CHAPTER-1

INTRODUCTION

These days commuting has become essential for all the people in cities to reach their destinations from present location. Taxi is one of the important modes of transport. So it has become a large scale business for many internet based companies like Uber, Ola But these companies and cab drivers are facing major problems. Searching for a passenger is one of the most important challenges for all cab drivers.If taxi driver spends more time in searching a new passenger, then fuel consumption will be high and the lesser number of passengers will be transported. As an inexperienced cab driver, we generally don't know where to pick-up a new customer as we don't have proper information about the demand of taxi over time and location.

This information regarding taxi demand in the future can be used to navigate both inexperienced and experienced taxi drivers to the areas where there is high demand in the city faster. So it helps to meet the demand with the supply for taxi services in the urban areas.This prediction of demand is challenging because it depends on many parameters. Suddenly there would be hike in demand due to rain in that particular location. Generally we rely on manual work but it is not sufficient. So we want better regression based machine learning and deep learning algorithms.

1.1 Objective of the Project

To prepare a predictive regression models by training it with the collected data set.By comparing the different models we will select the best model to predict the taxi demand which helps the taxi drivers.

1.2 Problem Statement

This project aims to predict the estimated price of a ride and demand of a uber cab given the factors like distance of the ride, surge multipliers, pick-up and drop location, weather and wind conditions and time of the commute.

CHAPTER 2

DATA SCIENCE PROCESS

Data Science Process is an agile, iterative data science methodology to deliver predictive analytics solutions and intelligent applications efficiently. DSP helps improve team collaboration and learning. It contains a distillation of the best practices and structures from Microsoft and others in the industry that facilitate the successful implementation of data science initiatives. The goal is to help companies fully realize the benefits of their analytics program. We provide a generic description of the process here that can be implemented with a variety of tools. A more detailed description of the project tasks and roles involved in the lifecycle of the process is provided in additional linked topics. The process may involve 7 clear cut steps for data analysis as shown in Figure

2.1 Process Model:

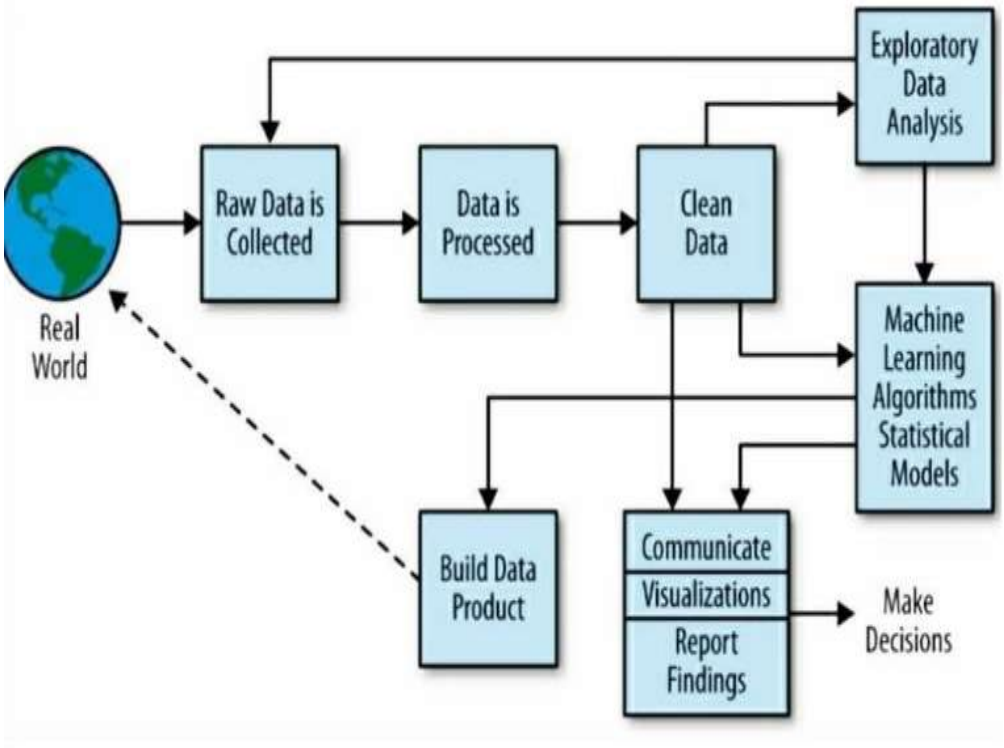


Fig:2.1 Process Model

Step 1: Frame or define the (business) problem.

Step 2: Collect the raw data needed for your problem (and map it to machine learning in case of bigdata.

Step 3: Data Preparation for process the data for analysis.

Step 4: Explore the data (Exploratory Data Analysis (EDA).

Step 5: Perform in-depth analysis (Modelling) and producing prescriptive Business Insights.

Step 6: Evaluation.

Step 7: Visualisation and Communication of Results of the Analyse

2.1.1 PREPROCESSING TECHNIQUES

Data modelling in software engineering is the process of simplifying the diagram or data model of a software System by applying certain formal techniques.It involves expressing data and information through text and symbols.

Data collection:

Relevant data is gathered from operational systems, data warehouses and other data sources. During this step, members of the BI team, other data professionals and end users gathering data themselvesshould confirm that the data is a good fit for the objectives of the planned applications.

Data cleaning:

In this step, the identified data errors are corrected to create complete and accurate data sets that are ready to be processed and analyze To handle irrelevant or missing data. Data is cleaned by filling in the missing values, smoothing noisy data, identifying and removing outliers and resolving any inconsistencies. For example, faulty data is removed or fixed, missing values are filled in and inconsistent entries are harmonized.

Filling missing values:

Method 1: Replacing missing values with zeros.

Method 2: Dropping rows with missing values.

Method 3: Replace Missing values with Mean/Mode/Median.

Data integration:

Redundant data can be detected using the concept of correlation analysis.
There are several methods used in correlation analysis to find the correlation coefficient (a value between 0 and 1), which measures the strength and the direction of a linear relationship between two variables.

One of the standard methods of finding the correlation coefficient is called:

Karl Pearson's Coefficient of Correlation (denoted by r)

N represents the number of pairs of observations and σ x, σ y represents the standard deviation of series x and y respectively.

r = +1 indicates a perfect positive relationship between two variables;

r = -1 indicates a perfect negative relationship between two variables, and r=0 indicates that there is no relationship between the two variables

Physical data integration:

Also called Common Data Storage, maintains a copy of the data from the source to a new system. The collected integrated data from several sources is stored and managed by the new system instead of the source system.

Followed by data warehouse systems.

Data transformation:

Various data transformation techniques are used during data preprocessing.

The choice of data transformation technique depends on how the data will be later used for analysis.

1. Rescaling data:

When the data encompasses attributes with varying scales, many statistical or machine learning techniques prefer rescaling the attributes to fall within a given scale. • Rescaling of data allows scaling all data values to lie between a specified minimum and maximum value (between 0 and 1).

2. Normalizing data:

Normalizing rescales data in such a way that each row of observation equals to a length of 1 (called a unit norm in linear algebra).

Data normalization is done prior to data analysis in many cases such as for sparse data having lots of zeroes or for attributes having high varying ranges of data values.

3. Binarizing data:

Binarizing is the process of converting data to either 0 or 1 based on a threshold value.

All the data values above the threshold value are marked 1 whereas all the data values equal to or below the threshold value are marked as 0.

Data reduction:

To reduce the unimportant or unwanted features from a dataset.

Strategies for data reduction include:

Dimensionally reduction

Data cube aggregation

Numerosity reduction

Converting the data to smaller forms so as to reduce the volume of data.

Reduction may be either parametric or non parametric.

Parametric use a model to represent data in which parameters of the data are stored, rather than data itself.

Examples: regression and log-linear models.

Non-parametric methods are used for storing reduced representations of the data.

Examples: clustering, histograms, sampling, and data cube aggregation.

Data discretization:

The data discretization method is used to partition the range of continuous attributes into intervals several continuous attribute values are replaced by lesser interval labels

Data discretization can be of two types:-

Top-down discretization

Bottom-up discretization

Top-down discretization:

Splitting, first finding one or few split points to divide the entire attribute range. This process is repeated on the resulting intervals until the stopping condition is found to be true.

Bottom-up discretization:

Merging , first considering all the continuous values as potential split-points. The process then further removes some continuous values by merging neighborhood values to form interval.

Data Transformation:

Data transformation can be divided into the following steps, each applicable as needed based on the of complexity the transformation required.

Data discovery

Data mapping

Code generation

Code execution

Data review

These steps are often the focus of developers or technical data analysts who may use multiple specialized tools to perform their tasks.

Types of data transformation:

1.Batch Data Transformation.

2.Interactive Data Transformation.

2.1.2 EXPLORATORY DATA ANALYSIS

Discovered in the 1970s by American mathematician John Tukey, exploratory data analysis (EDA) is a method of analysing and investigating the data sets to summarise their main characteristics. Scientists often use data visualisation methods to discover patterns, spot anomalies, check assumptions or test a hypothesis through summary statistics and graphical representations. EDA goes beyond the formal modelling or hypothesis to give maximum insight into the data set and its structure, and in identifying influential variables. It can also help in selecting the most suitable data analysis technique for a given project. Specific knowledge, such as the creation of a ranked list of relevant factors to be used as guidelines, can also be obtained using EDA.

The objectives of EDA are to:

Suggest hypotheses about the causes of observed phenomena
Assess assumptions on which statistical inference will be based
Support the selection of appropriate statistical tools and techniques
Provide a basis for further data collection through surveys or experiments

Many EDA techniques have been adopted into data mining, as well as into big data analytics.They are also being taught to young students as a way to introduce them to statistical thinking. In recent days, exploratory data analysis is a must and has been included in the big data analytics life cycle. The ability to find insight and be able to communicate it effectively in an organization is fueled with strong EDA capabilities. Based on Tuckey‘s ideas, Bell Labs developed the S programming languagein order to provide an interactive interface for doing statistics. The idea of S was to provide extensive graphical capabilities with an easy-to-use language. In today‘s world, in the context of Big Data, R that is based on the Sprogramming language is the most popular software for analytics.

The following program demonstrates the use of exploratory data analysis. In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tuckey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA

Data visualization tools:

Jupyter notebook:

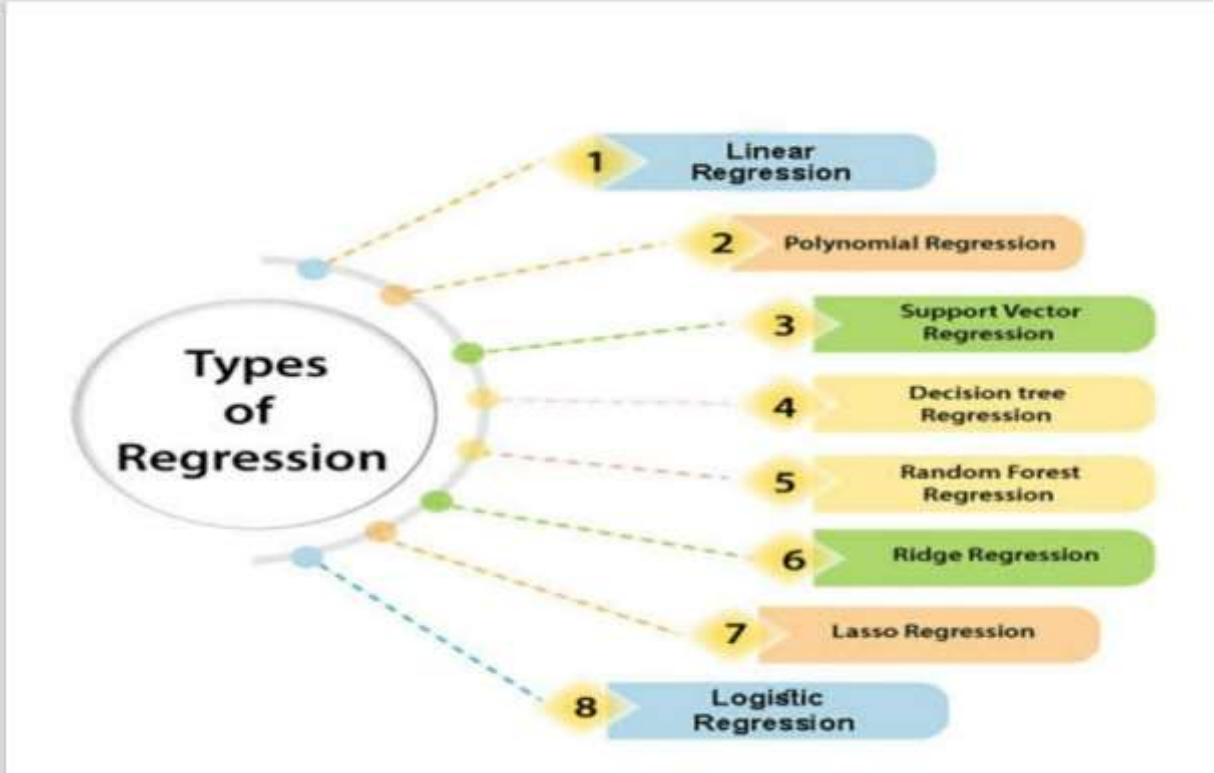
The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.Jupyter Notebooks are a spin-off project from the IPython project, which used 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.

2.1.3 Machine Learning Algorithms:

Regression:

The regression method of forecasting, as the name implies, is used for forecasting and for finding the casual relationship between variables. From a business point of view, the regression method of forecasting can be helpful for an individual working with data in the following ways:

Predicting sales in the near and long term.
Understanding demand and supply.
Understanding inventory levels.
Review and understand how variables impact all these factors



2.1.3 Types of Regressions

Linear Regression :

Linear regression is a statistical regression method which is used for predictive analysis.It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. It is used for solving the regression problem in machine learning. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression.

Logistic Regression :

Logistic regression is another supervised learning algorithm which is used to solve the classification problems.In classification problems, we have dependent variables in a binary or discrete format such as 0 or1.Logistic regression algorithm works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.

It is a predictive analysis algorithm which works on the concept of probability. Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used.Logistic regression uses sigmoid function or logistic function which is a complex cost function. This sigmoid 8 function is used to model the data in logistic regression. The function can be represented as:

Regression Analysis in Machine learning

f(x)= Output between the 0 and 1 value.

x= input to the function

e= base of natural logarithm.

Polynomial Regression:

Polynomial Regression is a type of regression which models the non-linear dataset using a linear model. Polynomial regression, the original features are transformed into polynomial features of given degree and then modeled using a linear model. Which means the datapoints are best fitted using a polynomial line.

Multi linear regression:

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

CHAPTER 3

DESIGN

3.1 Software requirments:

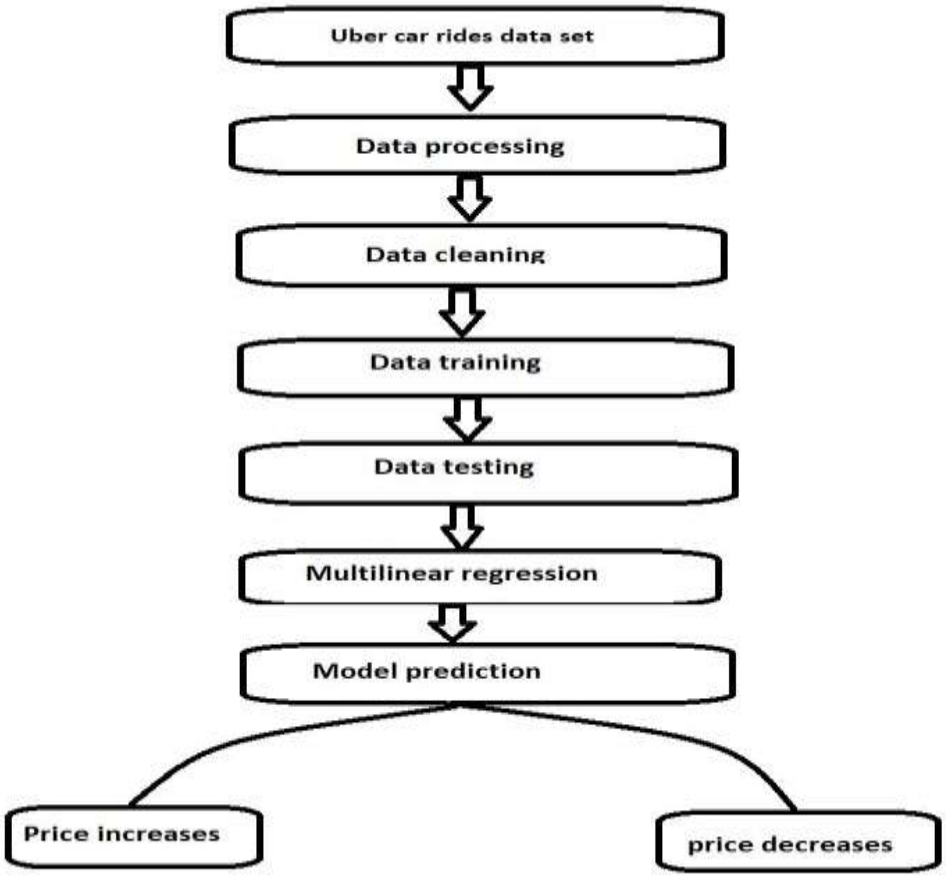
- ✓ Operating system: Windows
- ✓ Tools: Jupyter notebook
- ✓ Programming Language: Python

3.2 Hardware requirments:

- ✓ RAM: 4GB DDR-III
- ✓ HDD Size**:** 320GB ,7200 RPM
- ✓ Processor: Intel core I3 ,3rd generation

3.3 Data Flow Diagram/Flowchart:

Fig:3.3 Flow chart



CHAPTER 4

IMPLEMENTATION

4.1 Data preparation:

The project is about on world's largest taxi company Uber. The factors include distance, source, destination, as well as weather details.

	distance	cab_type	time_stamp	destination	source	price	surge_multiplier	id	product_id	name	temp	clouds	pressure	rain	humidity	wind
0	0.44	Lyft	1.54E+12	North Station	Haymarket Square	5	1	424553bb-7174-41ea-aab4-fe06d4f4b9d7	lyft_line	Shared	38.458571	0.66125	1010.13821	0.004846	0.749464	6.651786
1	0.44	Lyft	1.54E+12	North Station	Haymarket Square	11	1	4bd23055-6827-41c6-b23b-3c491f24e74d	lyft_premier	Lux	38.458571	0.66125	1010.13821	0.004846	0.749464	6.651786
2	0.44	Lyft	1.54E+12	North Station	Haymarket Square	7	1	981a3613-77af-4620-a42a-0c0866077d1e	lyft	Lyft	38.458571	0.66125	1010.13821	0.004846	0.749464	6.651786
3	0.44	Lyft	1.54E+12	North Station	Haymarket Square	26	1	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	lyft_luxsuv	Lux Black XL	38.458571	0.66125	1010.13821	0.004846	0.749464	6.651786
4	0.44	Lyft	1.54E+12	North Station	Haymarket Square	9	1	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	lyft_plus	Lyft XL	38.458571	0.66125	1010.13821	0.004846	0.749464	6.651786
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1176	1.26	Lyft	1.54E+12	Haymarket Square	Financial District	27.5	1	6481b16f-1ebe-4270-907c-d0cc295e98c2	lyft_luxsuv	Lux Black XL	37.946154	0.642404	1010.39048	0.004866	0.742981	6.458846
1177	1.26	Lyft	1.54E+12	Haymarket Square	Financial District	3	1	a54ea524-58ca-4084-bd9f-dab5222c6c01	lyft_line	Shared	37.946154	0.642404	1010.39048	0.004866	0.742981	6.458846
1178	1.26	Lyft	1.54E+12	Haymarket Square	Financial District	7	1	c0a06a71-596f-4e06-a4a7-a325bb2e5db3	lyft	Lyft	37.946154	0.642404	1010.39048	0.004866	0.742981	6.458846

Fig.4.1.1 Data set

Distance: Uber uses Google maps which in turn calculates accurate distance based on the GPS system. The GPS system helps in calculating distances between two places using coordinates.

Cab type: There are two types of cabs named Uber and Lyft

Time stamp: Wait Time fees allow you to be compensated by riders for the time spent waiting at a pickup location. Wait Time fees begin to accumulate after you arrive at your rider's pickup location, and are charged at a per minute rate.

Destination: The place to which someone or something is going or being sent.

Source: A place, person, or thing from which something originates or can be obtained.

Surge multiplier: Surge is a real-time opportunity to earn extra money for a delivery when it's busy. You can still earn quest and boost on top of money earned from a delivery with surge.

ID: All drivers and delivery people submit identity verification documents before using uber platform which are known as ID.

Product id: The Product ID is a permanent, unique identifier assigned to a data product by its producer.

Name: It indicates the type of uber.

Temperature: Degree of hotness or coldness measured on a definite scale either in source or in destination, , It is expressed in celisus.

Pressure: The air around you has weight, and it presses against everything it touches. That pressure is called atmospheric pressure, or air pressure, It is expressed in pascal.

Cloud: It indicates whether it rains or not at that particular place, , It is expressed in oktas.

Wind: It indicates the moment of air with high pressure or low pressure either at source or destination, , It is expressed in kilo meter per hour.

Humidity: Amount of water vapour in air, It is expressed in percentage.

Rain: It indicates the rain fall in mm.

Data integration:

We have collected the data sets of car rides and weather.

1.Rides

	distance	cab_type	time_stamp	destination	source	price	surge_multiplier	id	product_id	name
0	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	5.0	1.0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	lyft_line	Shared
1	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	11.0	1.0	4bd23055-6827-41c6-b23b-3c491f24e74d	lyft_premier	Lux
2	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	7.0	1.0	981a3613-77af-4620-a42a-0c0866077d1e	lyft	Lyft
3	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	26.0	1.0	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	lyft_luxsuv	Lux Black XL
4	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	9.0	1.0	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	lyft_plus	Lyft XL
...	...	...	...	...	...	...	...	...	...	...
1281	1.00	Uber	1.543710e+12	North End	West End	13.0	1.0	616d3611-1820-450a-9845-a9ff304a4842	6f72dfc5-27f1-42e8-84db-ccc7a75f6969	UberXL
1282	1.00	Uber	1.543710e+12	North End	West End	9.5	1.0	633a3fc3-1f86-4b9e-9d48-2b7132112341	55c66225-fbe7-4fd5-9072-eab1ece5e23e	UberX
1283	1.00	Uber	1.543710e+12	North End	West End	NaN	1.0	64d451d0-639f-47a4-9b7c-6fd92fbd264f	8cf7e821-f0d3-49c6-8eba-e679c0ebcf6a	Taxi
1284	1.00	Uber	1.543710e+12	North End	West End	27.0	1.0	727e5f07-a96b-4ad1-a2c7-9abc3ad55b4e	6d318bcc-22a3-4af6-bddd-b409bfce1546	Black SUV
1285	1.00	Uber	1.543710e+12	North End	West End	10.0	1.0	e7fdc087-fe86-40a5-a3c3-3b2a8badcbda	997acbb5-e102-41e1-b155-9df7de0a73f2	UberPool

1286 rows × 10 columns

Fig4.1.2rides data set

weather

	temp	location	clouds	pressure	rain	time_stamp	humidity	wind
0	42.42	Back Bay	1.00	1012.14	0.1228	1545003901	0.77	11.25
1	42.43	Beacon Hill	1.00	1012.15	0.1846	1545003901	0.76	11.32
2	42.50	Boston University	1.00	1012.15	0.1089	1545003901	0.76	11.07
3	42.11	Fenway	1.00	1012.13	0.0969	1545003901	0.77	11.09
4	43.13	Financial District	1.00	1012.14	0.1786	1545003901	0.75	11.49
...	...	...	...	...	...	...	...	...
1323	37.68	Theatre District	0.46	999.00	NaN	1543467459	0.70	9.73
1324	44.49	Fenway	0.97	1008.08	0.0330	1543271718	0.86	9.10
1325	45.48	South Station	0.97	1008.13	0.0664	1543271718	0.84	9.34
1326	45.21	Back Bay	0.88	1009.86	NaN	1543264509	0.83	9.35
1327	45.23	Beacon Hill	0.88	1009.87	NaN	1543264509	0.83	9.37

1328 rows × 8 columns

Fig:4.1.3 weather dataset

- We observed NULL values in both the data sets.
- Checking null values in rides dataset

Rides.isna( ).sum( )

```
distance             0
cab_type             0
time_stamp           0
destination          0
source               0
price              105
surge_multiplier     0
id                   0
product_id           0
name                 0
dtype: int64
```

Fig:4.1.4 rides null values

- Checking null values in weather dataset

weather.isna( ).sum( )

```
temp              0
location          0
clouds            0
pressure          0
rain           1216
time_stamp        0
humidity          0
wind              0
dtype: int64
```

Fig:4.1.5 weather null values

- We can drop NULL values in rides data set as it doesn't make any difference but, if we drop the NULL values in weather data set then it makes more difference
- So we drop the NULL values in rides data set and replace the NULL values with zero in weather data set.

Ridesdf=rides.dropna

Ridesdf

	distance	cab_type	time_stamp	destination	source	price	surge_multiplier	id	product_id	name
0	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	5.0	1.0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	lyft_line	Shared
1	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	11.0	1.0	4bd23055-6827-41c6-b23b-3c491f24e74d	lyft_premier	Lux
2	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	7.0	1.0	981a3613-77af-4620-a42a-0c0866077d1e	lyft	Lyft
3	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	26.0	1.0	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	lyft_luxsuv	Lux Black XL
4	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	9.0	1.0	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	lyft_plus	Lyft XL
...	...	...	...	...	...	...	...	...	...	...
1280	1.00	Uber	1.543710e+12	North End	West End	9.5	1.0	353e6566-b272-479e-a9c6-98bd6cb23f25	9a0e7b09-b92b-4c41-9779-2ad22b4d779d	WAV
1281	1.00	Uber	1.543710e+12	North End	West End	13.0	1.0	616d3611-1820-450a-9845-a9ff304a4842	6f72dfc5-27f1-42e8-84db-ccc7a75f8969	UberXL
1282	1.00	Uber	1.543710e+12	North End	West End	9.5	1.0	633a3fc3-1f86-4b9e-9d48-2b7132112341	55c66225-fbe7-4fd5-9072-eab1ece5e23e	UberX
1284	1.00	Uber	1.543710e+12	North End	West End	27.0	1.0	727e5f07-a96b-4ad1-a2c7-9abc3ad55b4e	6d318bcc-22a3-4af6-bddd-b409bfce1546	Black SUV
1285	1.00	Uber	1.543710e+12	North End	West End	10.0	1.0	e7fdc087-fe86-40a5-a3c3-3b2a8badcbda	997acbb5-e102-41e1-b155-9df7de0a73f2	UberPool

1181 rows × 10 columns

Fig:4.1.6 dropping null values

- Filling the values in data sets

Weatherdf=weather.fillna(0)

Weatherdf

	temp	location	clouds	pressure	rain	time_stamp	humidity	wind
0	42.42	Back Bay	1.00	1012.14	0.1228	1545003901	0.77	11.25
1	42.43	Beacon Hill	1.00	1012.15	0.1846	1545003901	0.76	11.32
2	42.50	Boston University	1.00	1012.15	0.1089	1545003901	0.76	11.07
3	42.11	Fenway	1.00	1012.13	0.0969	1545003901	0.77	11.09
4	43.13	Financial District	1.00	1012.14	0.1786	1545003901	0.75	11.49
...	...	...	...	...	...	...	...	...
1323	37.68	Theatre District	0.46	999.00	0.0000	1543467459	0.70	9.73
1324	44.49	Fenway	0.97	1008.08	0.0330	1543271718	0.88	9.10
1325	45.48	South Station	0.97	1008.13	0.0664	1543271718	0.84	9.34
1326	45.21	Back Bay	0.88	1009.86	0.0000	1543264509	0.83	9.35
1327	45.23	Beacon Hill	0.88	1009.87	0.0000	1543264509	0.83	9.37

1328 rows × 8 columns

Fig:4.1.7 filling null values by zero

- We merged the two datasets based on the time stamp present in both data sets

Data=risesdf

.merge(source_weatherdf,on= 'source')

.merge (destination_weatherdf,on= 'destination')

data

	distance	cab_type	time_stamp	destination	source	price	surge_multiplier	id	product_id	name	...	pressure_x	rain_x	humidity_
0	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	5.0	1.0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	lyft_line	Shared	...	1010.390481	0.004866	0.74298
1	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	11.0	1.0	4bd23055-6827-41c6-b23b-3c491f24e74d	lyft_premier	Lux	...	1010.390481	0.004866	0.74298
2	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	7.0	1.0	981a3613-77af-4620-a42a-0c0866077d1e	lyft	Lyft	...	1010.390481	0.004866	0.74298
3	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	26.0	1.0	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	lyft_luxsuv	Lux Black XL	...	1010.390481	0.004866	0.74298
4	0.44	Lyft	1.540000e+12	North Station	Haymarket Square	9.0	1.0	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	lyft_plus	Lyft XL	...	1010.390481	0.004866	0.74298
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1176	1.26	Lyft	1.540000e+12	Haymarket Square	Financial District	27.5	1.0	6481b16f-1ebe-4270-907c-d0cc295e98c2	lyft_luxsuv	Lux Black XL	...	1009.828909	0.004710	0.74027

fig:4.1.8 merging data

- We have removed [time_of_day', 'time_stamp', 'date', 'hour', 'time', 'weekday', 'dayofweek', 'name'] from our data set.

df=df.drop(["time_of_day","name","time_stamp","date","time","time_of_day","weekday","dayofweek","hour"],axis=1)df

	distance	cab_type	destination	source	price	surge_multiplier	temp	clouds	pressure	rain	humidity	wind
0	0.44	0	North Station	Haymarket Square	5.0	1.0	38.458571	0.661250	1010.138214	0.004846	0.749464	6.651786
1	0.44	0	North Station	Haymarket Square	11.0	1.0	38.458571	0.661250	1010.138214	0.004846	0.749464	6.651786
2	0.44	0	North Station	Haymarket Square	7.0	1.0	38.458571	0.661250	1010.138214	0.004846	0.749464	6.651786
3	0.44	0	North Station	Haymarket Square	26.0	1.0	38.458571	0.661250	1010.138214	0.004846	0.749464	6.651786
4	0.44	0	North Station	Haymarket Square	9.0	1.0	38.458571	0.661250	1010.138214	0.004846	0.749464	6.651786
...	...	...	...	...	...	...	...	...	...	...	...	...
1176	1.26	0	Haymarket Square	Financial District	27.5	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1177	1.26	0	Haymarket Square	Financial District	3.0	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1178	1.26	0	Haymarket Square	Financial District	7.0	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1179	1.26	0	Haymarket Square	Financial District	16.5	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1180	1.26	0	Haymarket Square	Financial District	10.5	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846

1181 rows × 12 columns

Fig:4.1.9 Dropping columns

4.2 Exploratory Data Analysis:

Exploratory data analysis (EDA) is an approach of analyzing data sets of summarize their main characteristics, often using statistical graphics and other data visualization methods.

Print("(row columns):"+str(data.shape))

df.columns

out:

```
(row,columns):(1181, 20)

Index(['distance', 'cab_type', 'destination', 'source', 'price',
       'surge_multiplier', 'temp', 'clouds', 'pressure', 'rain', 'humidity',
       'wind'],
      dtype='object')
```

fig:4.2.1 Columns

df.nunique( )

out:

```
distance            144
cab_type              2
destination          12
source               12
price                69
surge_multiplier      4
temp                 12
clouds               12
pressure             12
rain                 12
humidity             12
wind                 12
dtype: int64
```

Fig:4.2.2 unique

Df.head( )

	distance	cab_type	destination	source	price	surge_multiplier	temp	clouds	pressure	rain	humidity	wind
0	0.44	0	North Station	Haymarket Square	5.0	1.0	38.458571	0.66125	1010.138214	0.004846	0.749464	6.651786
1	0.44	0	North Station	Haymarket Square	11.0	1.0	38.458571	0.66125	1010.138214	0.004846	0.749464	6.651786
2	0.44	0	North Station	Haymarket Square	7.0	1.0	38.458571	0.66125	1010.138214	0.004846	0.749464	6.651786
3	0.44	0	North Station	Haymarket Square	26.0	1.0	38.458571	0.66125	1010.138214	0.004846	0.749464	6.651786
4	0.44	0	North Station	Haymarket Square	9.0	1.0	38.458571	0.66125	1010.138214	0.004846	0.749464	6.651786

Fig:4.2.3 Head

df.tail( )

	distance	cab_type	destination	source	price	surge_multiplier	temp	clouds	pressure	rain	humidity	wind
1176	1.26	0	Haymarket Square	Financial District	27.5	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1177	1.26	0	Haymarket Square	Financial District	3.0	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1178	1.26	0	Haymarket Square	Financial District	7.0	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1179	1.26	0	Haymarket Square	Financial District	16.5	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846
1180	1.26	0	Haymarket Square	Financial District	10.5	1.0	37.946154	0.642404	1010.390481	0.004866	0.742981	6.458846

Fig:4.2.4 Tail

df.describe( )

	distance	cab_type	price	surge_multiplier	temp	clouds	pressure	rain	humidity	wind
count	1181.000000	1181.000000	1181.000000	1181.000000	1181.000000	1181.000000	1181.000000	1181.000000	1181.000000	1181.000000
mean	2.160186	0.535986	16.349280	1.010373	38.354664	0.666234	1010.142963	0.004238	0.748450	6.596907
std	1.075548	0.498915	9.044624	0.087020	0.282738	0.014171	0.315940	0.000785	0.006960	0.172654
min	0.390000	0.000000	3.000000	1.000000	37.946154	0.642404	1009.409459	0.002974	0.737130	6.250455
25%	1.220000	0.000000	9.000000	1.000000	38.023818	0.653519	1009.927778	0.003324	0.744273	6.494298
50%	2.150000	1.000000	13.500000	1.000000	38.352348	0.669099	1010.138214	0.004710	0.746356	6.651786
75%	3.010000	1.000000	22.500000	1.000000	38.493596	0.675702	1010.390481	0.004846	0.754636	6.666909
max	7.460000	1.000000	67.500000	2.500000	38.916273	0.690182	1010.602273	0.005281	0.760877	6.966759

Fig:4.2.5 Describe

Df.isnull( )

	distance	cab_type	destination	source	price	surge_multiplier	temp	clouds	pressure	rain	humidity	wind
0	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
2	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
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
1176	False	False	False	False	False	False	False	False	False	False	False	False
1177	False	False	False	False	False	False	False	False	False	False	False	False
1178	False	False	False	False	False	False	False	False	False	False	False	False
1179	False	False	False	False	False	False	False	False	False	False	False	False
1180	False	False	False	False	False	False	False	False	False	False	False	False

1181 rows × 12 columns

fig:4.2.6 Is null

- As we can see that there are no null values or some blank values in our data set so there is
- no need of replacing or filling values with mean in our dataset.

Data Visualization

Data and information visualization is an interdisciplinary field that deals with the graphic representation of data and information.

Heat map:

A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colours.

data_corr = data[["distance", "price", "surge_multiplier"]]

sns.heatmap(data_corr.corr(),cmap='viridis')

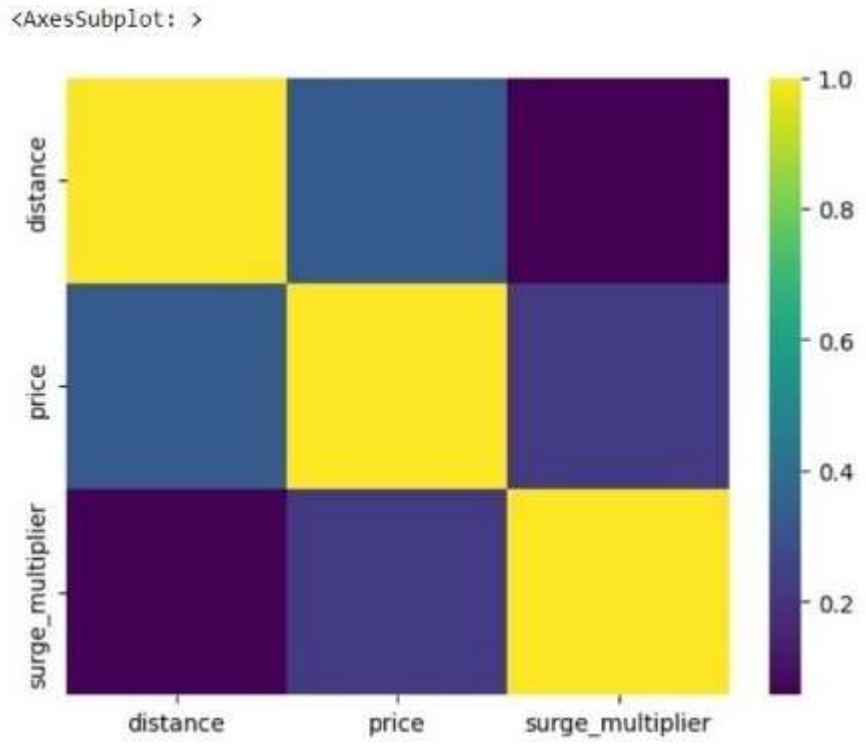


Fig:4.2.7 Heat map

Bar plot:

A barplot (or barchart) is one of the most common types of graphic. It shows the relationship between a numeric and a categoric variable.

data.head()

df=pd.DataFrame(data['cab_type'].value_counts())

df.reset_index()

df.plot(kind='bar')

plt.show()

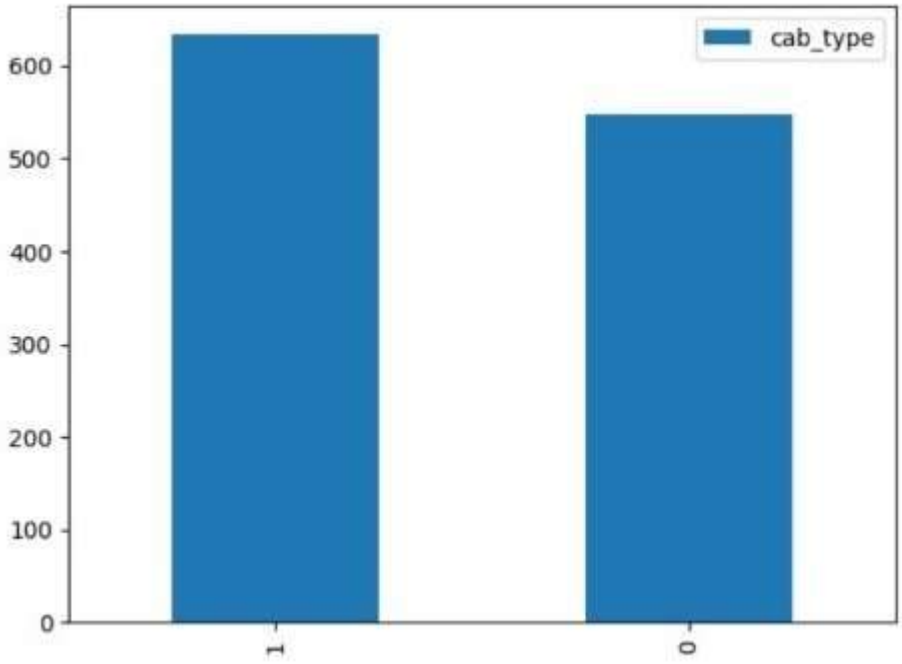


Fig:4.2.8 No. of rides

plt.title('price variation from source')

plt.xlabel('source')

plt.ylabel('price')

data.source.value_counts().plot(kind='bar',figsize=(20,5),color='blue')

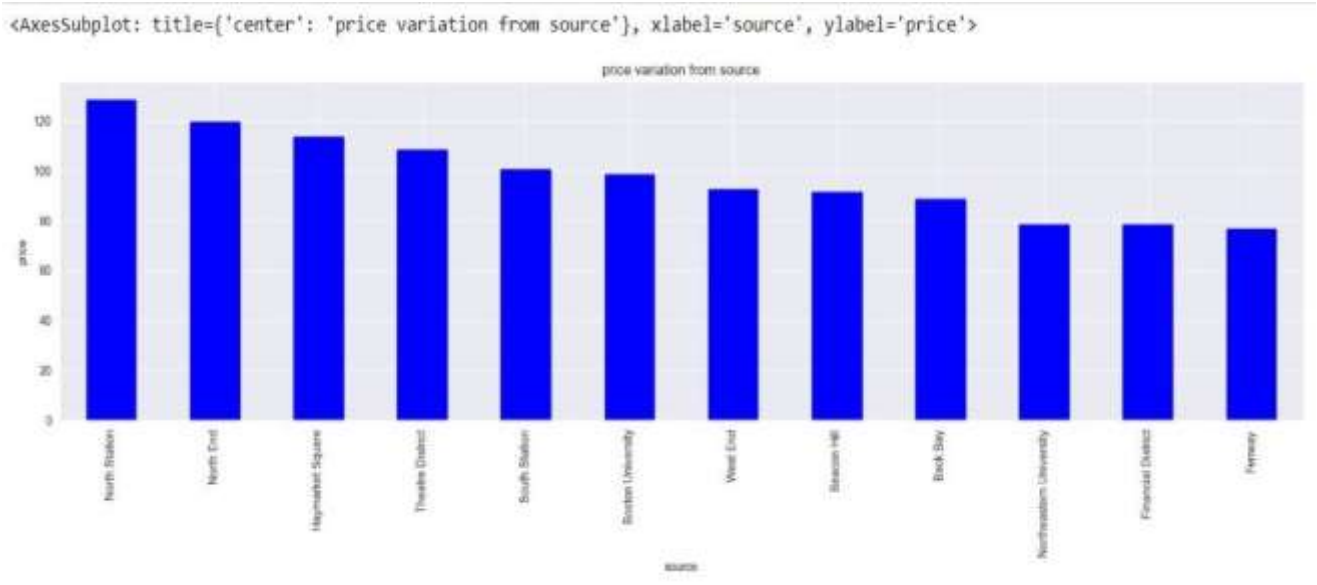


Fig:4.2.9Price variation from source

```
import matplotlib.pyplot as plt
df=pd.DataFrame(data['price'].groupby(data['rain']).sum())
df.plot(kind='bar')
plt.show()
```

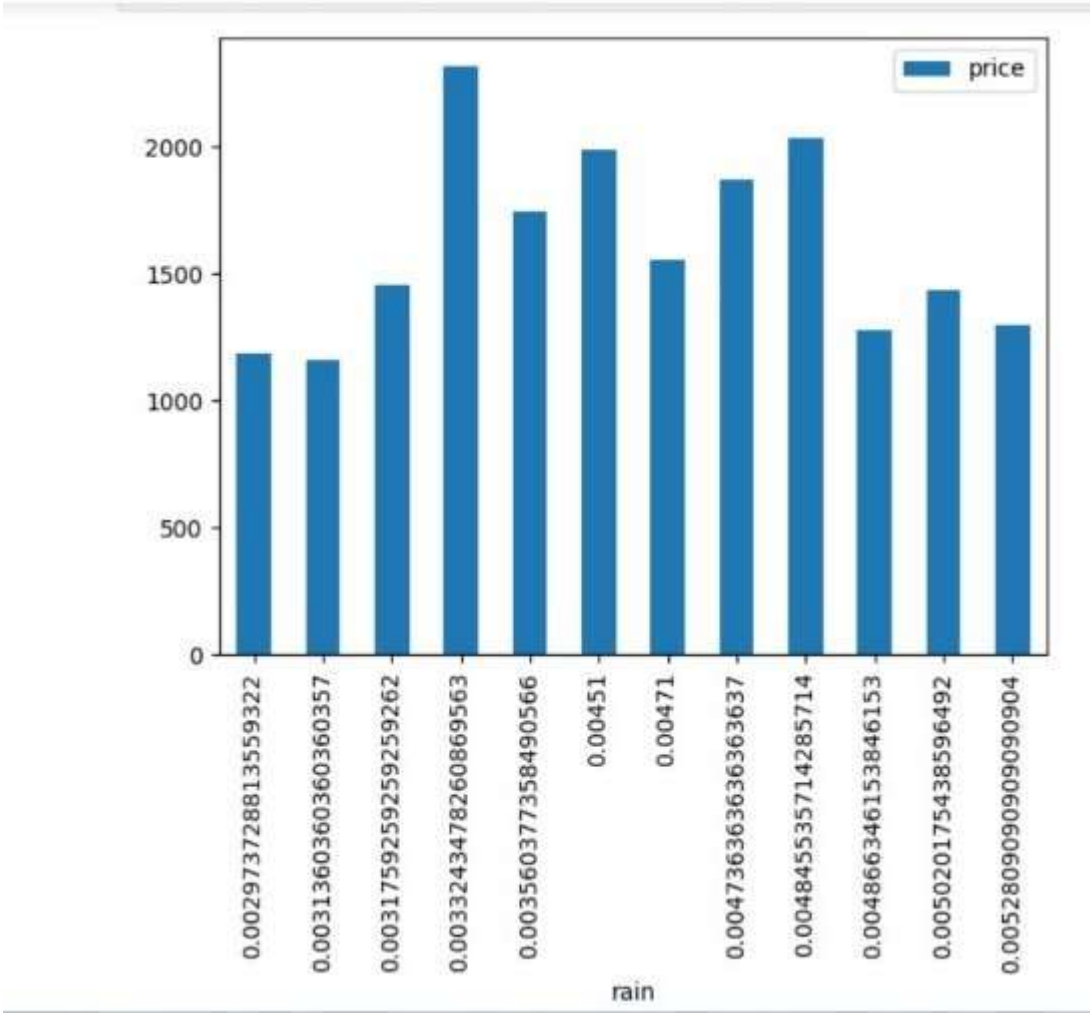


Fig:4.2.10 Variation of price w.r.t rain

```
df=data.reset_index()
sns.barplot(x=data['surge_multiplier'],y=data['price'])
```

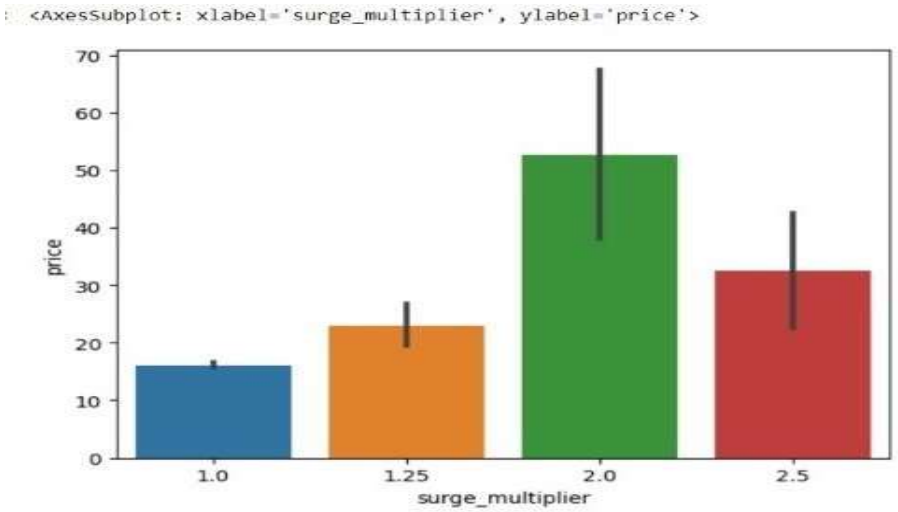


Fig 4.2.11 Variation of surge multiplier with price

high_surge_dataset = data[data["surge_multiplier"]> 1]

t_high_surge = pd.DataFrame(high_surge_dataset.groupby(["weekday", "surge_multiplier"]).size().reset_index())

t_high_surge.columns = ["Weekday", "Surge", "Count"]

plt.figure(figsize=(15, 5))

sns.barplot(x="Weekday", y="Count",hue="Surge", data=t_high_surge).set_title("Weekday wise Surge");

plt.legend(loc='upper right',frameon=False)

out:

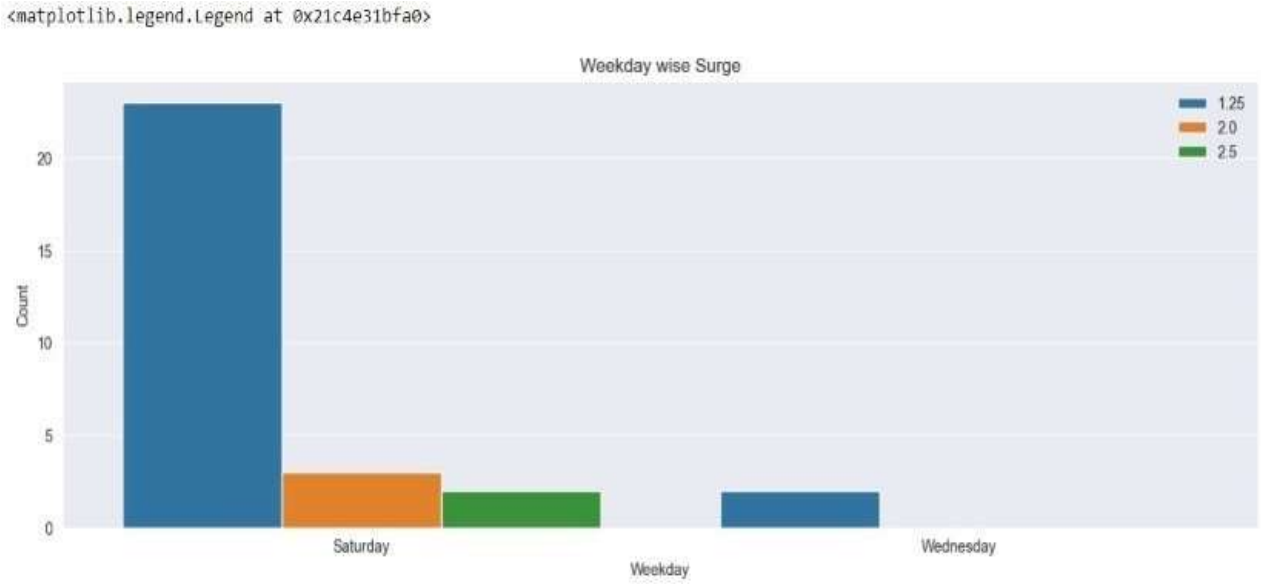


Fig:4.2.12 Weekday wise surge

df=data.reset_index()

sns.barplot(x=data['weekday'],y=data['price'])

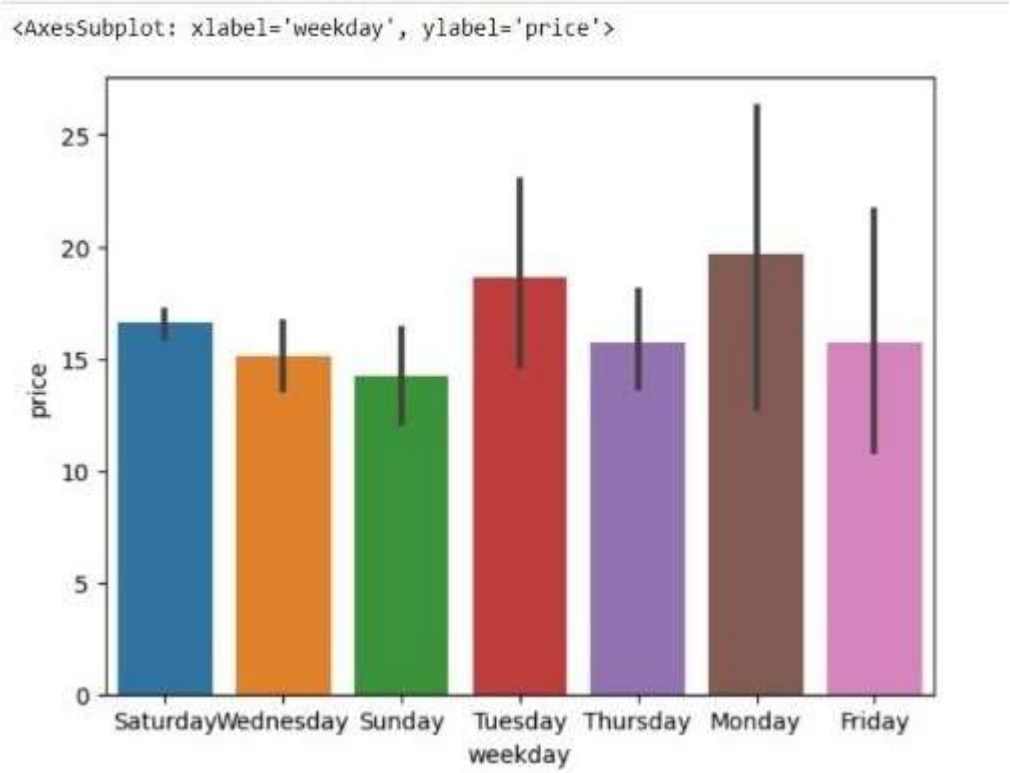


Fig:4.2.13 Price variation in weekdays

Correlation matrix:

A correlation matrix is a table containing correlation coefficients between variables. Each cell in the table represents the correlation between two variables. The value lies between 0$1

corr=data.corr()

plt.figure(figsize = (30,20))

mask = np.zeros_like(corr)

mask[np.tril_indices_from(mask, k = -1)] = True

sns.heatmap(corr, cmap = 'RdYlGn', vmax = 1.0, vmin = -1.0, annot = True,

annot_kws = {"size": 20}, mask = mask)

plt.xticks(fontsize = 12)

plt.yticks(fontsize = 15)

plt.show()

Fig:4.2.14 Correlation matrix

Feature selection:

Entropy:

Entropy is defined as the measure of randomness in the information being processed.

```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log
def ent(data,attribute):
    target_variables = data.price.unique()
    variables = data[attribute].unique()
    entropy_attribute = 0
    for variable in variables:
        entropy_each_feature = 0
        for target_variable in target_variables:
```

```
            num = len(data[attribute][data[attribute]==variable][data.price ==target_variable])
            den = len(data[attribute][data[attribute]==variable])
            fraction = num/(den+eps)  #pi
            entropy_each_feature += -fraction*log(fraction+eps)
        fraction2 = den/len(data)
        entropy_attribute += -fraction2*entropy_each_feature
    return(abs(entropy_attribute))
a_entropy = {k:ent(df,k) for k in df.keys()[:-1]}
a_entropy
```


```
{'distance': 2.8149865438356105,
 'cab_type': 4.684030774951697,
 'time_stamp': 4.814397912427494,
 'destination': 4.72424616504521,
 'source': 4.657558605497207,
 'price': 3.179014278129619e-16,
 'surge_multiplier': 5.187217119499152,
 'name': 3.2669048833897447,
 'temp': 4.72424616504521,
 'clouds': 4.72424616504521,
 'pressure': 4.72424616504521,
 'rain': 4.72424616504521,
 'humidity': 4.72424616504521,
 'wind': 4.72424616504521,
 'date': 4.941664029488899,
 'time': 4.814397912427494,
 'weekday': 5.083490165929721,
 'hour': 4.911261833962893}
```


fig:4.2.15 Entropy

Information gain:

```
def ig(e_dataset,e_attr):
    return(e_dataset-e_attr)
IG = {k:ig(entropy_node,a_entropy[k]) for k in a_entropy}
```

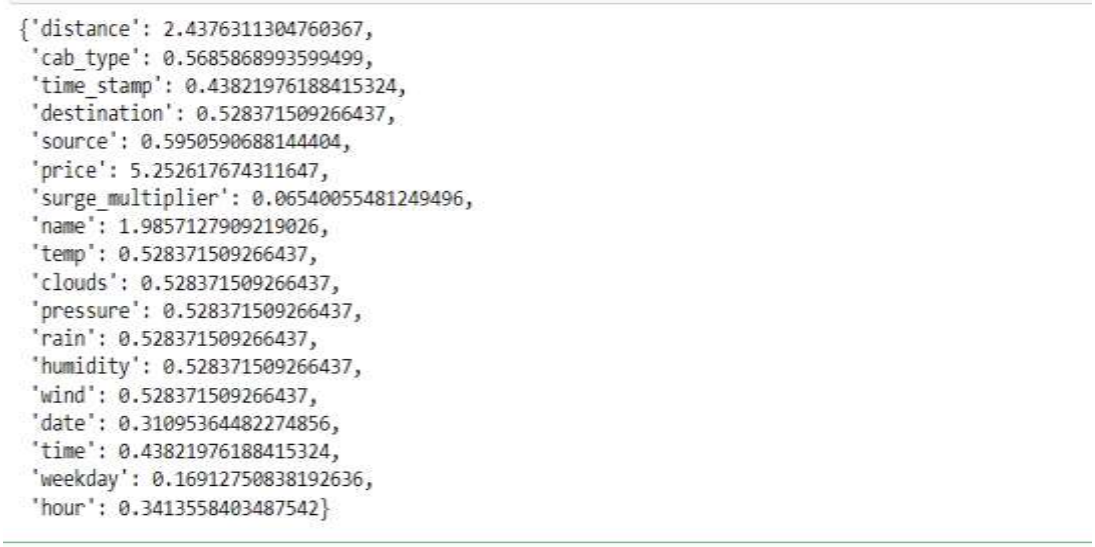


fig:4.2.16 Information gain

4.3 Statistics modelling/ Machine Learning algorithm:

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known second set is called the test data set, this set is solely used for predictions.

We used Multilinear regression as we have more than 2 independent variables and one dependent variable

IMPORTING LIBRARIES:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

considering x and y values:

```
x = df.iloc[:,:12]

y = df.iloc[:,4]
```

Column transformer :

Column Transformer is a sciket-learn class used to create and apply separate transformers for numerical and categorical data. To create transformers we need to specify the transformer object pass the list of transformations inside a tuple along with the column on which you want to apply the transformation.

One-hot encoding:

One hot encoding can be defined as the essential process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions asclassification accuracy of a model. One Hot Encoding is a common way of preprocessing categorical features for machine learning models.

```
col_trans=make_column_transformer(
   (OneHotEncoder(handle_unknown='ignore'),['source','destination']),
   remainder='passthrough')
x=col_trans.fit_transform(x)
x
```

out:

```
array([[0.0, 0.0, 0.0, ..., 1010.1382142857143, 0.004845535714285714,
        0.7494642857142857],
       [0.0, 0.0, 0.0, ..., 1010.1382142857143, 0.004845535714285714,
        0.7494642857142857],
       [0.0, 0.0, 0.0, ..., 1010.1382142857143, 0.004845535714285714,
        0.7494642857142857],
       ...,
       [0.0, 0.0, 0.0, ..., 1010.3904807692307, 0.004866346153846153,
        0.7429807692307692],
       [0.0, 0.0, 0.0, ..., 1010.3904807692307, 0.004866346153846153,
        0.7429807692307692],
       [0.0, 0.0, 0.0, ..., 1010.3904807692307, 0.004866346153846153,
        0.7429807692307692]], dtype=object)
```

Fig:4.3.1 Column Transformer

.

```
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(sparse=False)
x=ohe.fit_transform(df[['price']]
x
```

out:

```
array([[0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Fig:4.3.2 One Hot encoding

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
#shapes of splitted data
print("X_train:",x_train.shape)
print("X_test:",x_test.shape)
print("Y_train:",y_train.shape)
print("Y_test:",y_test.shape)
```

```
X_train: (944, 69)
X_test: (237, 69)
Y_train: (944,)
Y_test: (237,)
```

Fig:4.3.3 Shapes of Splitted Data

- To train the model, we have to import the Linear Regression model, which we have already created at the beginning. Use the fit method, and pass the training sets into it to train the model

linreg=LinearRegression()
linreg.fit(x_train,y_train)
ou

```
▾ LinearRegression
LinearRegression()
```

Fig:4.3.4 Importing model

y_pred=linreg.predict(x_test)

- Accuracy is calculated by using r2 score for which we got the following results as shown below

```
In [97]: Accuracy=r2_score(y_test,y_pred)*100
         print(" Accuracy of the model is %.2f" %Accuracy)

          Accuracy of the model is 97.05
```

Fig:4.3.5 Accuarcy

CHAPTER 5

RESULTS

- Here we can see that the difference between our actual and predicted value is correct

pred_df=pd.DataFrame({'Actual Value':y_test,'predicted value':y_pred,'difference':y_test-y_pred})

pred_df

	Actual Value	predicted value	difference
1100	56	56.0	2.842171e-14
351	9	9.0	-2.842171e-14
704	23	23.0	-7.105427e-15
434	18	18.0	1.065814e-14
513	9	9.0	-2.842171e-14
...	...	...	...
883	12	12.0	1.776357e-14
844	11	11.0	2.842171e-14
922	39	39.0	5.684342e-14
775	9	9.0	-2.842171e-14
689	49	49.0	1.989520e-13

237 rows × 3 columns

Fig:5.1 Predicted Values

- We will plot the scatter plot between actual values and predicted values. Use xlabel to label the x-axis and use ylabel to label the y-axis.

  plt.scatter(y_test,y_pred);

  plt.xlabel('Actual');

  plt.ylabel('Predicted');

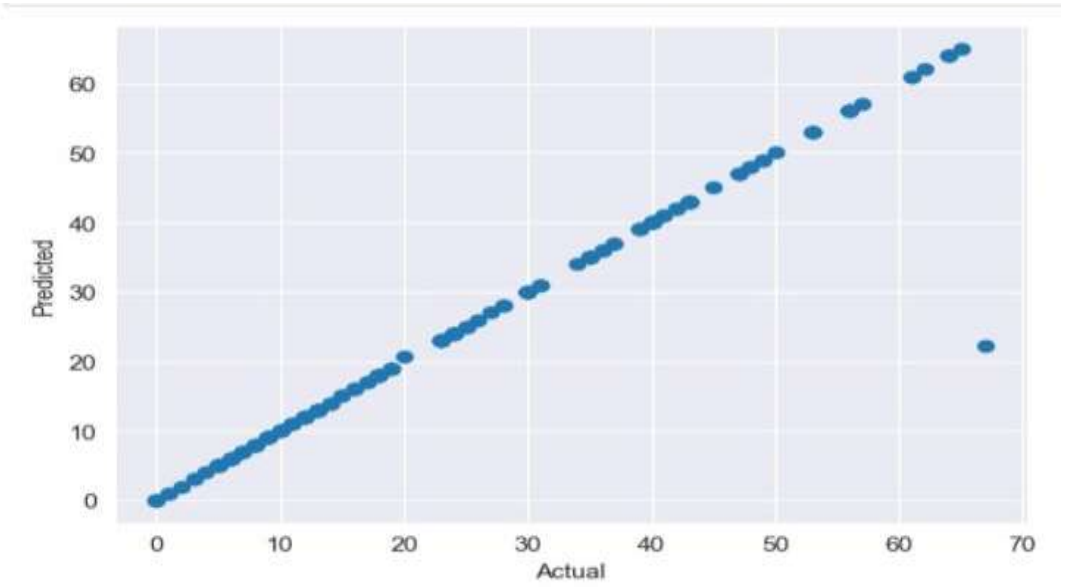


Fig:5.2 Plotting the results

- Regression plot of our model.
- A regression plot is useful to understand the linear relationship between two parameters. It creates a regression line in-between those parameters and then plots a scatter plot of those data points

  sns.regplot(x=y_test,y=y_pred,ci=None,color='red');

  out:

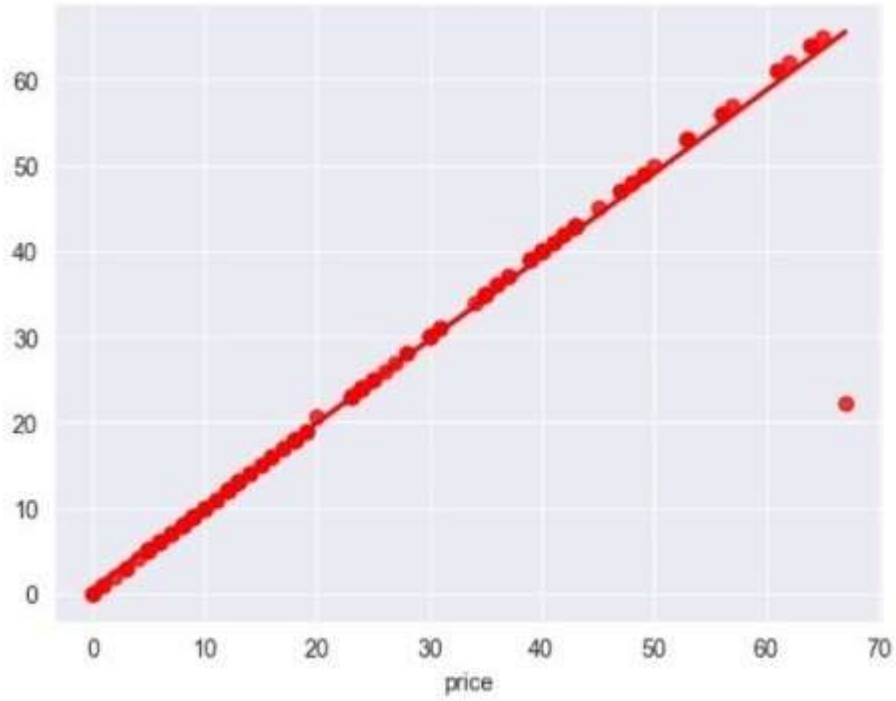


Fig:5.3 Regression plot

This is the final result in which we can see how accurate are the predicted values.

CHAPTER 6

CONCLUSION

We have created a new Linear Regression model, and we learned how to perform One-Hot Encoding and where to perform it. We used a column transformer and then trained the model, predicted the results, evaluated the model using r2_score metrics, and plotted the results. The above results shows that weather does influence uber rider ship, especially when it is little distracted from normal but not to extreme again. The insights derived from exploratory analysis and forecasting model can be very helpful in the optimal utilization of the existing resources as weather can change with time. This also says that price of uber ride may also increase if weather changes accordingly.

REFERENCES:

https://www.kaggle.com/code/rajg28/uber-fare-prediction/data

https://towardsdatascience.com/uber-price-prediction-73468d630cfc