A Mini-Project Report On

"Prediction Of Rented Bikes"

Submitted
By

Student Name:-
Abhishek
Chaudhari
PRN: 1132200066

Student Name:-
Anand Bang
PRN: 1132200110

Student Name:-
Aditya Kulkarni
PRN: 1132200415

Student Name:-
Saurabh
Deshpande
PRN: 1132200125

F.Y. M.Sc. (Data Science and Big Data Analytics)

School of Computer Science
Faculty of Science
MIT – World Peace University
Pune - 411038

Academic Year 2020-2021
MAY – 2021

MIT WORLD PEACE UNIVERSITY, PUNE
SCHOOL OF COMPUTER SCIENCE

Certificate

This is to certify that

Anand Bang

PRN: 1132200110

Of ***M.Sc. (Data Science and Big Data Analytics)*** successfully completed his/her Mini-Project in

"Prediction Of Rented Bikes"

to our satisfaction and submitted the same during the academic year 2019-2020 towards the partial fulfilment of degree of **Master of Science in Data Science and Big Data Analytics** of MIT World Peace University under the School of Computer Science, MIT WPU, Pune.

Prof. Dr. Shubhalaxmi Joshi **Associate Dean** **Faculty of Science** **MITWPU**	**Prof. Prajakta Soman** **Program Head** **School of Computer Science** **MIT WPU**	**Prof. Surabhi Thatte** **Assistant Professor** **School of Computer Science** **MIT WPU**

ACKNOWLEDGEMENT

In the accomplishment of this project I would like to express my special thanks of gratitude to my teachers Mrs. Surabhi Thatte, School of Computer Science, MIT World Peace University whose valuable guidance has been the ones that helped me patch this project and make it full proof success. Their suggestions and instructions have served as the major contributor towards the completion of the project.

As we were working in a group, I would like to thank my group members for their fabulous support throughout the completion of the project. We learnt a lot of things during this period as it were hard to work in this time of adversity; we were in touch with each other throughout the period and shared everything which was important from the aspect of our project. As this project was completed by staying at home I would also like to thank our families for their co-operation and for providing facilities to us.

Anand Bang
PRN: 1132200110

Contents

INTRODUCTION

Currently rental bikes are introduced in many urban cities for the enhancement of mobility comfort. The purpose of this movement is to modernize cities and encourage people to head to a green world. For example, in Paris 2007, "velibs" were introduced, and in Amsterdam, there are more bikes than cars. The goal is to facilitate commuting in the city and reduce the amount of cars and the pollution. Indeed, the development of the way to commute reduced the use of cars to go to
work and visit the city.

DOMAIN

A **bike-sharing system**, **public bicycle scheme**, or **public bike share** (PBS) scheme, is a shared transport service in which bikes are made available for shared use to individuals on a short term basis for a price or free. Many bike share systems allow people to borrow a bike from a "dock" and return it at another dock belonging to the same system. Docks are special bike racks that lock the bike, and only release it by computer control. The user enters payment information, and the computer unlocks a bike. The user returns the bike by placing it in the dock, which locks it in place

MOTIVATION

Prediction is very important in business. It helps us to design our strategies for optimizing our business. Here we can see that this is insanely fastest growing field. There will be several other competitors. So if we have already have future prediction based on current data, we can optimize our business in very efficient way. So strategic planning with the help of prediction was the real motivation behind this project.

PROBLEM STATEMENT

The goal of the company Seoul Bike is providing the city with a stable supply of rental bikes. It becomes a major concern to keep user satisfied. The crucial part is the prediction of bike count rents at each hour for a stable supply of rental bikes. We can suppose that this study could be reported to the company 'Seoul Bikes'. We think it could help them knowing if yes or not they have to supply bikes stations in the city, in order to keep a good satisfaction for the users and optimize their model. Rented Bike is a function of various parameters. So we are Rented Bike Count.

LITERATURE SURVEY

Bike sharing system is a fascinating topic to study. Numerous studies are carried out to know the capabilities and fascinating data about it. DeMaio 2009, describes bike sharing from the starting era of to the latest generation approaches. Fishman stated the research accomplished on all kinds of bicycle sharing system, the history of the documentation process, usage analysis, user relevance, mentioning the fact that some researchers used automated computerised devices for gathering data related to bicycle sharing system (Fishman, 2016). Researches additionally proposed an active public bike sharing problem Primarily Based on every day techniques including systems, while an modern approach for dealing with rental bicycle stations and variety of models to resolve the problem is developed by (Raviv & Kolka, 2013). Although several examinations have researched traffic stream and public rental bike demand prediction, just a few had been focussed about the moment-based demand in public bike sharing systems (Gao & Lee, 2019). We study the problem of predicting and analysing the general public rental bike demand in a bike sharing framework making use of data mining method

SOLUTION DESIGN

1.SOLUTION APPROCH

We have a regression problem because our target is the number of rented bikes per hour. So the goal of this part is to apply many algorithms in order to find the algorithm with the best indicator. The indicator we decided to choose is the Accuracy. This choice is because we wanted to be able to compare these algorithms between them and to choose which one is the most efficient. Let's apply regression techniques to our problem.

2.TECHNOLOGY STACK

1. Jupyter Notebook:

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

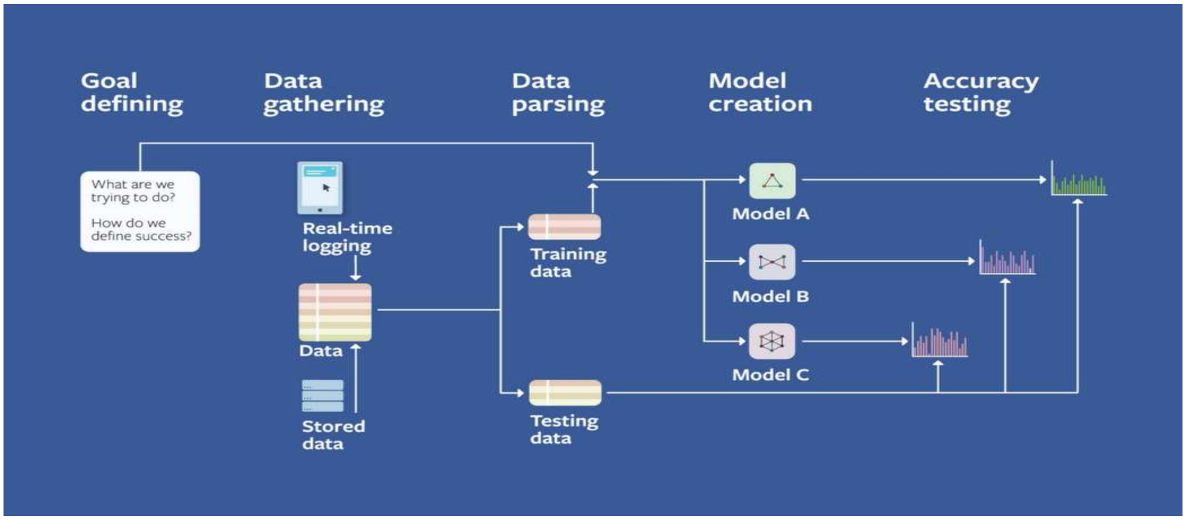
2.Various Libraries For Data Visualization, PreProcessing And Machine Learning Algorithms :-

A. Numpy:- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

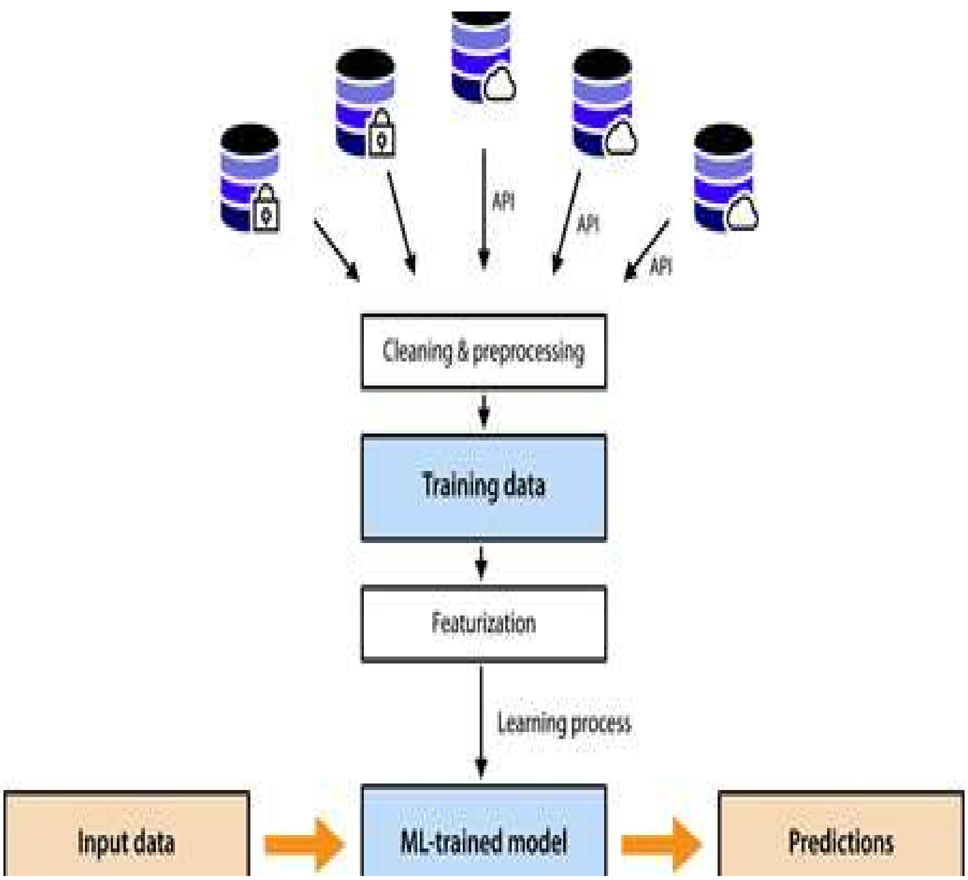
B. Pandas:- pandas. pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

C.Matplotlib & Seaborn:- This libraries are specifically used for Data Visualization.

D.SciKit -Learn:- Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.


Goal defining
Data gathering
Data parsing
Model creation
Accuracy testing
What are we trying to do?
How do we define success?
Real-time logging
Data
Stored data
Training data
Testing data
Model A
Model B
Model C


Preprocessing
Finds Out Relat ionship
Hypothesis Function
Input Data
PreProcessed and Scaled Data
Data Passed to Learning Algo.
Model Created


API
API
API
Cleaning & preprocessing
Training data
Featurization
Learning process
Input data
ML-trained model
Predictions

<u>SOLUTIONS IMPLEMENTATION AND RESULTS</u>

There are several Machine Learning Algorithms used for Prediction. We have used several algorithms. Before passing data to any prediction algorithm we have done stages like explained in above diagrams. The Stages are as follows:-

1. Reading the dataset with the help Pandas Library.
2. Then we performed EDA with help of Pandas Library. We understood and gain insights about data distribution.
3. Using Matplotlib we visualized the data.
4. We also perform PCA for better understanding of features.
5. After feature scaling, we passed data to various algorithms and calculated accuracies.
6. For Selecting the model we compared all accuracies and selected best fitting model.

ALGORITHMS USED

1. Multiple Linear Regression :- **Linear regression** is a **linear** approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).

A.Ridge:

B.Lasso:

2. KNN Regression:- KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood

A.Normal KNN:-

B.Weighted :-

3. SVR Regression:- Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences.

A.SVR(Linear)

4. Decision Tree Regressor:- Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Accuracies:-

5.Random Forest Regressor(Ensemble Learning):-

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

A random regression forest is an ensemble of randomized regression trees. Denote the predicted value at point by the -th tree, where are independent random variables, distributed as a generic random variable , independent of the sample .

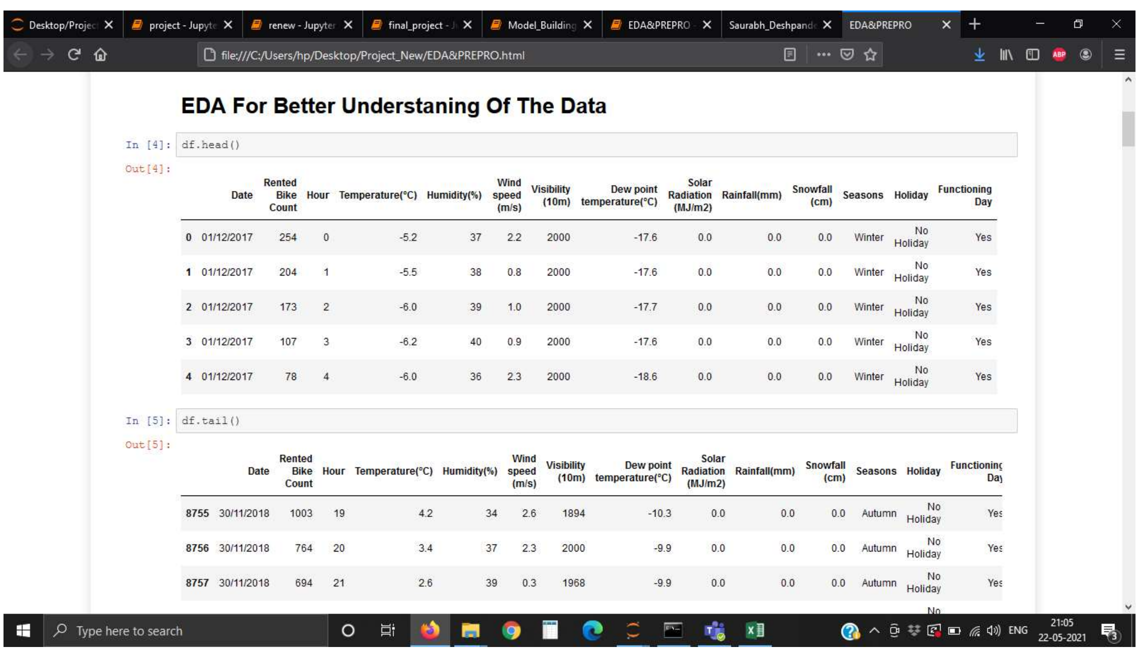
OBTAINING THE DATA

We obtained our dataset from Kaggle. **Kaggle** is the world's largest data science community with powerful tools and resources to help you achieve your data science goals. It has many public datasets available for learning purpose.

EDA & DATA VIZUALIZATION

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.



EDA For Better Understaning Of The Data

In [4]:
```
df.head()
```
Out[4]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	Yes
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	Winter	No Holiday	Yes

In [5]:
```
df.tail()
```
Out[5]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
8755	30/11/2018	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0	Autumn	No Holiday	Yes
8756	30/11/2018	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0	Autumn	No Holiday	Yes
8757	30/11/2018	694	21	2.6	39	0.3	1968	-9.9	0.0	0.0	0.0	Autumn	No Holiday	Yes

Description Of The Features Of Dataset

1.Date

2.Rented Bike Count : How many bikes were rented

3.Hour : The hour of the Day

4.Temperature(°C): Temprerature at Given Hour

5.Humidity(%) : Humidity in Environment

6.Wind speed (m/s) : Speed of the wind

7.Visibility : the distance one can see as determined by light and weather conditions.

8.Dew point temperature(°C) : the temperature to which the air would have to cool in order to reach saturation.

9.Solar Radiation (MJ/m2) : radiant energy emitted by the sun

10.Rainfall(mm) : How much raining that day

11.Snowfall (cm) : How much snowfall was occuring that day

12.Seasons : Season of the year

13.Holiday : Were the day was Holiday?

14.FunctioningDay : Were the day was FunctionDay

In [6]:
```
print('Datatypes of the Features')
df.info()
```

```
Datatypes of the Features
<class 'pandas.core.frame.DataFrame'>
```


14.FunctioningDay : Were the day was FunctionDay

In [6]:
```
print('Datatypes of the Features')
df.info()
```


```
Datatypes of the Features
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Date                    8760 non-null   object
 1   Rented Bike Count       8760 non-null   int64
 2   Hour                    8760 non-null   int64
 3   Temperature(°C)         8760 non-null   float64
 4   Humidity(%)             8760 non-null   int64
 5   Wind speed (m/s)        8760 non-null   float64
 6   Visibility (10m)        8760 non-null   int64
 7   Dew point temperature(°C)  8760 non-null   float64
 8   Solar Radiation (MJ/m2)    8760 non-null   float64
 9   Rainfall(mm)            8760 non-null   float64
 10  Snowfall (cm)           8760 non-null   float64
 11  Seasons                 8760 non-null   object
 12  Holiday                 8760 non-null   object
 13  Functioning Day         8760 non-null   object
dtypes: float64(6), int64(4), object(4)
memory usage: 958.2+ KB
```

In [7]:
```
df.isnull()
```

Out[7]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False

In [8]:
```
df.describe()
```
Out[8]:

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000
mean	704.602055	11.500000	12.882922	58.226256	1.724909	1436.825799	4.073813	0.569111	0.148687	0.075068
std	644.997468	6.922582	11.944825	20.362413	1.036300	608.298712	13.060369	0.868746	1.128193	0.436746
min	0.000000	0.000000	-17.800000	0.000000	0.000000	27.000000	-30.600000	0.000000	0.000000	0.000000
25%	191.000000	5.750000	3.500000	42.000000	0.900000	940.000000	-4.700000	0.000000	0.000000	0.000000
50%	504.500000	11.500000	13.700000	57.000000	1.500000	1698.000000	5.100000	0.010000	0.000000	0.000000
75%	1065.250000	17.250000	22.500000	74.000000	2.300000	2000.000000	14.800000	0.930000	0.000000	0.000000
max	3556.000000	23.000000	39.400000	98.000000	7.400000	2000.000000	27.200000	3.520000	35.000000	8.800000

In [9]:
```
df.isna().sum()
```
Out[9]:
```
Date                         0
Rented Bike Count            0
Hour                         0
Temperature(°C)              0
Humidity(%)                  0
Wind speed (m/s)             0
Visibility (10m)             0
Dew point temperature(°C)    0
Solar Radiation (MJ/m2)      0
Rainfall(mm)                 0
Snowfall (cm)                0
Seasons                      0
Holiday                      0
Functioning Day              0
dtype: int64
```

In [7]:
```
df.isnull()
```
Out[7]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	False	False	False	False	False	False	False	False	False	False	False	False	False	False
8756	False	False	False	False	False	False	False	False	False	False	False	False	False	False
8757	False	False	False	False	False	False	False	False	False	False	False	False	False	False
8758	False	False	False	False	False	False	False	False	False	False	False	False	False	False
8759	False	False	False	False	False	False	False	False	False	False	False	False	False	False

8760 rows × 14 columns

As we can see that there are some null values present in the dataset. So we will use Mean Imputer Strategy at Data Preprocessing stage. By doing this our model will give better Accuracy.

In [8]:
```
df.describe()
```
Out[8]:

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000

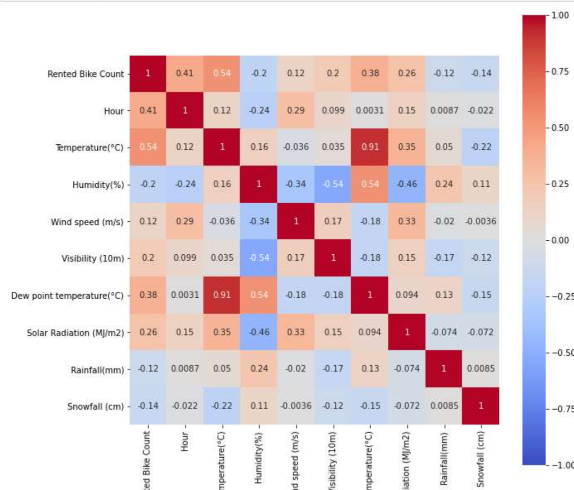
```
Seasons                    0
Holiday                    0
Functioning Day            0
dtype: int64
```

In [10]:
```python
df.corr()
```

Out[10]:

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)
Rented Bike Count	1.000000	0.410257	0.538558	-0.199780	0.121108	0.199280	0.379788	0.261837	-0.123074	-0.141804
Hour	0.410257	1.000000	0.124114	-0.241644	0.285197	0.098753	0.003054	0.145131	0.008715	-0.021516
Temperature(°C)	0.538558	0.124114	1.000000	0.159371	-0.036252	0.034794	0.912798	0.353505	0.050282	-0.218405
Humidity(%)	-0.199780	-0.241644	0.159371	1.000000	-0.336683	-0.543090	0.536894	-0.461919	0.236397	0.108183
Wind speed (m/s)	0.121108	0.285197	-0.036252	-0.336683	1.000000	0.171507	-0.176486	0.332274	-0.019674	-0.003554
Visibility (10m)	0.199280	0.098753	0.034794	-0.543090	0.171507	1.000000	-0.176630	0.149738	-0.167629	-0.121695
Dew point temperature(°C)	0.379788	0.003054	0.912798	0.536894	-0.176486	-0.176630	1.000000	0.094381	0.125597	-0.150887
Solar Radiation (MJ/m2)	0.261837	0.145131	0.353505	-0.461919	0.332274	0.149738	0.094381	1.000000	-0.074290	-0.072301
Rainfall(mm)	-0.123074	0.008715	0.050282	0.236397	-0.019674	-0.167629	0.125597	-0.074290	1.000000	0.008500
Snowfall (cm)	-0.141804	-0.021516	-0.218405	0.108183	-0.003554	-0.121695	-0.150887	-0.072301	0.008500	1.000000

In [11]:
```python
plt.figure(figsize=(10,10))
sns.heatmap(df.corr("pearson"),
            vmin=-1, vmax=1,
            cmap='coolwarm',
            annot=True,
            square=True);
```



In [14]:
```
df1Corr=pd.DataFrame(df.corr().unstack().sort_values(ascending=False)['Rented Bike Count'],columns=['Correlation to the targe
t'])
df1Corr.style.background_gradient(cmap=sns.light_palette("red", as_cmap=True))
```

Out[14]:

	Correlation to the target
Rented Bike Count	1.000000
Temperature(°C)	0.538558
Hour	0.410257
Dew point temperature(°C)	0.379788
Solar Radiation (MJ/m2)	0.261837
Visibility (10m)	0.199280
Wind speed (m/s)	0.121108
Rainfall(mm)	-0.123074
Snowfall (cm)	-0.141804
Humidity(%)	-0.199780

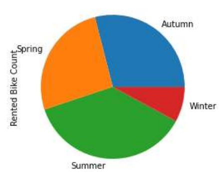
In [15]:
```
df1=pd.DataFrame(df.groupby('Seasons').sum()['Rented Bike Count'].sort_values(ascending=False))
df1.style.background_gradient(cmap=sns.light_palette("blue", as_cmap=True))
```

Out[15]:

	Rented Bike Count
Winter	644.070000

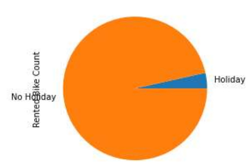
In [16]:
```
df.groupby('Seasons').sum()['Rented Bike Count'].plot.pie()
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x20933a6b430>



In [17]:
```
df.groupby('Holiday').sum()['Rented Bike Count'].plot.pie(radius=1)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x20933aa1b50>



Snowfall (cm)	-0.141804
Humidity(%)	-0.199780

In [15]:
```
df1=pd.DataFrame(df.groupby('Seasons').sum()['Rented Bike Count'].sort_values(ascending=False))
df1.style.background_gradient(cmap=sns.light_palette("blue", as_cmap=True))
```

Out[15]:

Seasons	Rented Bike Count
Summer	2283234
Autumn	1790002
Spring	1611909
Winter	487169

In [42]:
```
df1=pd.DataFrame(df.groupby('Seasons').sum()['Solar Radiation (MJ/m2)'].sort_values(ascending=False))
df1.style.background_gradient(cmap=sns.light_palette("Orange", as_cmap=True))
```

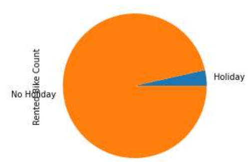
Out[42]:

Seasons	Solar Radiation (MJ/m2)
Summer	1680.850000
Spring	1520.840000
Autumn	1139.650000
Winter	644.070000

In [16]:
```
df.groupby('Seasons').sum()['Rented Bike Count'].plot.pie()
```

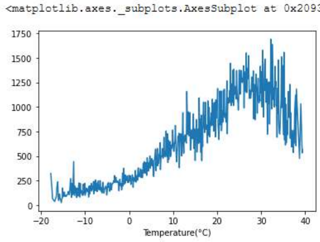
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x20933a6b430>

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x20933aa1b50>



In [18]:
```
df.groupby('Temperature(°C)').mean()['Rented Bike Count'].plot()
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x20933a9f640>



In [20]:
```
df.groupby('Snowfall (cm)').mean()['Rented Bike Count'].plot()
```

PREPROCESSING OF THE DATA

PreProcessing Of The Data

In [35]:

In [38]:
```
df=pd.read_csv("SeoulBikeData.csv",encoding="unicode escape")
df
```

Out[38]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functionin
0	01-12-2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	
1	01-12-2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	
2	01-12-2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	
3	01-12-2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	
4	01-12-2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	Winter	No Holiday	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8755	30-11-2018	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0	Autumn	No Holiday	
8756	30-11-2018	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0	Autumn	No Holiday	
8757	30-11-2018	694	21	2.6	39	0.3	1968	-9.9	0.0	0.0	0.0	Autumn	No Holiday	
8758	30-11-2018	712	22	2.1	41	1.0	1859	-9.8	0.0	0.0	0.0	Autumn	No Holiday	
8759	30-11-2018	584	23	1.9	43	1.3	1909	-9.3	0.0	0.0	0.0	Autumn	No Holiday	

In [39]:
```
Seasons={'Winter':1,'Spring':2,'Summer':3,'Autumn':4}
df['Seasons']=df.Seasons.map(Seasons)
Holiday={'No Holiday':1,'Holiday':0}
df['Holiday']=df.Holiday.map(Holiday)
Functioning_Day={'Yes':1,'No':0}
df['Functiong_Day']=df.Functioning_Day.map(Functioning_Day)
```

In [40]:
```
df
```

Out[40]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functionin
0	01-12-2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	1	1	
1	01-12-2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	1	1	
2	01-12-2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	1	1	
3	01-12-2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	1	1	
4	01-12-2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8755	30-11-2018	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0	4	1	
8756	30-11-2018	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0	4	1	
8757	30-11-2018	694	21	2.6	39	0.3	1968	-9.9	0.0	0.0	0.0	4	1	
8758	30-11-2018	712	22	2.1	41	1.0	1859	-9.8	0.0	0.0	0.0	4	1	
8759	30-11-2018	584	23	1.9	43	1.3	1909	-9.3	0.0	0.0	0.0	4	1	

8760 rows × 15 columns

In [41]:
```
del df['Functioning_Day']
```

8760 rows × 15 columns

In [41]:
```
del df['Functioning_Day']
del df['Date']
```

In [42]:
```
df
```

Out[42]:

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functiong_Day
0	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	1	1	1
1	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	1	1	1
2	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	1	1	1
3	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	1	1	1
4	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0	4	1	1
8756	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0	4	1	1
8757	694	21	2.6	39	0.3	1968	-9.9	0.0	0.0	0.0	4	1	1
8758	712	22	2.1	41	1.0	1859	-9.8	0.0	0.0	0.0	4	1	1
8759	584	23	1.9	43	1.3	1909	-9.3	0.0	0.0	0.0	4	1	1

8760 rows × 13 columns

3	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	1	1	1
4	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0	4	1	1
8756	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0	4	1	1
8757	694	21	2.6	39	0.3	1968	-9.9	0.0	0.0	0.0	4	1	1
8758	712	22	2.1	41	1.0	1859	-9.8	0.0	0.0	0.0	4	1	1
8759	584	23	1.9	43	1.3	1909	-9.3	0.0	0.0	0.0	4	1	1

8760 rows × 13 columns

In [43]:
```
#pca
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled=scaler.fit_transform(df)
```

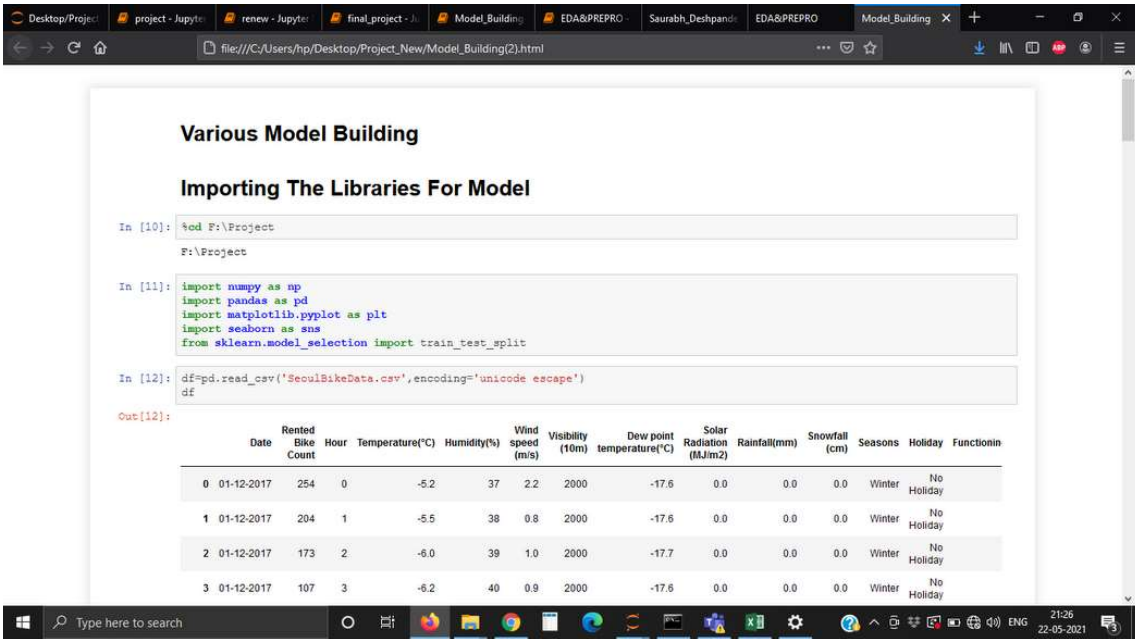
In [45]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=13)
pca.fit(df)
pc=pca.transform(df)
```

In [46]:
```
explained_variance=pca.explained_variance_ratio_
explained_variance
```

Out[46]:
```
array([6.03233245e-01, 3.96045030e-01, 5.07570367e-04, 1.60413449e-04,
       4.82528159e-05, 1.57559249e-06, 1.28914767e-06, 1.02861342e-06,
       8.47135580e-07, 4.35145089e-07, 2.19523432e-07, 5.89100974e-08,
       3.45017039e-08])
```

In [ ]:

MODEL BUILDING



Various Model Building

Importing The Libraries For Model

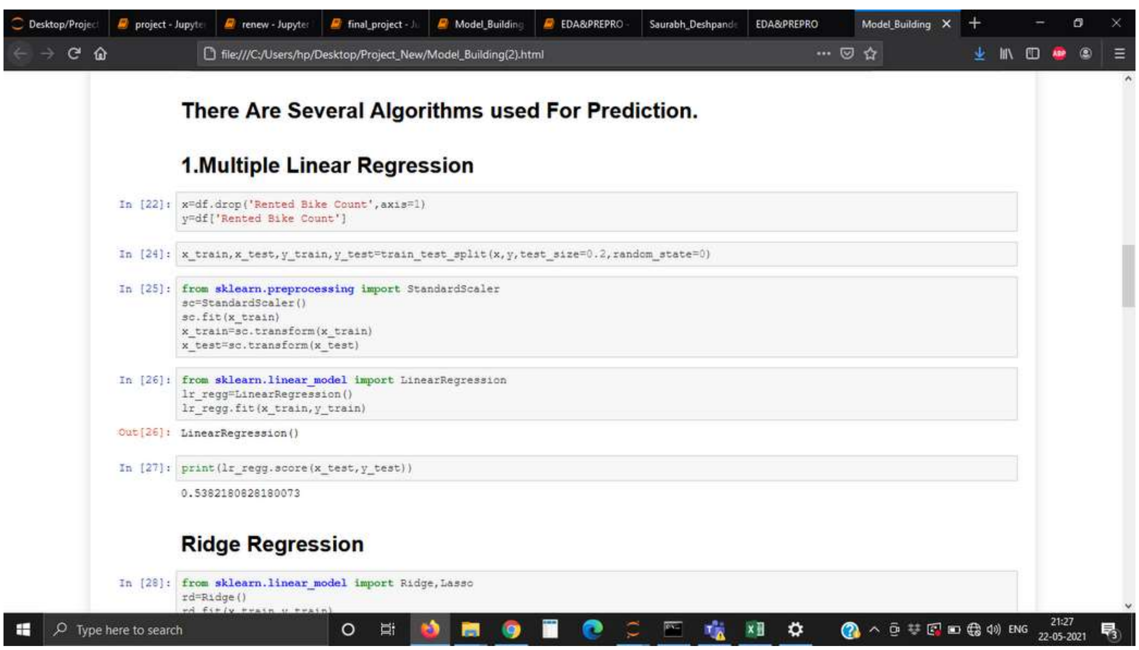
In [10]:
```
%cd F:\Project
```
```
F:\Project
```

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [12]:
```
df=pd.read_csv('SeoulBikeData.csv',encoding='unicode escape')
df
```

Out[12]:

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functionin
0	01-12-2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	
1	01-12-2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	
2	01-12-2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	
3	01-12-2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	



There Are Several Algorithms used For Prediction.

1.Multiple Linear Regression

In [22]:
```
x=df.drop('Rented Bike Count',axis=1)
y=df['Rented Bike Count']
```

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [25]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(x_train)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
```

In [26]:
```
from sklearn.linear_model import LinearRegression
lr_regg=LinearRegression()
lr_regg.fit(x_train,y_train)
```

Out[26]:
```
LinearRegression()
```

In [27]:
```
print(lr_regg.score(x_test,y_test))
```
```
0.5382180828180073
```

Ridge Regression

In [28]:
```
from sklearn.linear_model import Ridge,Lasso
rd=Ridge()
rd.fit(x_train,y_train)
```

Ridge Regression

In [28]:
```
from sklearn.linear_model import Ridge,Lasso
rd=Ridge()
rd.fit(x_train,y_train)
```
Out[28]: Ridge()

In [29]:
```
rd.score(x_test,y_test)
```
Out[29]: 0.5382139213733286

Lasso Regression

In [30]:
```
ls=Lasso()
ls.fit(x_train,y_train)
```
Out[30]: Lasso()

In [31]:
```
ls.score(x_test,y_test)
```
Out[31]: 0.5381096893206536

KNearestNeighbor Regression

Normal KNN Approch

In [33]:
```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train,y_train)
```


Desktop/Project | project - Jupyter | renew - Jupyter | final_project - Ju | Model_Building | EDA&PREPRO - | Saurabh_Deshpande | EDA&PREPRO | Model_Building
file:///C:/Users/hp/Desktop/Project_New/Model_Building(2).html

Out[31]: 0.5381096893206536

KNearestNeighbor Regression

Normal KNN Approch

In [33]:
```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train,y_train)
```
Out[33]: KNeighborsRegressor()

In [41]:
```
knn.score(x_test,y_test)
```
Out[41]: 0.7853490001354537

Weighted KNN Approch

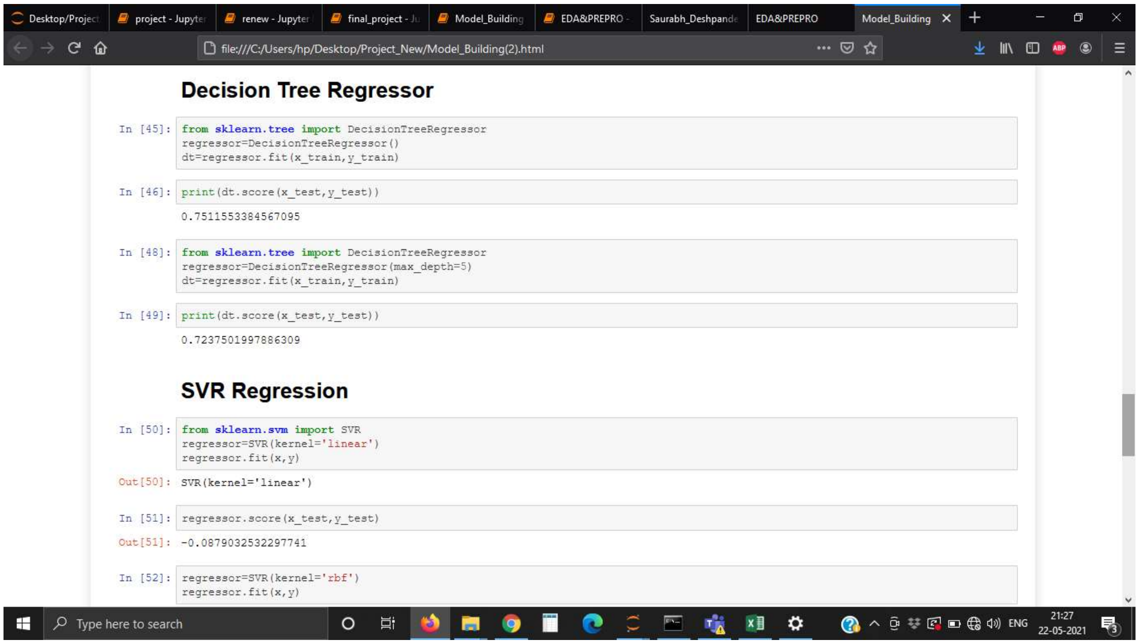
In [43]:
```
knn=KNeighborsRegressor(n_neighbors=5,weights='distance')
knn.fit(x_train,y_train)
```
Out[43]: KNeighborsRegressor(weights='distance')

In [44]:
```
knn.score(x_test,y_test)
```
Out[44]: 0.792630001038249

Decision Tree Regressor

In [45]:
```
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor()
dt=regressor.fit(x_train,y_train)
```

Type here to search
ENG 21:27 22-05-2021



Decision Tree Regressor

In [45]:
```
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor()
dt=regressor.fit(x_train,y_train)
```

In [46]:
```
print(dt.score(x_test,y_test))
```

0.7511553384567095

In [48]:
```
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor(max_depth=5)
dt=regressor.fit(x_train,y_train)
```

In [49]:
```
print(dt.score(x_test,y_test))
```

0.7237501997886309

SVR Regression

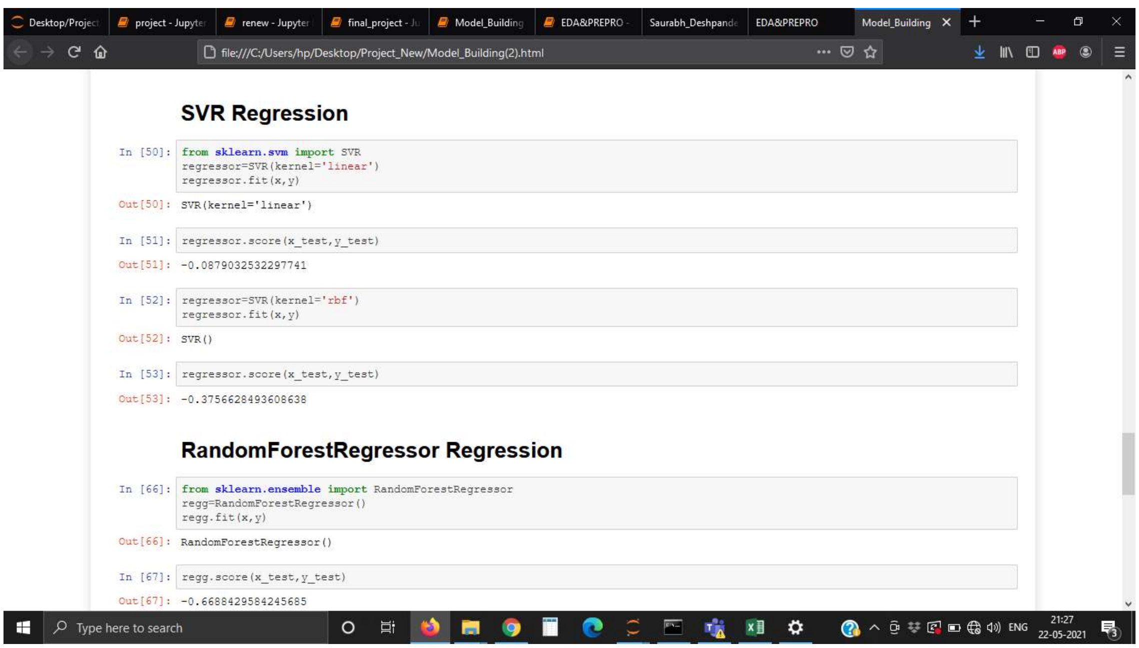
In [50]:
```
from sklearn.svm import SVR
regressor=SVR(kernel='linear')
regressor.fit(x,y)
```

Out[50]: SVR(kernel='linear')

In [51]:
```
regressor.score(x_test,y_test)
```

Out[51]: -0.0879032532297741

In [52]:
```
regressor=SVR(kernel='rbf')
regressor.fit(x,y)
```



SVR Regression

In [50]:
```
from sklearn.svm import SVR
regressor=SVR(kernel='linear')
regressor.fit(x,y)
```

Out[50]: SVR(kernel='linear')

In [51]:
```
regressor.score(x_test,y_test)
```

Out[51]: -0.0879032532297741

In [52]:
```
regressor=SVR(kernel='rbf')
regressor.fit(x,y)
```


Out[52]: SVR()

In [53]:
```
regressor.score(x_test,y_test)
```

Out[53]: -0.3756628493608638

RandomForestRegressor Regression

In [66]:
```
from sklearn.ensemble import RandomForestRegressor
regg=RandomForestRegressor()
regg.fit(x,y)
```

Out[66]: RandomForestRegressor()

In [67]:
```
regg.score(x_test,y_test)
```

Out[67]: -0.6688429584245685

RandomForestRegressor Regression

In [66]:
```
from sklearn.ensemble import RandomForestRegressor
regg=RandomForestRegressor()
regg.fit(x,y)
```
Out[66]: `RandomForestRegressor()`

In [67]:
```
regg.score(x_test,y_test)
```
Out[67]: `-0.6688429584245685`

In [68]:
```
from sklearn.ensemble import RandomForestRegressor
regre=RandomForestRegressor(n_estimators=500,criterion='mse')
regre.fit(x_train,y_train)
```
Out[68]: `RandomForestRegressor(n_estimators=500)`

In [69]:
```
regre.score(x_test,y_test)
```
Out[69]: `0.8717076032381055`

In [70]:
```
y_pred=regressor.predict(x_test)
```

In [71]:
```
df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred,'Total_Cost':y_pred*750,'Profit':((y_pred*750)*15/100)})
df
```
Out[71]:

	Actual	Predicted	Total_Cost	Profit
4059	320	231.082	173311.5	25996.725
6914	580	477.134	357850.5	53677.575
233	226	387.008	290256.0	43538.400

In [68]:
```
from sklearn.ensemble import RandomForestRegressor
regre=RandomForestRegressor(n_estimators=500,criterion='mse')
regre.fit(x_train,y_train)
```
Out[68]: `RandomForestRegressor(n_estimators=500)`

In [69]:
```
regre.score(x_test,y_test)
```
Out[69]: `0.8717076032381055`

In [70]:
```
y_pred=regressor.predict(x_test)
```

In [71]:
```
df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred,'Total_Cost':y_pred*750,'Profit':((y_pred*750)*15/100)})
df
```

Out[71]:

	Actual	Predicted	Total_Cost	Profit
4059	320	231.082	173311.5	25996.725
6914	580	477.134	357850.5	53677.575
233	226	387.008	290256.0	43538.400
982	165	206.526	154894.5	23234.175
5967	646	709.686	532264.5	79839.675
...	...	...	...	...
3152	0	298.200	223650.0	33547.500
5867	547	589.424	442068.0	66310.200
3456	213	409.282	306961.5	46044.225
7750	1331	1251.078	938308.5	140746.275
7728	783	713.660	535245.0	80286.750

1752 rows × 4 columns

References:

Elliot Fishman (2016) E-bikes in the Mainstream: Reviewing a Decade of Research

Tel Raviv Optimal inventory management of a bike-sharing station