of the Case Study

To get a better understanding and chalking out a plan of action for the approach taken and the possibility of a sale that might happen with certain customers.

Based on the type of buyers and the information like the number of children, education, occupation, marital status, already having a car, etc the sale or whether the buyer will buy or not is to be predicted.

5 Data Preprocessing/Feature Engineering and EDA

5.1 Preprocessing of The Data

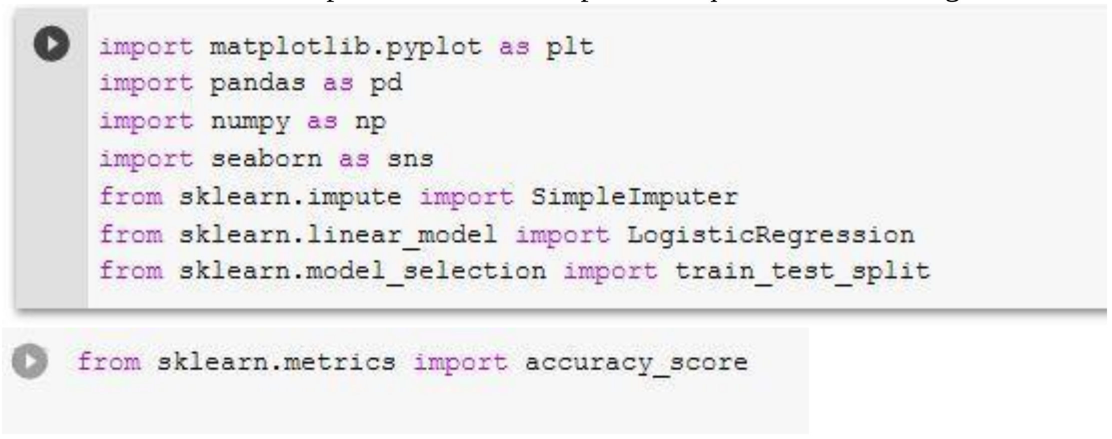
Prepossessing the data actually involves the following steps:

5.1.1 Getting the dataset

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

5.1.2 Importing the Libraries

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



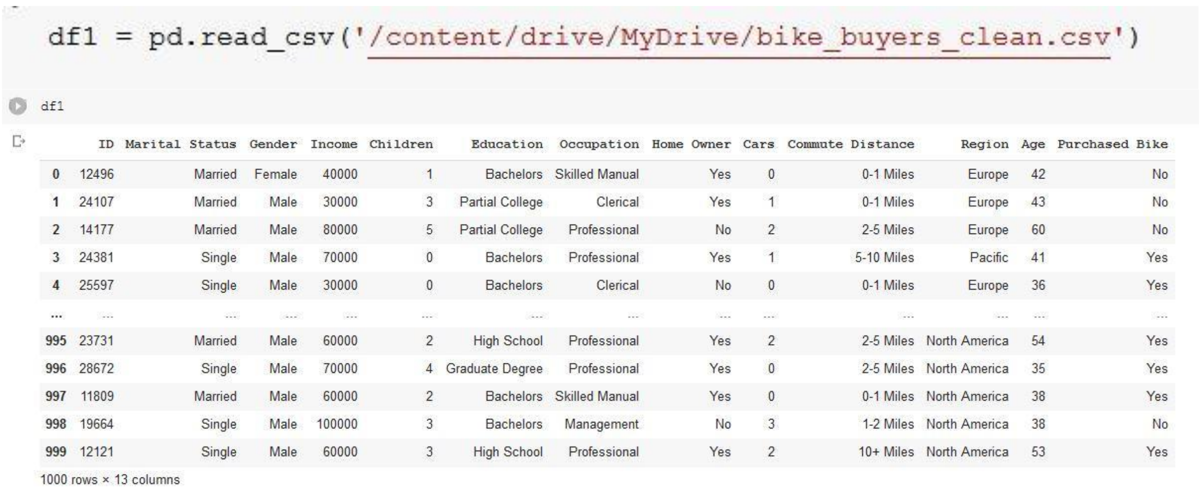
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score
```

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.



```
df1 = pd.read_csv('/content/drive/MyDrive/bike_buyers_clean.csv')
```

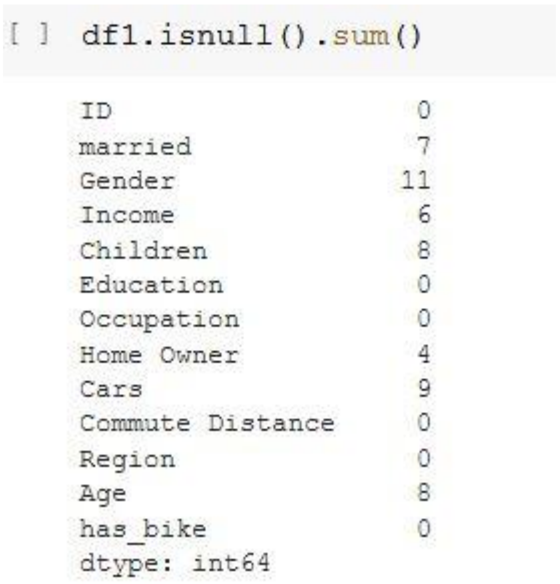
```
df1
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	12496	Married	Female	40000	1	Bachelors	Skilled Manual	Yes	0	0-1 Miles	Europe	42	No
1	24107	Married	Male	30000	3	Partial College	Clerical	Yes	1	0-1 Miles	Europe	43	No
2	14177	Married	Male	80000	5	Partial College	Professional	No	2	2-5 Miles	Europe	60	No
3	24381	Single	Male	70000	0	Bachelors	Professional	Yes	1	5-10 Miles	Pacific	41	Yes
4	25597	Single	Male	30000	0	Bachelors	Clerical	No	0	0-1 Miles	Europe	36	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	23731	Married	Male	60000	2	High School	Professional	Yes	2	2-5 Miles	North America	54	Yes
996	28672	Single	Male	70000	4	Graduate Degree	Professional	Yes	0	2-5 Miles	North America	35	Yes
997	11809	Married	Male	60000	2	Bachelors	Skilled Manual	Yes	0	0-1 Miles	North America	38	Yes
998	19664	Single	Male	100000	3	Bachelors	Management	No	3	1-2 Miles	North America	38	No
999	12121	Single	Male	60000	3	High School	Professional	Yes	2	10+ Miles	North America	53	Yes

1000 rows × 13 columns

Fig 5.1.3.1: Reading the data set

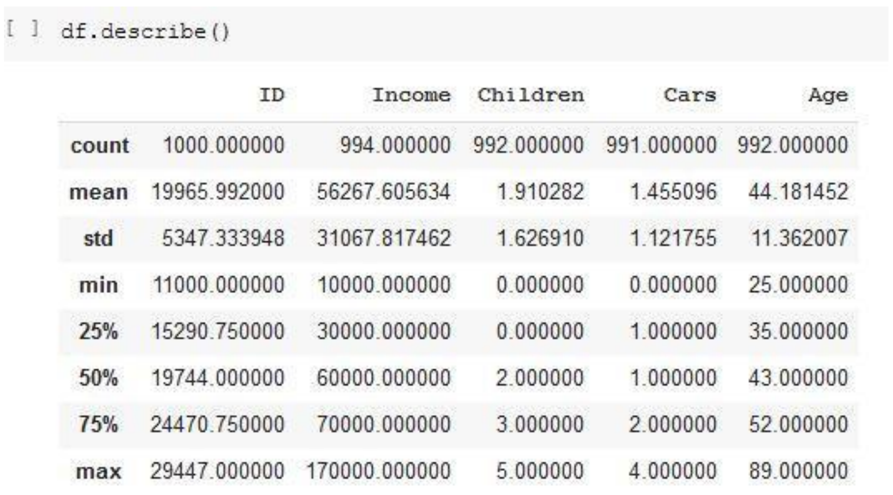
5.1.4 Handling Missing Values:


```
[ ]  df1.isnull().sum()

     ID                  0
     married             7
     Gender             11
     Income              6
     Children            8
     Education           0
     Occupation          0
     Home Owner          4
     Cars                9
     Commute Distance    0
     Region              0
     Age                 8
     has_bike            0
     dtype: int64
```

Missing values can be handled in many ways.

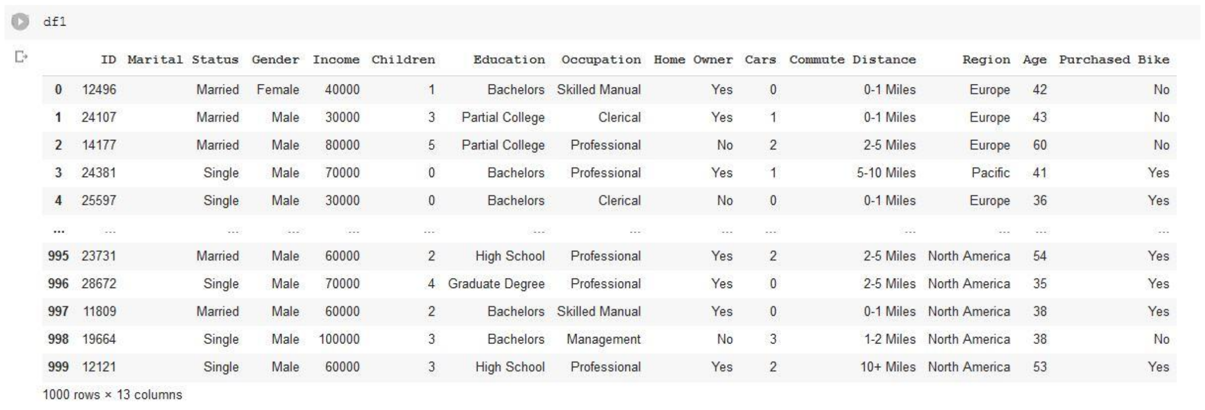
5.1.5 Categorical Data

- 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.describe()
```

```
                 ID         Income    Children        Cars         Age
count   1000.000000     994.000000  992.000000  991.000000  992.000000
mean   19965.992000   56267.605634    1.910282    1.455096   44.181452
std     5347.333948   31067.817462    1.626910    1.121755   11.362007
min    11000.000000   10000.000000    0.000000    0.000000   25.000000
25%    15290.750000   30000.000000    0.000000    1.000000   35.000000
50%    19744.000000   60000.000000    2.000000    1.000000   43.000000
75%    24470.750000   70000.000000    3.000000    2.000000   52.000000
max    29447.000000  170000.000000    5.000000    4.000000   89.000000
```



```
df1
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	12496	Married	Female	40000	1	Bachelors	Skilled Manual	Yes	0	0-1 Miles	Europe	42	No
1	24107	Married	Male	30000	3	Partial College	Clerical	Yes	1	0-1 Miles	Europe	43	No
2	14177	Married	Male	80000	5	Partial College	Professional	No	2	2-5 Miles	Europe	60	No
3	24381	Single	Male	70000	0	Bachelors	Professional	Yes	1	5-10 Miles	Pacific	41	Yes
4	25597	Single	Male	30000	0	Bachelors	Clerical	No	0	0-1 Miles	Europe	36	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	23731	Married	Male	60000	2	High School	Professional	Yes	2	2-5 Miles	North America	54	Yes
996	28672	Single	Male	70000	4	Graduate Degree	Professional	Yes	0	2-5 Miles	North America	35	Yes
997	11809	Married	Male	60000	2	Bachelors	Skilled Manual	Yes	0	0-1 Miles	North America	38	Yes
998	19664	Single	Male	100000	3	Bachelors	Management	No	3	1-2 Miles	North America	38	No
999	12121	Single	Male	60000	3	High School	Professional	Yes	2	10+ Miles	North America	53	Yes

1000 rows × 13 columns

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

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:

```
plt.title('residence region with respect to having bike')
df1.Region.value_counts(sort=False).plot.pie()

plt.show()
```

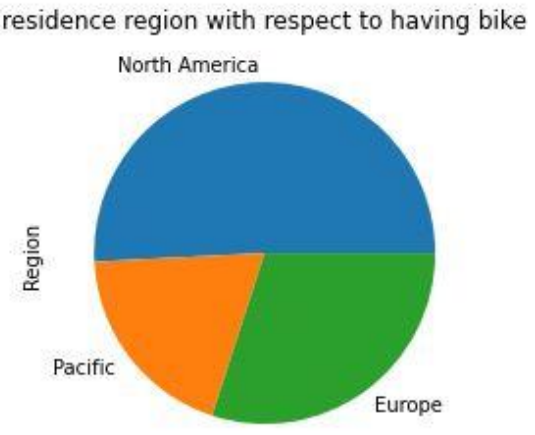


Fig: 5.2.1.1: visualization of data in between all the features

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

```
cols = ['Gender','married','Home Owner','has_bike']

i = 0

j = 0

k = 0

fig,ax = plt.subplots(2,2,figsize = (13,10))

for i in range(2):

    for j in range(2):

        df1[cols[k]].hist(ax=ax[i][j])

        k = k+1
```

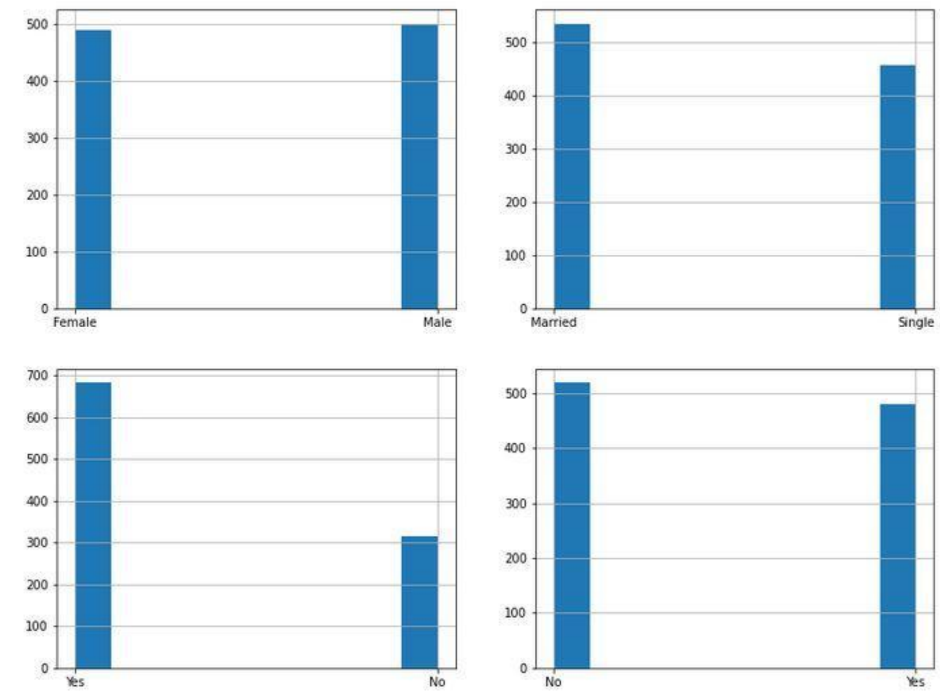
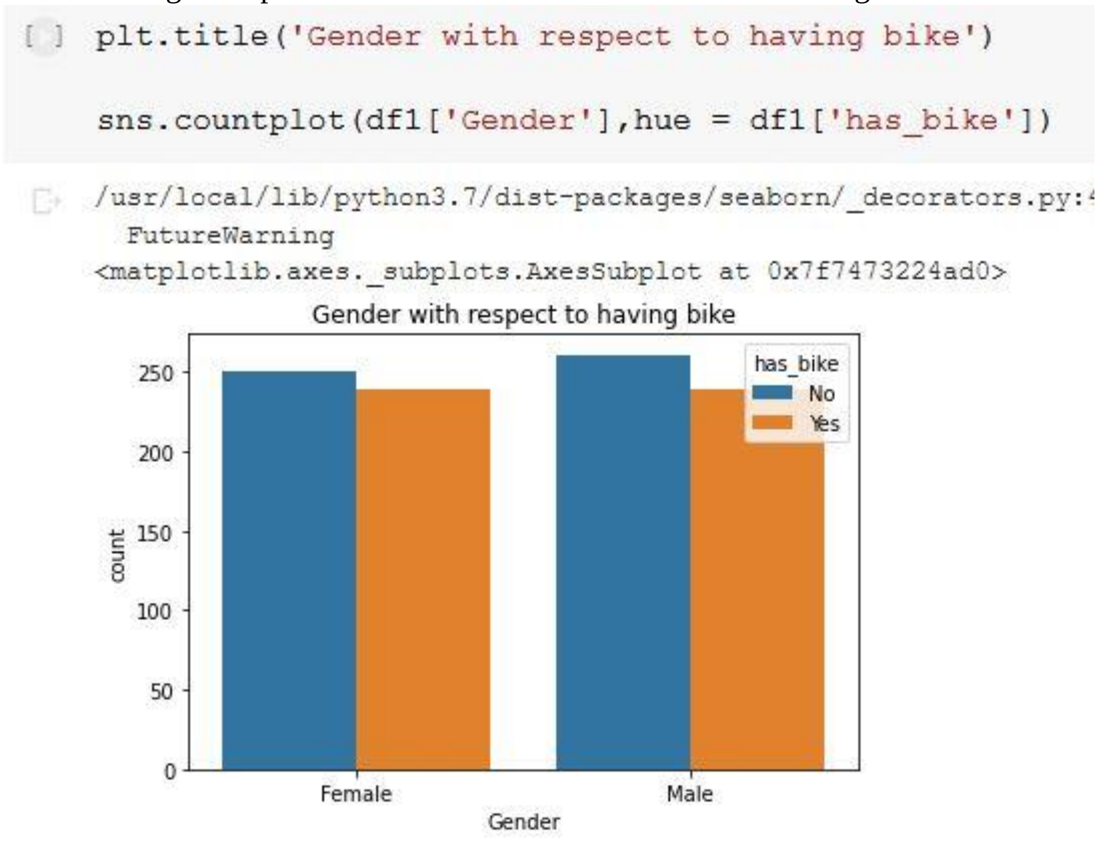


Fig 5.2.2.1

The below figure depicts the relation between maintenance and target value



```
plt.title('Gender with respect to having bike')

sns.countplot(df1['Gender'],hue = df1['has_bike'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7473224ad0>
```

Fig 5.2.2.2

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

```
[ ] plt.title('owing home with respect to having bike')

    sns.countplot(df1['Home Owner'],hue = df1['has_bike'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Fu
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f74730ed490>
```

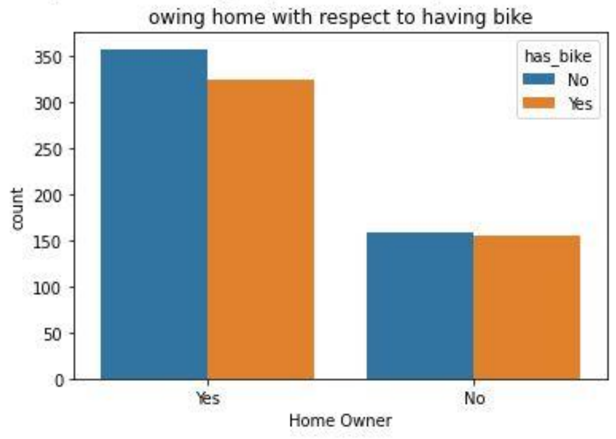


Fig 5.2.2.3

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

```
[ ] plt.title('Marital Status with respect to having bike')

    sns.countplot(df1['married'],hue = df1['has_bike'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Fu
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7473048350>
```

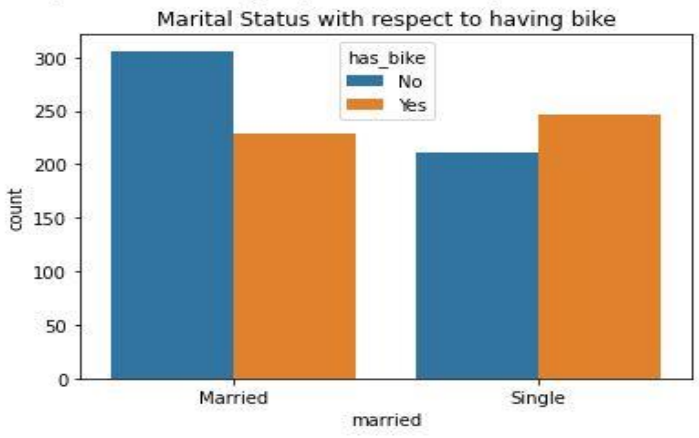
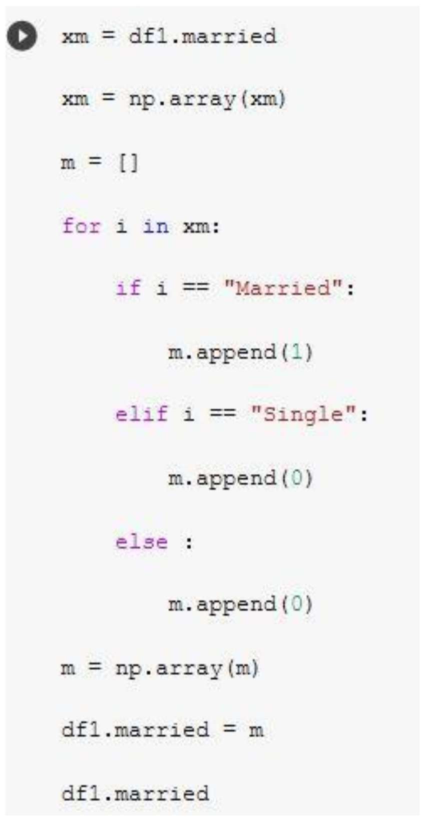


Fig 5.2.2.4

Fig 5.3.1: data cleaning

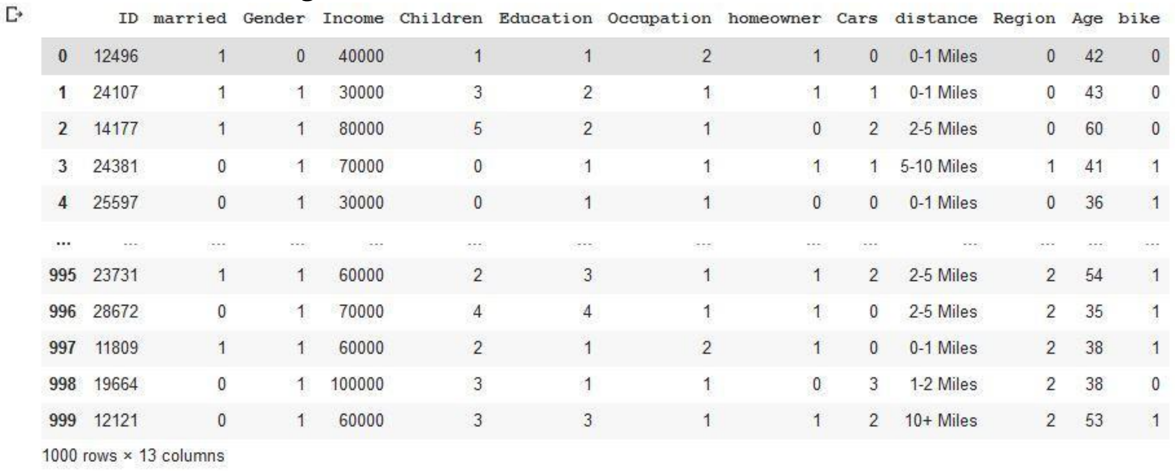
5.3 Data Cleaning:



```python
xm = df1.married

xm = np.array(xm)

m = []

for i in xm:

    if i == "Married":

        m.append(1)

    elif i == "Single":

        m.append(0)

    else :

        m.append(0)

m = np.array(m)

df1.married = m

df1.married
```

This type of process is done for all the relevant columns

I am Standardizing the features..



	ID	married	Gender	Income	Children	Education	Occupation	homeowner	Cars	distance	Region	Age	bike
0	12496	1	0	40000	1	1	2	1	0	0-1 Miles	0	42	0
1	24107	1	1	30000	3	2	1	1	1	0-1 Miles	0	43	0
2	14177	1	1	80000	5	2	1	0	2	2-5 Miles	0	60	0
3	24381	0	1	70000	0	1	1	1	1	5-10 Miles	1	41	1
4	25597	0	1	30000	0	1	1	0	0	0-1 Miles	0	36	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	23731	1	1	60000	2	3	1	1	2	2-5 Miles	2	54	1
996	28672	0	1	70000	4	4	1	1	0	2-5 Miles	2	35	1
997	11809	1	1	60000	2	1	2	1	0	0-1 Miles	2	38	1
998	19664	0	1	100000	3	1	1	0	3	1-2 Miles	2	38	0
999	12121	0	1	60000	3	3	1	1	2	10+ Miles	2	53	1

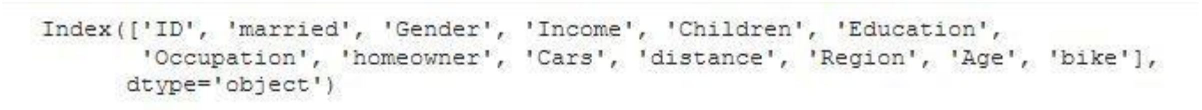
1000 rows × 13 columns

Fig 5.3.1: data cleaning

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



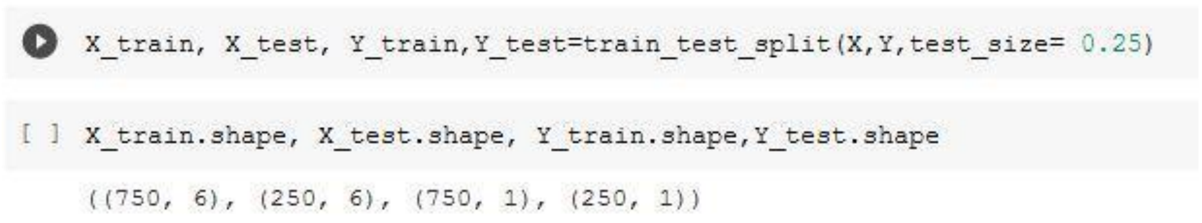
```
Index(['ID', 'married', 'Gender', 'Income', 'Children', 'Education',
       'Occupation', 'homeowner', 'Cars', 'distance', 'Region', 'Age', 'bike'],
      dtype='object')
```

=

Fig 6.1.1: splitting data

6.1.1 Train and Test Split:

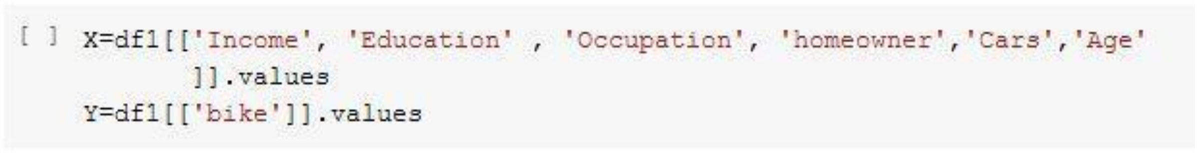
Splitting data into Train and Test



```
X_train, X_test, Y_train,Y_test=train_test_split(X,Y,test_size= 0.25)
```

```
X_train.shape, X_test.shape, Y_train.shape,Y_test.shape
```

```
((750, 6), (250, 6), (750, 1), (250, 1))
```



```
X=df1[['Income', 'Education' , 'Occupation', 'homeowner','Cars','Age'
       ]].values
Y=df1[['bike']].values
```

Fig 6.2.1: train & test split data

7 MODEL BUILDING AND EVALUATION

I Used 2 classifiers for predicting the output

- Logistic regression
- Random forest

7.1 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.

Regression:sum (y — 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.

Classification: G = sum (pk * (1 — pk))

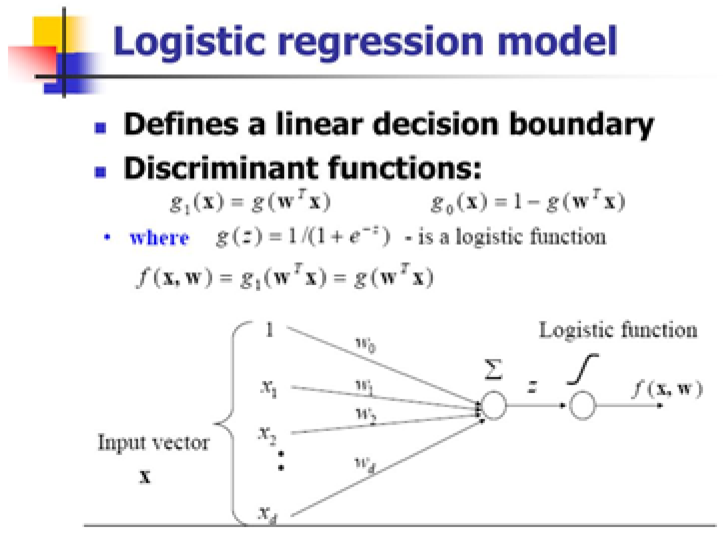
A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split. Here, pk is proportion of same class inputs present in a particular group. A perfect class purity occurs when a group contains all inputs from the same class, in which case pk is 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

7.2 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).



Logistic regression model

- Defines a linear decision boundary
- Discriminant functions:

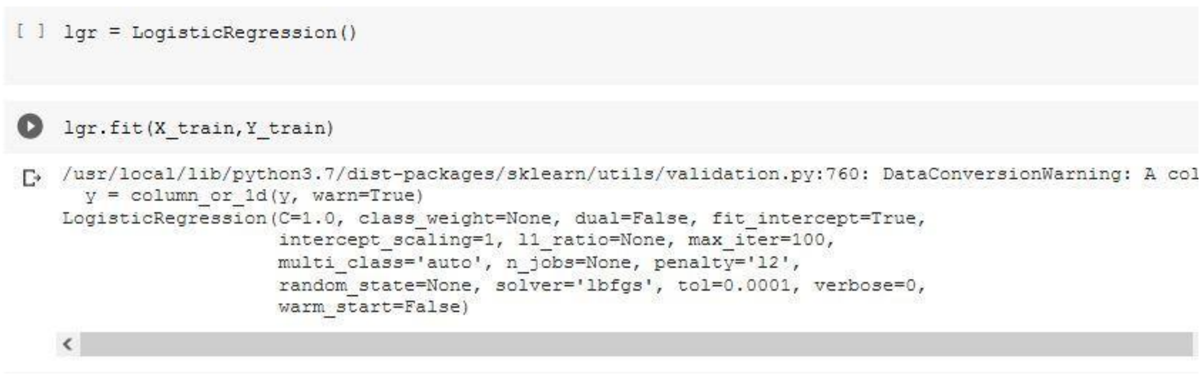
$g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$ $g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x})$

- where $g(z) = 1/(1+e^{-z})$ - is a logistic function

$f(\mathbf{x},\mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$

7.2.1 Train the Models

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```



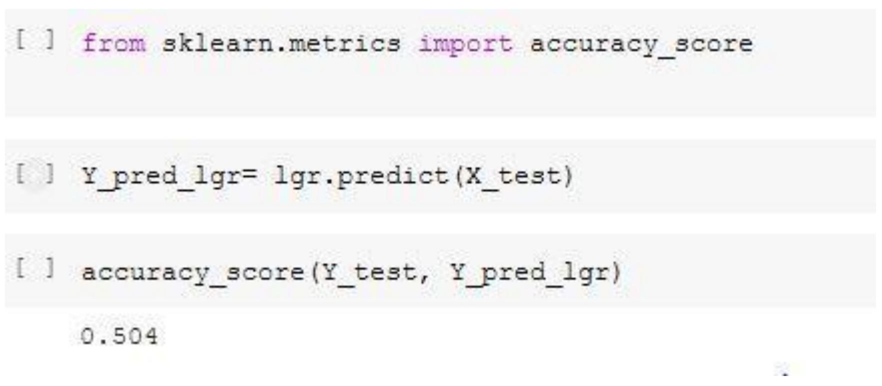
```
[ ] lgr = LogisticRegression()
```

```
lgr.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A col
  y = column_or_1d(y, warn=True)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig 7.1.4.1: logistic regression

7.2.2 METRICS Accuracy Score:



```
[ ] from sklearn.metrics import accuracy_score

[ ] Y_pred_lgr= lgr.predict(X_test)

[ ] accuracy_score(Y_test, Y_pred_lgr)

    0.504
```

Fig 7.1.5.1:logistic regression (accuracy score)

7.3 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.

7.3.1 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.

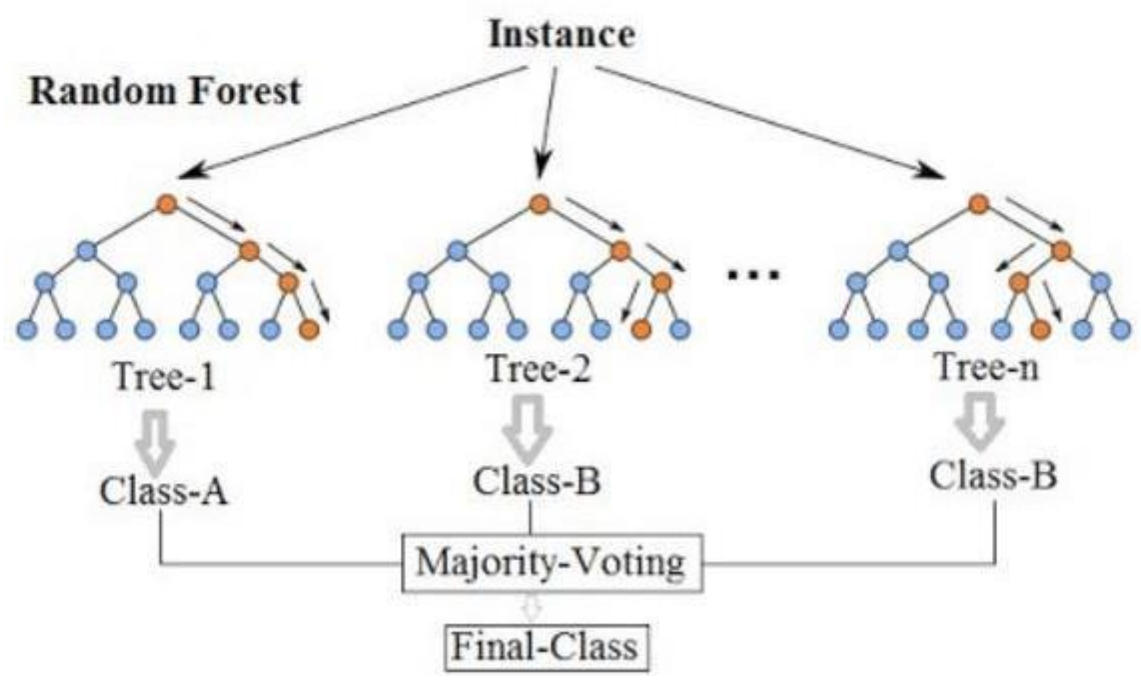
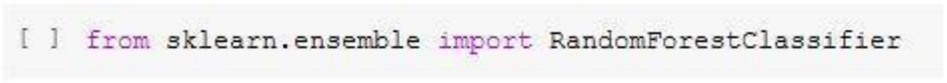


Fig7.2.1.1: Structure of Random Forest Classification

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.

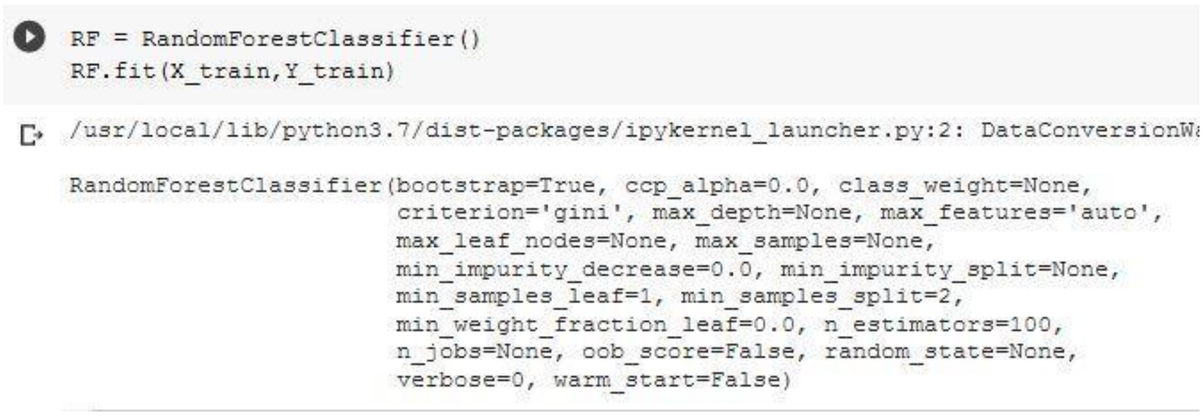
7.3.2 Model Creation:



```
[ ] from sklearn.ensemble import RandomForestClassifier
```

Fig7.2.1.4: random forest (model creation)

7.3.3 Model Fitting:

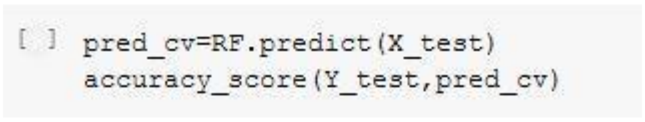


```
RF = RandomForestClassifier()
RF.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DataConversionWa

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Fig7.2.1.5: random forest (model fitting)

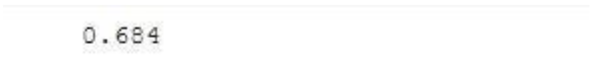
7.3.4 Evaluating the model:



```
[ ] pred_cv=RF.predict(X_test)
    accuracy_score(Y_test,pred_cv)
```

Fig 7.2.1.6: random forest (evaluating the model)

7.3.5 METRICS Accuracy Score:



```
0.684
```

Fig 7.2.1.7: random forest (accuracy score)

8 Conclusion

Through this project, the prediction of bike buyers over evaluation is made up of a data science model that can be used to predict the sale of a bike based on the customer's data, 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.