

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Engineering Degree in
Computer Science and Engineering

By:

A. MOHAMMED HASHIM

(Reg.No:40731063)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY JEPPIAAR NAGAR, RAJIV GANDHI
SALAI,
CHENNAI – 600119, TAMILNADU**

APRIL 2023



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI- 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
A. MOHAMMED HASHIM (40731063) who carried out the project
entitled “**ESTIMATING FORECAST USE OF A CITY BIKESHARE**
SYSTEM USING MACHINE LEARNING”, under my supervision from
June 2022 to November 2022.

Internal Guide
MS.DEVI, M.E.

Head of the Department
Dr.L.Lakshmanan M.E., Ph.D.
Dr.S.Vigneshwari,M.E.,Ph.D.

Submitted for Viva voce Examination held on

12-09-2022

Internal Examiner

External Examiner

DECLARATION

I, **A.MOHAMMED HASHIM** hereby declare that the project report entitled **“ESTIMATING FORECAST USE OF A CITY BIKESHARE SYSTEM USING MACHINE LEARNING”** done by me under the guidance of **MS . DEVI, M.E.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE: 12-09-2022



PLACE: Chennai

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean, School of Computing, Dr.S.Vigneshwari,M.E.,Ph.D. and Dr. L. Lakshmanan M.E.,Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **MS. DEVI,M.E.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

COGNIBOT LABS
CERTIFICATE
OF TRAINING

THE CERTIFICATE IS PRESENTED TO

Mohammad Hashim . A

40731063

from Sathyabama Institute of Science and Technology for successfully completing the 45 hours professional training
on Artificial Intelligence - 1 (Machine Learning) conducted by Cognibot Labs

Awarded on October 7, 2022



Hariharasudhan
Trainer - Cognibot Labs

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	(i)
	LIST OF FIGURES	(ii)
1	INTRODUCTION	12
	1.1 PROPOSED SYSTEM	12
	1.2 SYSTEM ARCHITECTURE	13
	1.3 IMPLEMENTATION	14
	1.4 DESCRIPTION OF MODULES	
	1.4.1 DOCK	15
	1.4.2 SERVOMOTOR	
	1.4.3 BLUETOOTH MODULE	
	1.4.4 GPS TRACKER	
	1.4.5 MOBILE APPLICATION	
	1.5 EXISTING SYSTEM	17
	1.6 OUTCOME	18
	1.7 ESTIMATING BIKESHARE SYSTEM DATASET	18
	1.8 COMMON MACHINE LEARNING ALGORITHMS AND GOALS	19
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	
	2.1 AIM	26
	2.2 SCOPE	26
	2.3 ATTRIBUTE INFORMATION	27

ABSTRACT

Machine Learning techniques give computers the capability to train and learn by using sample inputs and provide outputs which leads a model to test the test cases instead of being programmed. Lip reading recognition is a task of converting speech (lip movements) to readable format like text without audio. Lip reading recognition is a technology that helps for communication using devices. The uses of this concept are immense for the people who are who have hearing deficiency and vocally challenged. In situations where there is lot of noise, its very difficult to listen to each other. The noise around us can sometimes cause a huge impact on the conversation which may lead to miscommunication, the example can be a phone call in a theatre. In future, this problem can be solved by Lip Reading Recognition Technology. This technology helps one to transmit information without using vocal cords. This technology uses Lip Tracking which is a biometric system, it works based on a genuine system that can be developed using different levels of video processing, and hence it's possible to get lip contour and the exact location of key points in the subsequent frames which is usually referred as lip tracking. This technology works on the lip movement and generates frames where the other end person can easily get the result of it.

(i)

LIST OF FIGURES

Fig no	Fig title	Page no
1.1	Proposed system	13
1.2	Architecture diagram	13
1.3	Implementation of project	14
1.4.4	GPS tracker	16
1.4.5	Mobile application	17
1.7	Xcel sheet of given estimating bike share system dataset	19
1.8.1	Types of ml along the field of use	20
1.8.2	Linear regression scatter plot	21
1.8.3	Linear regression scatter plot with regression line	21
1.8.4	Logistic regression with probability output in s space	23
1.8.5	Example of decision tree	24
1.8.6	Random forest outliner	25
1.8.7	Scatter plot on k-means clustering	25
2.5.1	Code for reading the dataset using pandas	29
2.5.2	Imported bike sharing dataset in jupyter notebook	30
2.5.3	Using isnull() function checking for null values	30
3.2.1	Various python libraries for ml	34
3.2.2	Pandas libraries used to read	35

	dataset	
3.2.3	Numpy library for changing to Numpy array	35
3.2.4	Usage sklearn.train_test_split library	37
3.2.5	Example implementation of seaborn library	38
3.2.6	Plot using matplotlib Imported libraries in jupyter notebook	39
3.3.1	Given dataset	40
3.3.2	Applying decision trees	40
3.3.3	Training data	41
3.3.4	Testing data	42
3.4.1	Given data set	43
3.4.2	Applying random forest	43
3.4.3	Training data	44
3.4.4	Testing data	45

(ii)

CHAPTER-1

INTRODUCTION

1.1 LIP READING RECOGNITION

Lip Reading Recognition technology is mainly used for speech tracking by means converting the speech into understanding format by externally read the developments of the lips, face, and tongue. In lively situations, where the acknowledgment of any discussion is troublesome, which can be visual discourse acknowledgment that gives a successful way to get discourse to get better results. Lip reading can be difficult sometimes because of the different types' accents, speed of talking, facial highlights, skin shading, and so forth. However, there are a large group of utilizations, because of which this issue expects noteworthiness. It is extremely useful to remove the communication gap especially when it is communicated in a noisy place, and so forth. Lipreading is a difficult task for anybody without the knowledge, especially without any context of the concept.

Most lip-reading actions, besides the lips may sometimes make use of tongue and teeth to pronounce a word or letter which are latent and difficult to disambiguate without context. Lip reading is a complex issue because of different kind of lip shapes & color. Lipreading is similar to guesswork and will not be accurate because of pronunciation differences, bad position of faces, having hands over mouth, moustaches, too fast speech, low lighting, and beards etc.

The lip-reading recognition has been documented since long time, which includes the people with hearing deficiency use lip-reading method where it can be an adjustment to get and understand the fluent speech of the speaker. When it comes to converting these kinds of issues into automation there can be many difficult challenges comparing the ordinary recognition which can be audio. These audio speeches which has the pre-defined units are known as phonemes, mapping between the phonemes and the words are given by the pronunciation dictionaries.

And its easy to map video than audio as a result because of the compressed video rates are higher than uncompressed audio. So, it's a easy and efficient way to get the better results using video than the audio formats.

This paper investigates the task of speech recognition from video without audio. We present several neural network models with varying successes in this classification task. The input data to our algorithm is sequences of still images taken from frames of video footage. We use different neural network models to output one of 10 words that are spoken (or mouthed) by a face in the input images. We explore and combine a number of different models including CNNs, RNNs, and existing publically available pre-trained networks to assist in mouth recognition. This is an interesting learning task given that video traffic is growing at a high rate throughout the web, and this model could help extract data and process it to gain insights into the action or topics occurring in a video. Applications of a visual audio classifier range from play prediction in sporting events to profanity detection on social media sites to a live action lip reading mobile application.

In the past, research efforts have been far more focused on gesture recognition rather than visual speech recognition, making this for a new and exciting field to explore. There are a few existing systems and applications for lip reading, although most do not use neural networks but instead other machine learning techniques. More advanced visual speech recognition models such as Google's DeepMind LipNet network were published only a few months ago.

1.2 LITERATURE SURVEY

A neural network-based lip reading system is suggested in this study. The system lacks a language and relies only on visual clues. With only a few number of visemes to recognize as classes, the system is designed to lip read sentences with a wide variety of vocabulary and recognizewords that may not have been included in system training. The system was put through its paces on the difficult BBC Lip Reading Sentences 2 (LRS2) benchmark dataset.

Speech identification from visual-only recordings of a speaker's face is possible using a processing pipeline based only on neural networks, producing much higher accuracy than traditional approaches. Feed forward and recurrent neural network layers (specifically, Long Short-Term Memory (LSTM)) are layered to form a single structure that is trained by back-propagating error gradients across all layers. In this study, the performance of a stacked network was experimentally assessed and compared to that of a typical Support Vector Machine classifier using common computer vision characteristics.

They cover the key strategies for audio-visual speech recognition that have been developed during the last two decades in this chapter. They start with the visual feature extraction challenge and then go on to audio-visual fusion. In both cases, they describe some of the strategies used during the Johns Hopkins summer 2000 workshop (Neti et al., 2000). They also explore the issue of audio-visual speaker adaptation, which is critical for establishing systems across databases or building speaker-specific models. They next go over the primary audio-visual corpora that have been utilized in the literature for ASR trials, such as the IBM audio-visual LVCSR database. Following that, they report experimental findings on automated speech reading and audio-visual ASR. They address the challenge of automated recognition of damaged speech as an application of speaker adaptation. Finally, they wrap off the chapter with a review of the present status of audio-visual ASR and what they see as unresolved issues in this field.

CHAPTER-2

AIM AND SCOPE OF PRESENT INVESTIGATION

2.1 AIM :

To predict the “**ESTIMATING FORECAST USE OF A CITY BIKESHARE SYSTEM**” Data set using features from the given Data set.

2.2 SCOPE :

The aim of this project is to develop a demand forecast model for the Capital Bikeshare system . This is a model that for every day and hour, given some meteorological variables and whether the day is a holiday or not, attempts to predict the total number of bicycles that will be rented that hour.

In order to build the model, these steps have to be followed:

- Research methodologies that have been used to analyze problems of a similar nature.
- Acquire the required data to build the model.
- Analyze the data and find out which variables drive the demand.
- Design a predictive model to forecast the demand.
- Implement and tune the model to better fit the data.
- Draw conclusions from the previous work.

2.3 ATTRIBUTES INFORMATION :

Both hour.csv and day.csv have the following fields, except hr which is not available in day.csv

- instant: record index
- dteday : date

- season : season (1:winter, 2:spring, 3:summer, 4:fall)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)
- holiday : weather day is holiday or not (extracted from [\[Web Link\]](#))
- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
- + weathersit :
- 1: Clear, Few clouds, Partly cloudy, Partly cloudy
- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are derived via $(t_{\min})/(t_{\max}-t_{\min})$, $t_{\min}=-8$, $t_{\max}=+39$ (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius. The values are derived via $(t_{\min})/(t_{\max}-t_{\min})$, $t_{\min}=-16$, $t_{\max}=+50$ (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

2.4 MISSING ATTRIBUTES : (denoted by "?")

Attribute #: Number of Missing Values: (total: 0)

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0

2.5 DATA PREPROCESSING:

In this project as a backbone tool python is used to carry out machine learning concepts. With the help of a software called Anaconda Navigator, a jupyter notebook is launched where it is already installed along with the navigator. jupyter notebook is an open-source web application that allows to create and share documents and has live code and also visualization.

After importing the required libraries the dataset will be read in the note book with help of data frame (two-dimensional labeled data structure with columns of potentially different types) and read_csv(desired file type).

Reading the data set:

```
In [5]: df = pd.read_csv("train.csv")  
df.head()
```

Fig 2.5.1- code for reading the dataset using pandas

The dataset provided cannot always be a fully valued set, in that case we need to prepare the data in such a way the machine understands what is the value that has been entered.

Out[5]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

Fig 2.5.2 – Imported Bike sharing dataset in jupyter notebook

Check the values for null using isnull() function.

```
In [6]: df.isnull().sum()
Out[6]: datetime      0
        season        0
        holiday        0
        workingday     0
        weather        0
        temp           0
        atemp          0
        humidity        0
        windspeed       0
        casual         0
        registered     0
        count          0
        dtype: int64
```

Fig 2.5.3- using isnull() function checking for null values

The dataset is now ready for creating a machine learning model.

CHAPTER- 3

EXPERIMENTAL OR MATERIAL AND METHODS, ALGORITHMS USED

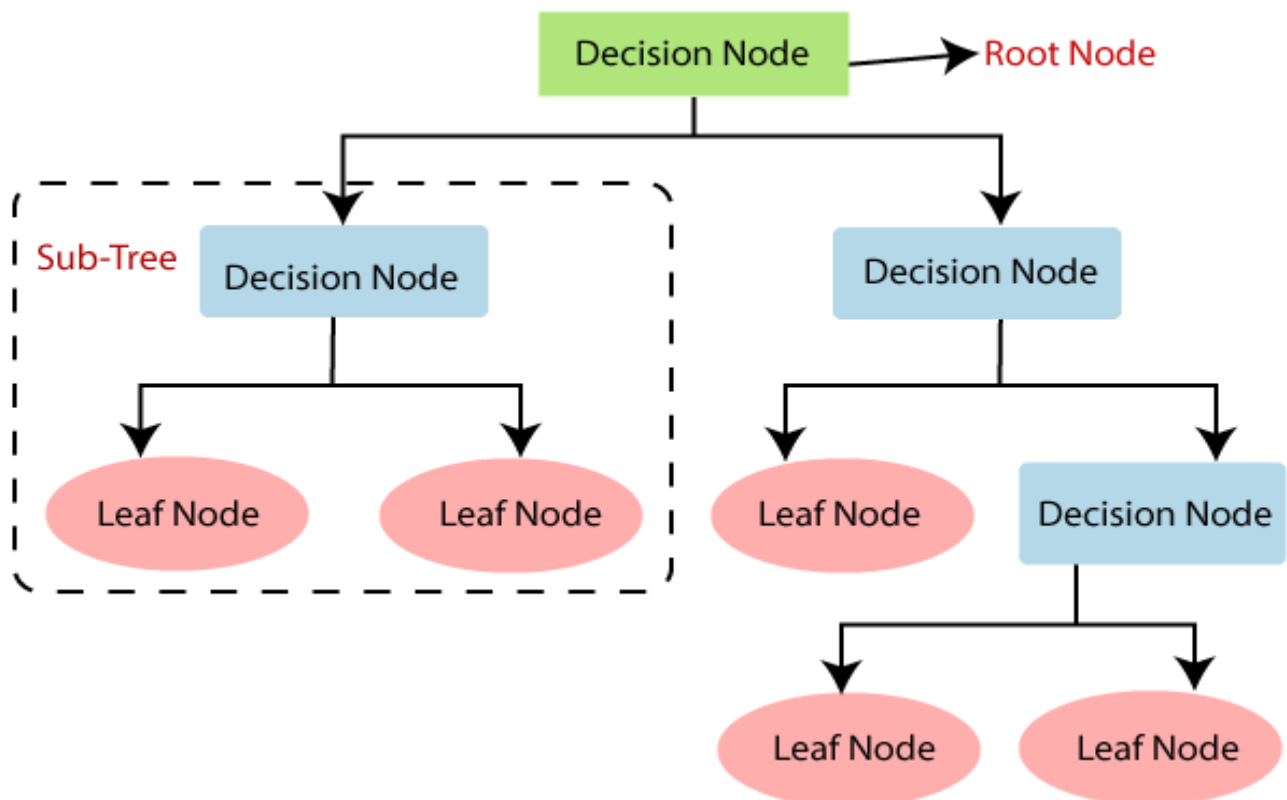
3.1 Types of classification algorithms used :

1.Decision Tree

2.Random Forest

3.1.1 Decision Tree :

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node**. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm**, which stands for **Classification and Regression Tree algorithm**.



3.1.2 Random Forest

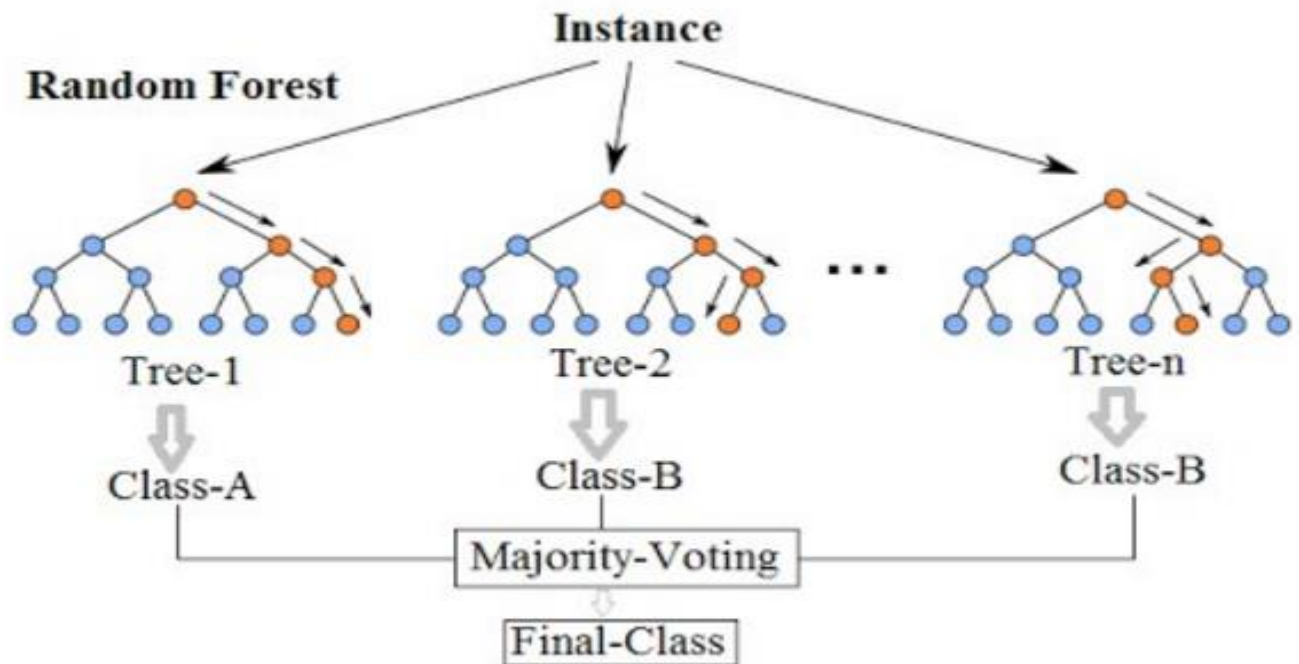
The Random Forest is also known as Decision Tree Forest. It is one of the popular **decision tree-based** ensemble models. The accuracy of these models is higher than other decision trees. This algorithm is used for both classification and regression applications.

In a random forest, we create a large number of decision trees, and in each decision tree, every observation is fed. The final output is the most common outcome for each observation. We take a majority vote for each classification model by feeding a new observation into all the trees.

An error estimate is made for cases that were not used when constructing the tree. This is called an **out-of-bag(OOB)** error estimate mentioned as a percentage.

The decision trees are prone to overfitting, and this is the main drawback of it. The reason is that trees, if deepened, are able to fit all types of variations in the data, including noise. It is possible to address this by partial pruning, and the results are often less than satisfactory

Random Forest Simplified



3.2 IMPORTED LIBRARIES

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let's look at the libraries that came in handy for mammographic dataset.

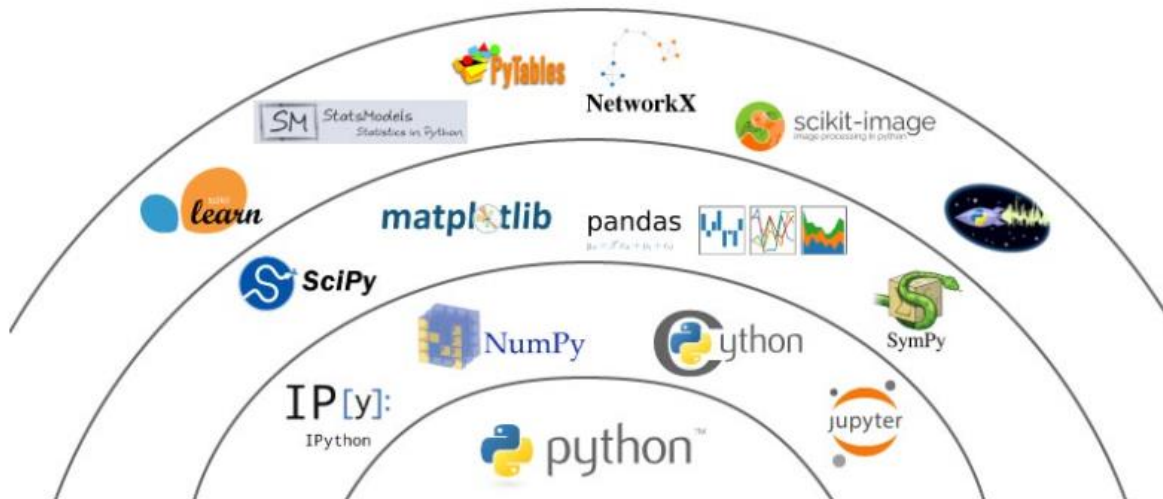


Fig 3.2.1- various python libraries for Machine Learning

LIBRARIES USED:

1. Pandas:

Pandas is a widely-used data analysis and manipulation library for python. It provides a lot of functions and methods that expedite the data analysis and preprocessing steps. It provides fast, flexible and expressive structures working with relational or labeled or both easy and intuitive. Considered as fundamental high-level building block in performing practical, real-world data analysis as in python. Has powerful tools like DataFrame and Series for analyzing.

```
df = pd.read_csv(r"C:\Users\Hi\Desktop\ObesityDataSet_raw_and_data_synthetic.csv")
df.head()
```

Fig 3.2.2- pandas library is used to read data sheet

2. Numpy:

Numpy stands for Numerical Python, is a library consisting of the one multidimensional array objects and a collection of countless of routines for processing those arrays. Using this mathematical and logical operations on arrays can be performed. The difference in using Numpy from pandas is, it works on numerical data whereas pandas on tabular data.

```
# changing the input_data to numpy array  
input_data_as_numpy_array = np.asarray(input_data)
```

Fig 3.2.3-Numpy library for changing to numpy array

3. sklearn:

sklearn stands for Scikit-learn, a machine learning library. It is imported for various classification, regression and clustering algorithms including k-means, random forest, support vector machines, gradient boosting and DBSCAN. It is designed using libraries Numpy and Scipy.

From the sklearn library and from the tree inside the library Decision Tree as Classifier. It is a class capable of performing multi-class classifier on a dataset. When compared with other classifiers, DecisionTreeClassifier takes input as two arrays: an array X, sparse or dense, of shape(n_samples,n_features) holding the training samples and an array Y of integer values, shape(n_samples), holding the class labels for training sample.

From sklearn another one called model_selection for training and testing the model imports train_test_split. It is a method setting a blueprint to analyze the data and the using it to measure new data. Selecting a proper model allows to

generate accurate results while making prediction. For proceeding, we need to train the model by using a specific dataset and test the model against another dataset.

By default, sklearn train_test_split will make random partitions for two subsets. We can also specify a random state for the operation. First, we need to split the dataset and then allocate the size for train and test. For this mammographic dataset we need train size as 70% (0.70) and test size as 30% (0.30) with the random state of 100.

```
X = df[['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight', 'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF',  
y = df['NObeyesdad']  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=100)
```

Fig 3.2.4- usage of sklearn.train_test_split library

4. Math:

Math is a built-in module that you we can use for mathematical tasks. It has set of methods and constants. It is a standard module in python an it is always available.

5. Seaborn:

Seaborn is a library built on top of matplotlib. It used for data visualization exploratory data analysis. They work easily with data frames and pandas is a library. The graphs created can also be customized easily. It provides default styles and color palettes to make statistical plots more attractive. Also closely integrated to the data structures from pandas. By using the seaborn library, we can easily represent our data on a plot.

This library is used to visualize our data; we do not need to take care of the internal details; we just have to pass our data set or data inside the relplot() function, and it calculate and place the value accordingly.

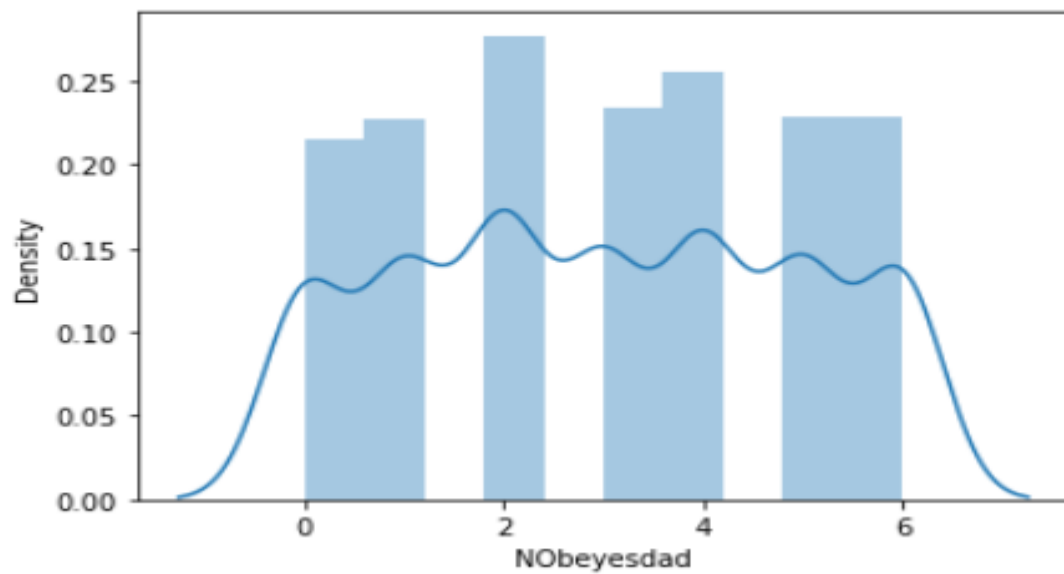
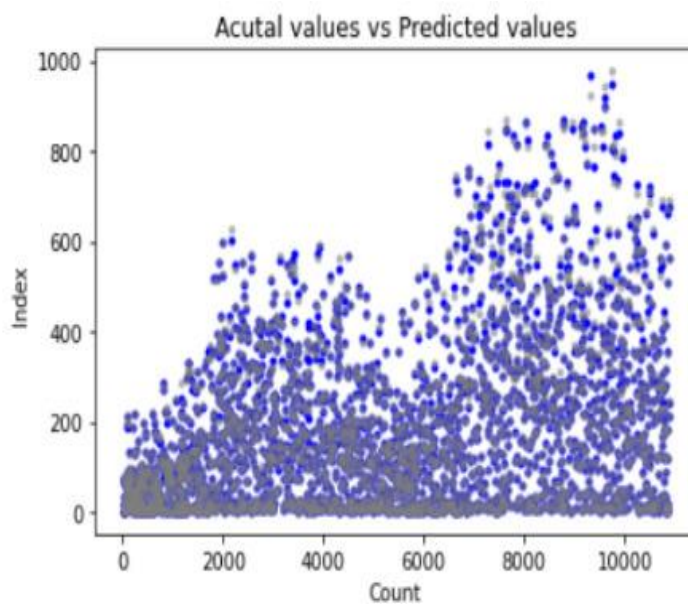


Fig 3.2.5- Example implementation of Seaborn library

6. Matplotlib.pyplot is a state-based interface to matplotlib. It provides MATLAB-like way of plotting. It makes changes to figure

```
Mean absolute error: 3.012123438648053
Mean absolute percentage error: 0.014032820781970849
R2 score: 0.9989414908098242
```



Mean absolute error: 1.385991917707568
Mean absolute percentage error: 0.006822891183174697
R2 score: 0.9996475560623459

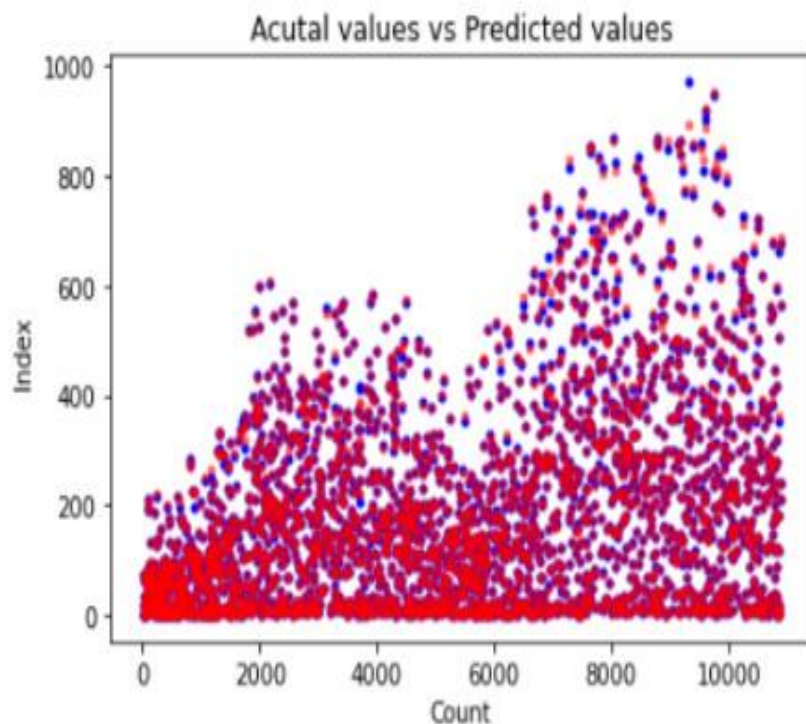


Fig 3.2.6- Plot using matplotlib

```
%matplotlib inline

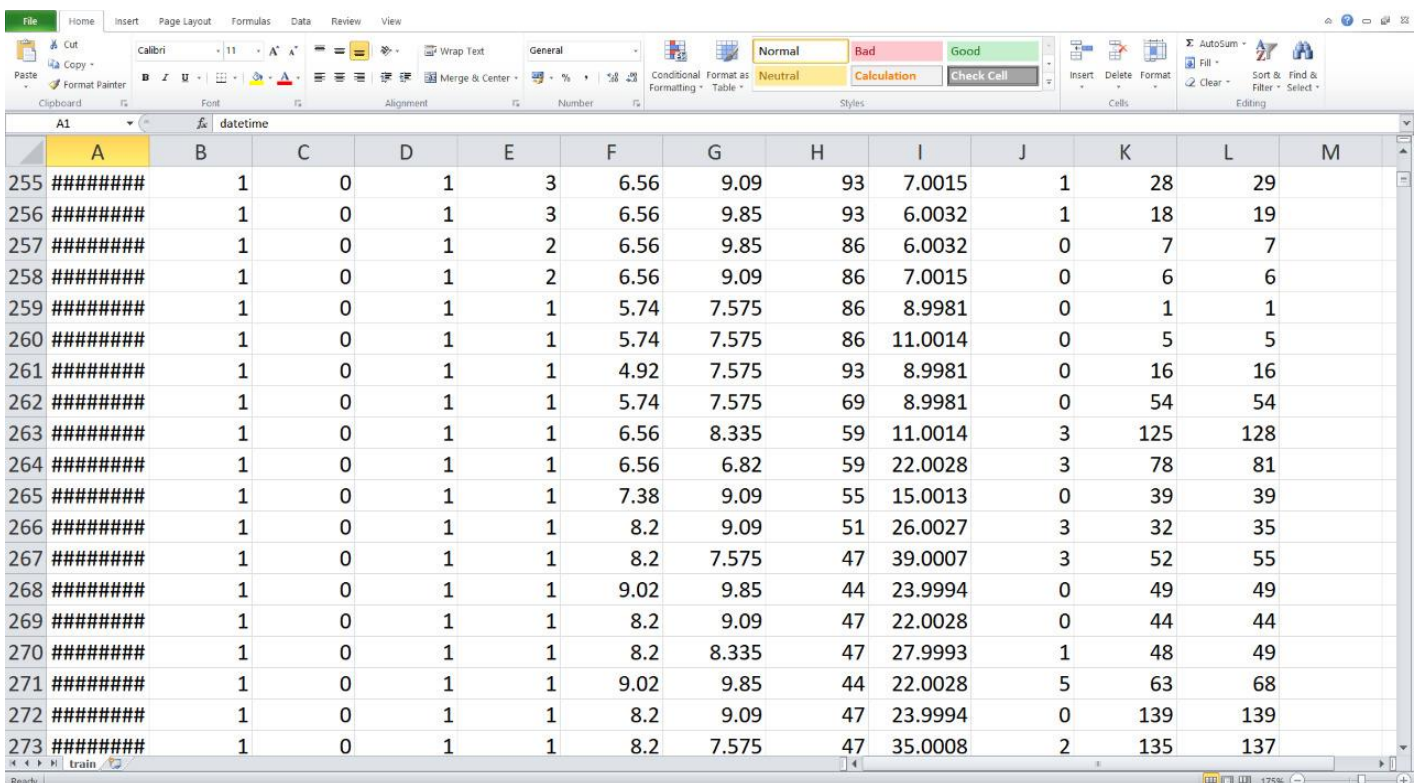
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, f1_score, plot_confusion_matrix
from math import sqrt
from sklearn import metrics
from sklearn import tree
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig 3.2.7- Imported libraries in jupyter notebook

3.3 Implementation of Decision Tree:

1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a Decision Tress model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions

Fig 3.3.1 Given Dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M
255	#####	1	0	1	3	6.56	9.09	93	7.0015	1	28	29	
256	#####	1	0	1	3	6.56	9.85	93	6.0032	1	18	19	
257	#####	1	0	1	2	6.56	9.85	86	6.0032	0	7	7	
258	#####	1	0	1	2	6.56	9.09	86	7.0015	0	6	6	
259	#####	1	0	1	1	5.74	7.575	86	8.9981	0	1	1	
260	#####	1	0	1	1	5.74	7.575	86	11.0014	0	5	5	
261	#####	1	0	1	1	4.92	7.575	93	8.9981	0	16	16	
262	#####	1	0	1	1	5.74	7.575	69	8.9981	0	54	54	
263	#####	1	0	1	1	6.56	8.335	59	11.0014	3	125	128	
264	#####	1	0	1	1	6.56	6.82	59	22.0028	3	78	81	
265	#####	1	0	1	1	7.38	9.09	55	15.0013	0	39	39	
266	#####	1	0	1	1	8.2	9.09	51	26.0027	3	32	35	
267	#####	1	0	1	1	8.2	7.575	47	39.0007	3	52	55	
268	#####	1	0	1	1	9.02	9.85	44	23.9994	0	49	49	
269	#####	1	0	1	1	8.2	9.09	47	22.0028	0	44	44	
270	#####	1	0	1	1	8.2	8.335	47	27.9993	1	48	49	
271	#####	1	0	1	1	9.02	9.85	44	22.0028	5	63	68	
272	#####	1	0	1	1	8.2	9.09	47	23.9994	0	139	139	
273	#####	1	0	1	1	8.2	7.575	47	35.0008	2	135	137	

Fig 3.3.2 Applying Decision Tree

```
In [13]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

```
In [14]: print(X_train)
```

	season	holiday	workingday	weather	temp	atemp	humidity	\
9639	4	0	0	1	22.96	26.515	83	
7284	2	0	1	1	21.32	25.000	77	
10463	4	0	0	2	11.48	13.635	93	
3079	3	0	0	1	26.24	30.305	69	
5630	1	0	1	3	9.02	13.635	87	
...	...	...	...	...	...	...	...	
2895	3	0	0	1	33.62	37.120	41	
7813	2	0	1	1	22.96	26.515	52	
905	1	0	1	1	8.20	10.605	75	
5192	4	0	0	1	13.12	15.150	57	
235	1	0	1	2	6.56	7.575	55	

	windspeed	casual	registered	time
9639	11.0014	37	154	0
7284	15.0013	21	218	21
10463	11.0014	37	135	9
3079	15.0013	4	8	5
5630	0.0000	5	147	17
...	...	...	...	...
2895	15.0013	167	249	13
7813	7.0015	22	171	22
905	8.9981	1	0	4
5192	16.9979	31	161	10
235	12.9980	0	3	2

[8164 rows x 11 columns]

```
In [23]: print(y_train)
```

9639	191
7284	239
10463	172
3079	12
5630	152
...	
2895	416
7813	193
905	1
5192	192
235	3

Name: count, Length: 8164, dtype: int64

Fig 3.3.3 Training data

In [24]: `print(X_test)`

	season	holiday	workingday	weather	temp	atemp	humidity	\
3709	3	1	0	2	28.70	33.335		74
6741	1	0	0	2	17.22	21.210		94
9848	4	0	0	1	26.24	31.060		44
1410	2	0	1	1	31.16	33.335		23
10672	4	0	1	2	16.40	20.455		66
...	...	...	...	...	...	...		...
5096	4	0	1	3	18.86	22.725		94
7007	2	0	1	1	13.94	17.425		71
6813	2	0	1	1	16.40	20.455		76
301	1	0	1	1	4.92	6.820		50
374	1	1	0	2	7.38	8.335		43

	windspeed	casual	registered	time
3709	11.0014	101	207	11
6741	11.0014	6	8	4
9848	12.9980	193	346	17
1410	36.9974	47	96	15
10672	22.0028	0	1	2
...	...	...	...	...
5096	19.0012	6	44	10
7007	7.0015	16	368	7
6813	32.9975	2	22	5
301	12.9980	0	14	0
374	16.9979	0	8	2

[2722 rows x 11 columns]

In [25]: `print(y_test)`

3709	308
6741	14
9848	539
1410	143
10672	1
...	
5096	50
7007	384
6813	24
301	14
374	8

Name: count, Length: 2722, dtype: int64

Fig 3.3.4 Testing data

3.4 Implementation of Random Forest :

1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a Random Forest model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions

	A	B	C	D	E	F	G	H	I	J	K	L	M
255	#####	1	0	1	3	6.56	9.09	93	7.0015	1	28	29	
256	#####	1	0	1	3	6.56	9.85	93	6.0032	1	18	19	
257	#####	1	0	1	2	6.56	9.85	86	6.0032	0	7	7	
258	#####	1	0	1	2	6.56	9.09	86	7.0015	0	6	6	
259	#####	1	0	1	1	5.74	7.575	86	8.9981	0	1	1	
260	#####	1	0	1	1	5.74	7.575	86	11.0014	0	5	5	
261	#####	1	0	1	1	4.92	7.575	93	8.9981	0	16	16	
262	#####	1	0	1	1	5.74	7.575	69	8.9981	0	54	54	
263	#####	1	0	1	1	6.56	8.335	59	11.0014	3	125	128	
264	#####	1	0	1	1	6.56	6.82	59	22.0028	3	78	81	
265	#####	1	0	1	1	7.38	9.09	55	15.0013	0	39	39	
266	#####	1	0	1	1	8.2	9.09	51	26.0027	3	32	35	
267	#####	1	0	1	1	8.2	7.575	47	39.0007	3	52	55	
268	#####	1	0	1	1	9.02	9.85	44	23.9994	0	49	49	
269	#####	1	0	1	1	8.2	9.09	47	22.0028	0	44	44	
270	#####	1	0	1	1	8.2	8.335	47	27.9993	1	48	49	
271	#####	1	0	1	1	9.02	9.85	44	22.0028	5	63	68	
272	#####	1	0	1	1	8.2	9.09	47	23.9994	0	139	139	
273	#####	1	0	1	1	8.2	7.575	47	35.0008	2	135	137	

Fig 3.4.1 Given Dataset

```
In [14]: from sklearn.model_selection import train_test_split
```

```
In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

Fig 3.4.2 Applying Random Forest

In [16]: `print(X_train)`

	season	holiday	workingday	weather	temp	atemp	humidity \
9639	4	0	0	1	22.96	26.515	83
7284	2	0	1	1	21.32	25.000	77
10463	4	0	0	2	11.48	13.635	93
3079	3	0	0	1	26.24	30.305	69
5630	1	0	1	3	9.02	13.635	87
...	...	...	...	...	...	...	...
2895	3	0	0	1	33.62	37.120	41
7813	2	0	1	1	22.96	26.515	52
905	1	0	1	1	8.20	10.605	75
5192	4	0	0	1	13.12	15.150	57
235	1	0	1	2	6.56	7.575	55

	windspeed	casual	registered	time
9639	11.0014	37	154	0
7284	15.0013	21	218	21
10463	11.0014	37	135	9
3079	15.0013	4	8	5
5630	0.0000	5	147	17
...	...	...	...	...
2895	15.0013	167	249	13
7813	7.0015	22	171	22
905	8.9981	1	0	4
5192	16.9979	31	161	10
235	12.9980	0	3	2

[8164 rows x 11 columns]

In [25]: `print(y_train)`

9639	191
7284	239
10463	172
3079	12
5630	152
...	
2895	416
7813	193
905	1
5192	192
235	3

Name: count, Length: 8164, dtype: int64

Fig 3.4.3 Training data

```
In [26]: print(X_test)
```

	season	holiday	workingday	weather	temp	atemp	humidity	\
3709	3	1	0	2	28.70	33.335	74	
6741	1	0	0	2	17.22	21.210	94	
9848	4	0	0	1	26.24	31.060	44	
1410	2	0	1	1	31.16	33.335	23	
10672	4	0	1	2	16.40	20.455	66	
...	...	...	...	...	...	...	...	...
5096	4	0	1	3	18.86	22.725	94	
7007	2	0	1	1	13.94	17.425	71	
6813	2	0	1	1	16.40	20.455	76	
301	1	0	1	1	4.92	6.820	50	
374	1	1	0	2	7.38	8.335	43	

	windspeed	casual	registered	time
3709	11.0014	101	207	11
6741	11.0014	6	8	4
9848	12.9980	193	346	17
1410	36.9974	47	96	15
10672	22.0028	0	1	2
...	...	...	...	...
5096	19.0012	6	44	10
7007	7.0015	16	368	7
6813	32.9975	2	22	5
301	12.9980	0	14	0
374	16.9979	0	8	2

[2722 rows x 11 columns]

```
In [27]: print(y_test)
```

3709	308
6741	14
9848	539
1410	143
10672	1
...	
5096	50
7007	384
6813	24
301	14
374	8

Name: count, Length: 2722, dtype: int64

Fig 3.4.4 Testing data

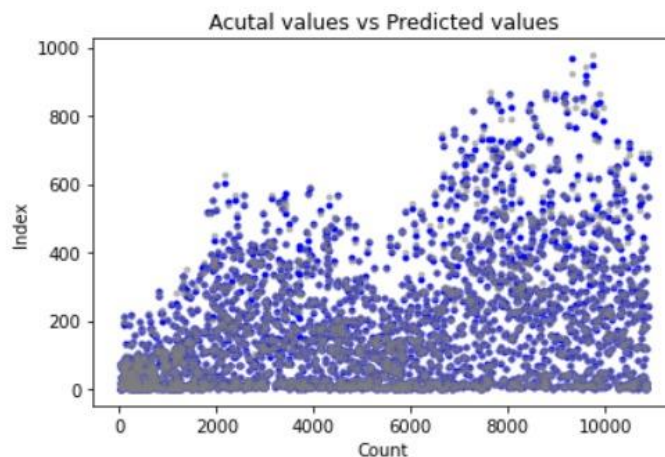
CHAPTER-4

4.1 RESULTS AND PERFORMANCE ANALYSIS

DECISION TREE RESULT :

```
In [18]: import matplotlib.pyplot as plt
```

```
In [19]: plt.scatter(X_test.index, y_test, marker=".", color="blue")
plt.scatter(X_test.index, pred, marker=".", color="gray", alpha=0.5)
plt.title("Acutal values vs Predicted values")
plt.xlabel("Count")
plt.ylabel("Index")
plt.show()
```



Decision Tree Performance Analysis :

```
In [35]: from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, r2_score
```

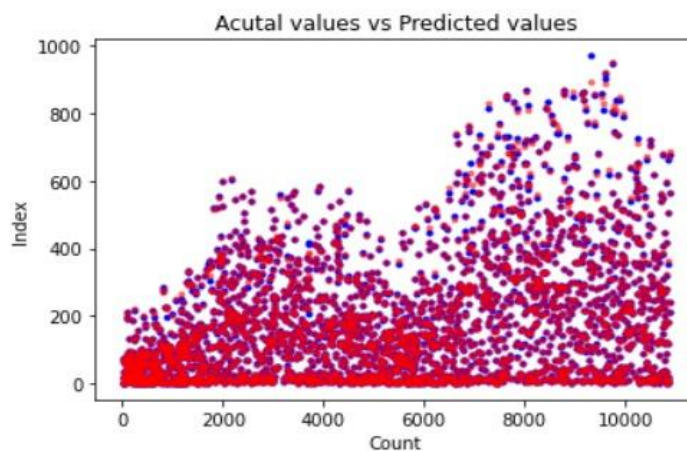
```
In [36]: mea = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
r2 = r2_score(y_test, pred)
print(f"""
Mean absolute error: {mea}
Mean absolute percentage error: {mape}
R2 score: {r2}
""")
```

```
Mean absolute error: 3.012123438648053
Mean absolute percentage error: 0.014032820781970849
R2 score: 0.9989414908098242
```

Random Forest Result :

```
In [121]: import matplotlib.pyplot as plt
```

```
In [129]: plt.scatter(X_test.index, y_test, marker=".", color="blue")
plt.scatter(X_test.index, pred, marker=".", color="red", alpha=0.5)
plt.title("Acutal values vs Predicted values")
plt.xlabel("Count")
plt.ylabel("Index")
plt.show()
```



Random Forest Performance Analysis:

```
In [124]: from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, r2_score
```

```
In [125]: mea = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
r2 = r2_score(y_test, pred)
print(f"""
Mean absolute error: {mea}
Mean absolute percentage error: {mape}
R2 score: {r2}
""")
```

```
Mean absolute error: 1.385991917707568
Mean absolute percentage error: 0.006822891183174697
R2 score: 0.9996475560623459
```

CHAPTER-5

5.1 SUMMARY AND CONCLUSIONS

Bicycle sharing has been the greatest growth of any transport initiative in recent years. When you look at the future, there are obvious innovations that are certain to take place. How and when those will happen, however, is more difficult to predict the future. Optimizing the fare structure will expand user access to a wider range of users, as well as increase the overall cost recovery by use. It will also serve to make the fare structure more transparent to users, lowering complaints and increasing customer satisfaction, which is important to potential sponsors, to the City and to the operator.” This is not a bad idea but might be hard to implement in cities where the free period is entrenched in the concept of bicycle sharing. Cohen further states that, however the new bundling of trip fares is done, there still must be the usage fee after a certain period of time to motivate users to return the bicycles. The use of collection of data and analysis to monitor and optimize bicycle sharing systems has only scratched only the surface. As with all transport, bicycle sharing will continue to embrace technology that will make system design, innovative revenue streams and integration easier and cheaper. E-bikes and scooters will definitely find their way into bicycle sharing systems or use bicycle sharing models to create their own system. Work is currently being undertaken in developing the hardware and associated operational support to handle charging both at the station and at a central location. An area of potential huge growth is whether e-bike sharing can replace private scooter trips in Asian cities, where scooter use can be up to 75 percent. If it could, the environmental and social benefits could be enormous, as governments could then control emissions, speeds and parking issues of significant importance.

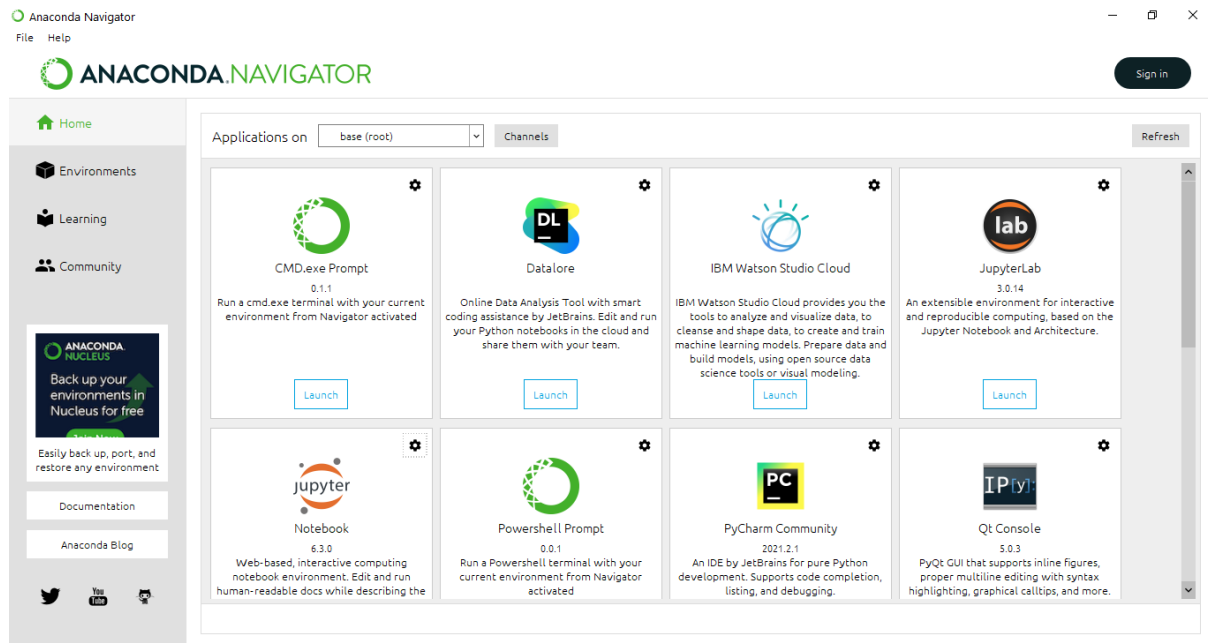
REFERENCES

- [1] Bike-Sharing Programs Hit the Streets in Over 500 Cities Worldwide; Earth Policy Institute; Larsen, Janet; 25 April 2013
- [2] "In Focus: The Last Mile and Transit Ridership". Institute for Local Government. January 2011.
- [3] James May (21 October 2010). "May, James, "Cycling Proficiency With James May", The Daily Telegraph, 21 October 2010". London: Telegraph.co.uk. Retrieved 2012-01-15.
- [4] Free City Bike Schemes, Søren B. Jensen, City of Copenhagen, Conference Proceedings, Amsterdam 2000
- [5] "Bike sharing: Research on health effects, helmet use and equitable access". Journalist's Resource. 2018-04-05. Retrieved 2018-09-06.
- [6] "Mobiped". Mobiped. 11 October 2010. Archived from the original on 22 September 2011. Retrieved 2012-01-15.
- [7] Bieler, Andrew (December 2008). "Bikes as a Public Good: What is the Future of Public Bike Sharing in Toronto"(PDF). Retrieved 1 March 2018.
- [8] "The Sharing Economy is Learning How to Ride a Bike". Energy Institute Blog. 2018-02-20. Retrieved 2018-03-07.
- [9] Taylor, Alan (22 March 2018). "The Bike-Share Oversupply in China: Huge Piles of Abandoned and Broken Bicycles". The Atlantic. Retrieved 16 April 2018.
- [10] "The Bike-sharing World Map is 10 years old". Bike- Sharing.blogspot.com. Retrieved 25 May 2018.
- [11] "Bike Sharing Program Rolls on Campus during Summer". Georgia Tech News Room.

APPENDIX

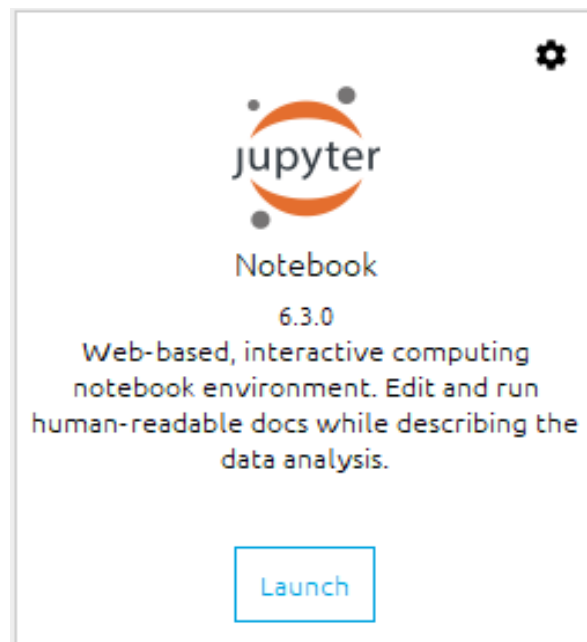
WORKING ENVIRONMENT

ANACONDA NAVIGATOR is desktop GUI used to launch applications and also manage packages in one place. Outlook



CODING ENVIRONMENT

Jupyter notebook from the anaconda navigator is launched along with all the preinstalled packages for python.



Screenshots and Outputs

Decision Tree Algorithm

jupyter PT1 Decision tree Last Checkpoint: 15 minutes ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O

Run Code

```
In [21]: import pandas as pd
```

```
In [22]: df = pd.read_csv("train.csv")
df.head()
```

```
Out[22]:
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
In [23]: df.isnull().sum()
```

```
Out[23]: datetime    0
season          0
holiday          0
workingday       0
weather          0
temp            0
atemp           0
humidity         0
windspeed       0
casual          0
registered       0
count           0
dtype: int64
```

```
In [24]: print(f"""
Shape: {df.shape}
Tot size: {df.size}
""")
```

```
In [24]: print(f"""
Shape: {df.shape}
Tot size: {df.size}
""")
```

```
Shape: (10886, 12)
Tot size: 130632
```

```
In [25]: def convertDatetime(x):
time = x[11:]
hour = time[:2]
hour = int(hour)
return hour

df["time"] = df["datetime"].apply(convertDatetime)
df.head()
```

```
Out[25]:
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	time
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4

```
In [26]: df = df.drop("datetime", axis=1)
df.head()
```

```
Out[26]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2

```
Out[26]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4

Create Model

```
In [27]: x = df.drop("count", axis=1)
y = df["count"]
```

```
In [28]: x.head()
```

```
Out[28]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	4

```
In [29]: y.head()
```

```
Out[29]:
```

0	16
1	40
2	32
3	13
4	1

Name: count, dtype: int64

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
Out[29]: 0    16
         1    40
         2    32
         3    13
         4     1
         Name: count, dtype: int64
```

```
In [30]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

```
In [31]: print(f"""
         X_train size: {X_train.shape}
         X_test size: {X_test.shape}

         y_train size: {len(y_train)}
         y_test size: {len(y_test)}
         """)
```

```
X_train size: (8164, 11)
X_test size: (2722, 11)
```

```
y_train size: 8164
y_test size: 2722
```

```
In [32]: from sklearn.tree import DecisionTreeRegressor
```

```
In [33]: model = DecisionTreeRegressor()
         model.fit(X_train, y_train)
         pred = model.predict(X_test)
```

```
In [34]: print(f"""
         Size of pred: {len(pred)}
         """)
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [34]: print(f"""
         Size of pred: {len(pred)}
         """)
```

```
Size of pred: 2722
```

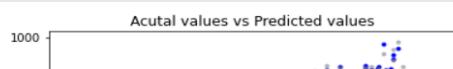
```
In [35]: from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, r2_score
```

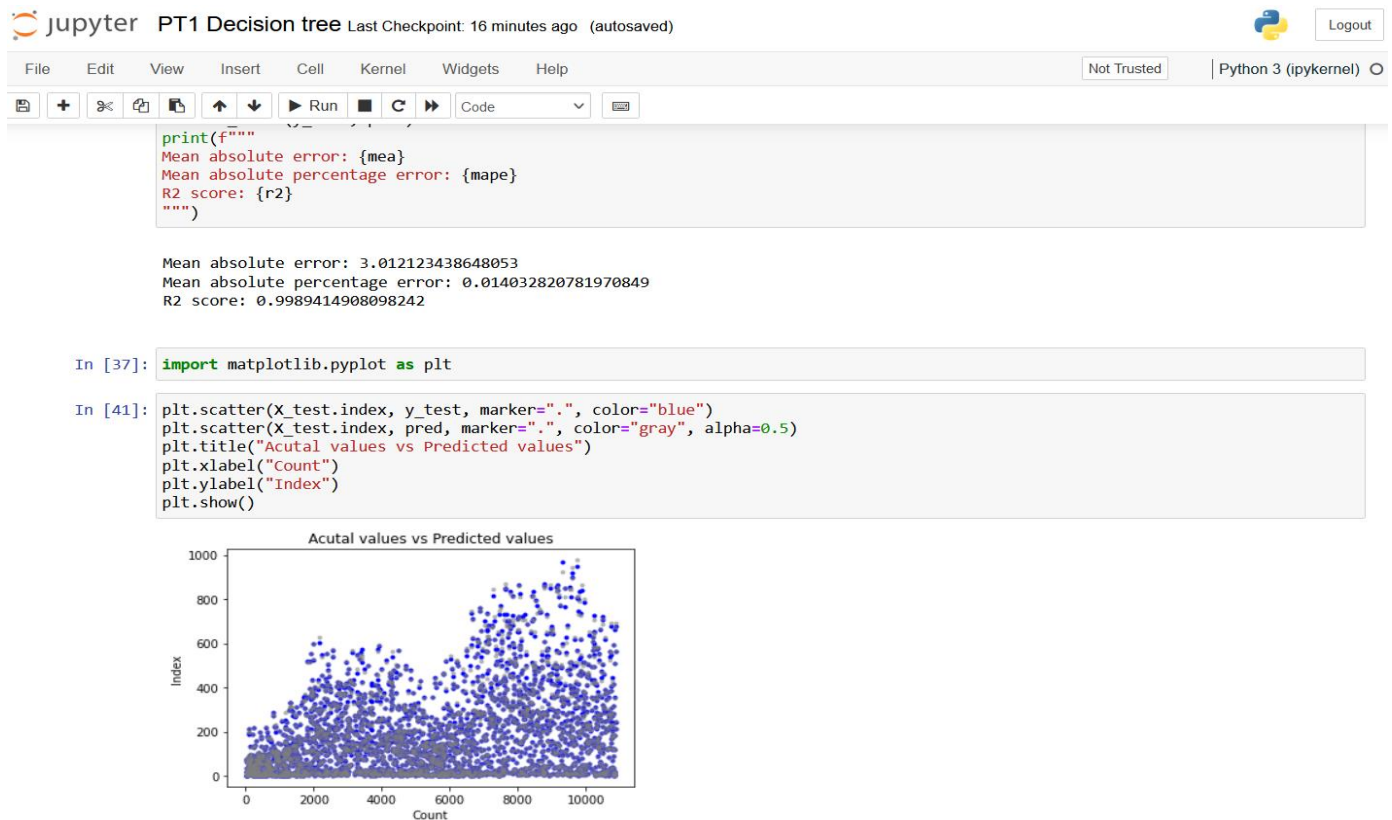
```
In [36]: mea = mean_absolute_error(y_test, pred)
         mape = mean_absolute_percentage_error(y_test, pred)
         r2 = r2_score(y_test, pred)
         print(f"""
         Mean absolute error: {mea}
         Mean absolute percentage error: {mape}
         R2 score: {r2}
         """)
```

```
Mean absolute error: 3.012123438648053
Mean absolute percentage error: 0.014032820781970849
R2 score: 0.9989414908098242
```

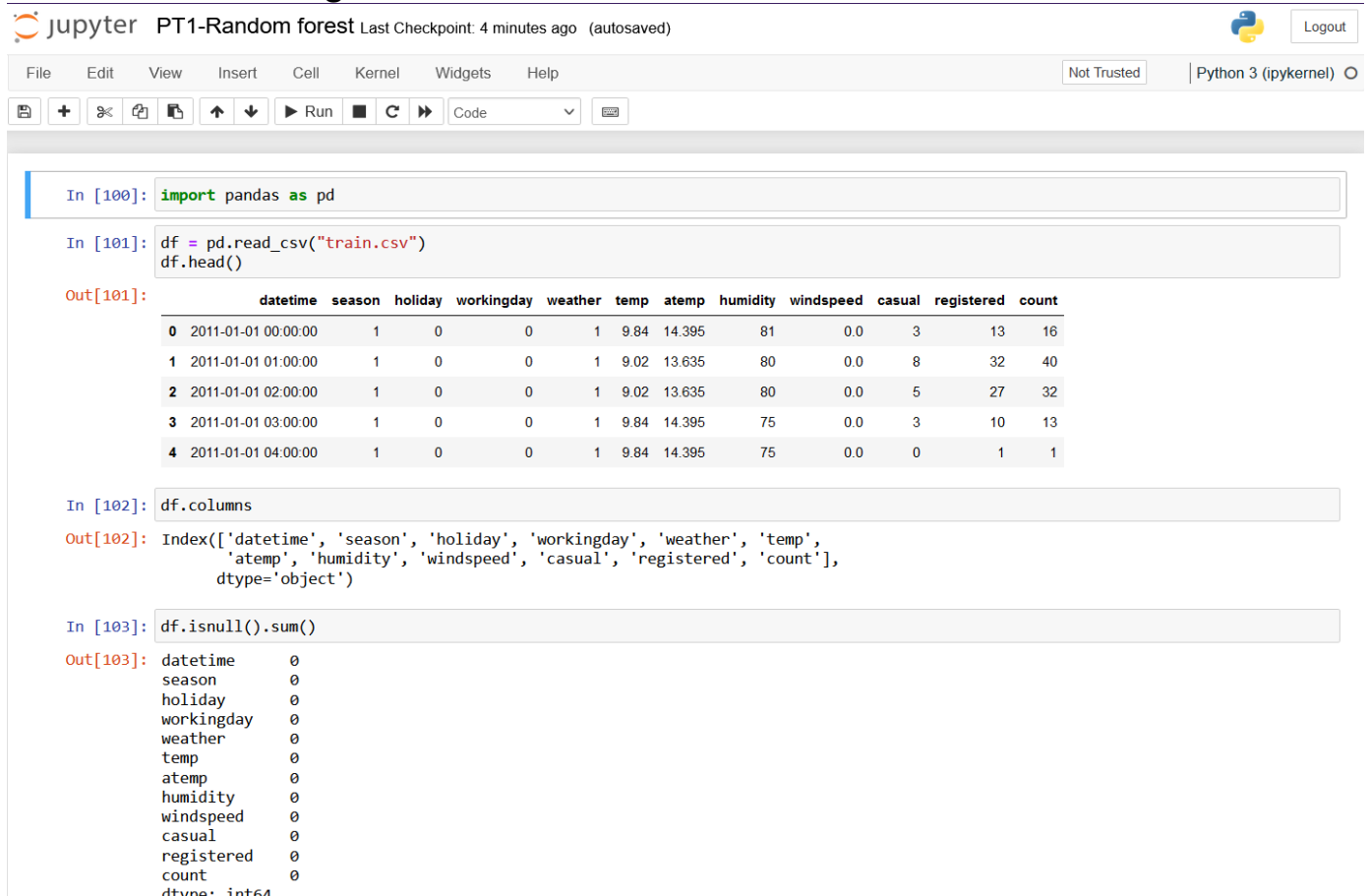
```
In [37]: import matplotlib.pyplot as plt
```

```
In [41]: plt.scatter(X_test.index, y_test, marker=".", color="blue")
         plt.scatter(X_test.index, pred, marker=".", color="gray", alpha=0.5)
         plt.title("Acutal values vs Predicted values")
         plt.xlabel("Count")
         plt.ylabel("Index")
         plt.show()
```





Random Forest Algorithm



```
Out[103]: datetime    0
season          0
holiday         0
workingday      0
weather         0
temp            0
atemp           0
humidity        0
windspeed       0
casual          0
registered      0
count           0
dtype: int64
```

```
In [104]: df.describe()
```

```
Out[104]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.980477	151.039033	181.144454
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

```
In [105]: print(f"""
Shape: {df.shape}
Tot size: {df.size}
""")
```

```
Shape: (10886, 12)
```

```
In [105]: print(f"""
Shape: {df.shape}
Tot size: {df.size}
""")
```

```
Shape: (10886, 12)
Tot size: 130632
```

```
In [106]: df["datetime"][0][11:]
```

```
Out[106]: '00:00:00'
```

```
In [107]: def convertDatetime(x):
time = x[11:]
hour = time[:2]
hour = int(hour)
return hour

df["time"] = df["datetime"].apply(convertDatetime)
# df["time"] = df["datetime"].apply(lambda x: int(x[11:13]))
df.head()
```

```
Out[107]:
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	time
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4

```
In [108]: df = df.drop("datetime", axis=1)
df.head()
```

```
Out[108]:
```

```
In [108]: df = df.drop("datetime", axis=1)
df.head()
```

```
Out[108]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4

Create model

```
In [109]: df.head()
```

```
Out[109]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	4

```
In [110]: X = df.drop("count", axis=1)
y = df["count"]
```

```
In [111]: X.head()
```

```
Out[111]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	4

```
Out[111]:
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	time
0	1	0	0	1	9.84	14.395	81	0.0	3	13	0
1	1	0	0	1	9.02	13.635	80	0.0	8	32	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	2
3	1	0	0	1	9.84	14.395	75	0.0	3	10	3
4	1	0	0	1	9.84	14.395	75	0.0	0	1	4

```
In [112]: y.head()
```

```
Out[112]:
```

0	16
1	40
2	32
3	13
4	1

Name: count, dtype: int64

```
In [113]: from sklearn.model_selection import train_test_split
```

```
In [114]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

```
In [115]: print(f"""
X_train size: {X_train.shape}
X_test size: {X_test.shape}

y_train size: {len(y_train)}
y_test size: {len(y_test)}
""")
```

```
X_train size: (8164, 11)
X_test size: (2722, 11)
```

```
y_train size: 8164
y_test size: 2722
```

```
X_train size: (8164, 11)
X_test size: (2722, 11)
```

```
y_train size: 8164
y_test size: 2722
```

```
In [116]: from sklearn.ensemble import RandomForestRegressor
```

```
In [117]: model = RandomForestRegressor()
model.fit(X_train, y_train)
pred = model.predict(X_test)
```

```
In [118]: print(f"""
Size of pred: {len(pred)}
""")
```

```
Size of pred: 2722
```

```
In [124]: from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, r2_score
```

```
In [125]: mea = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
r2 = r2_score(y_test, pred)
print(f"""
Mean absolute error: {mea}
Mean absolute percentage error: {mape}
R2 score: {r2}
""")
```

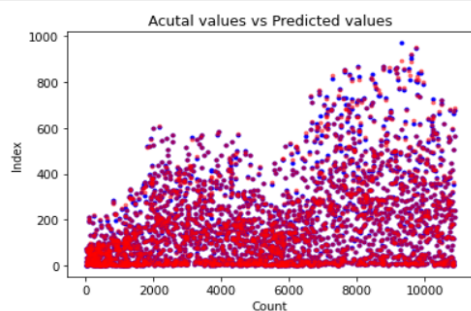
```
Mean absolute error: 1.385991917707568
Mean absolute percentage error: 0.006822891183174697
R2 score: 0.9996475560623459
```

```
r2 = r2_score(y_test, pred)
print(f"""
Mean absolute error: {mea}
Mean absolute percentage error: {mape}
R2 score: {r2}
""")
```

```
Mean absolute error: 1.385991917707568
Mean absolute percentage error: 0.006822891183174697
R2 score: 0.9996475560623459
```

```
In [121]: import matplotlib.pyplot as plt
```

```
In [129]: plt.scatter(X_test.index, y_test, marker=".", color="blue")
plt.scatter(X_test.index, pred, marker=".", color="red", alpha=0.5)
plt.title("Acutal values vs Predicted values")
plt.xlabel("Count")
plt.ylabel("Index")
plt.show()
```



SOURCE CODE :

Decision Tree:

```
import pandas as pd
df = pd.read_csv("train.csv")
df.head()
df.isnull().sum()
print(f"""
Shape: {df.shape}
Tot size: {df.size}
""")
def convertDatetime(x):
    time = x[11:]
    hour = time[:2]
    hour = int(hour)
    return hour

df["time"] = df["datetime"].apply(convertDatetime)
df.head()
df = df.drop("datetime", axis=1)
df.head()
X = df.drop("count", axis=1)
y = df["count"]
X.head()
y.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
print(X_train)
print(y_train)
print(X_test)
print(y_test)
print(f"""
X_train size: {X_train.shape}
```

```
X_test size: {X_test.shape}
```

```
y_train size: {len(y_train)}
```

```
y_test size: {len(y_test)}
```

```
"""
```

```
from sklearn.tree import DecisionTreeRegressor
```

```
model = DecisionTreeRegressor()
```

```
model.fit(X_train, y_train)
```

```
pred = model.predict(X_test)
```

```
print(f"""
```

```
Size of pred: {len(pred)}
```

```
""")
```

```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,
```

```
r2_score
```

```
mea = mean_absolute_error(y_test, pred)
```

```
mape = mean_absolute_percentage_error(y_test, pred)
```

```
r2 = r2_score(y_test, pred)
```

```
print(f"""
```

```
Mean absolute error: {mea}
```

```
Mean absolute percentage error: {mape}
```

```
R2 score: {r2}
```

```
""")
```

```
import matplotlib.pyplot as plt
```

```
plt.scatter(X_test.index, y_test, marker=".", color="blue")
```

```
plt.scatter(X_test.index, pred, marker=".", color="gray", alpha=0.5)
```

```
plt.title("Actual values vs Predicted values")
```

```
plt.xlabel("Count")
```

```
plt.ylabel("Index")
```

```
plt.show()
```

Random Forest:

```
import pandas as pd
df = pd.read_csv("train.csv")
df.head()
df.columns
df.isnull().sum()
df.describe()
print(f"""
Shape: {df.shape}
Tot size: {df.size}
""")
df["datetime"][0][11:]
def convertDatetime(x):
    time = x[11:]
    hour = time[:2]
    hour = int(hour)
    return hour

df["time"] = df["datetime"].apply(convertDatetime)
# df["time"] = df["datetime"].apply(lambda x: int(x[11:13]))
df.head()
df = df.drop("datetime", axis=1)
df.head()
df.head()
X = df.drop("count", axis=1)
y = df["count"]
X.head()
y.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
print(X_train)
print(y_train)
print(X_test)
```

```

print(y_test)
print(f"""
X_train size: {X_train.shape}
X_test size: {X_test.shape}

y_train size: {len(y_train)}
y_test size: {len(y_test)}
""")
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train, y_train)
pred = model.predict(X_test)
print(f"""
Size of pred: {len(pred)}
""")
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,
r2_score
mea = mean_absolute_error(y_test, pred)
mape = mean_absolute_percentage_error(y_test, pred)
r2 = r2_score(y_test, pred)
print(f"""
Mean absolute error: {mea}
Mean absolute percentage error: {mape}
R2 score: {r2}
""")
import matplotlib.pyplot as plt
plt.scatter(X_test.index, y_test, marker=".", color="blue")
plt.scatter(X_test.index, pred, marker=".", color="red", alpha=0.5)
plt.title("Actual values vs Predicted values")
plt.xlabel("Count")
plt.ylabel("Index")
plt.show()

```