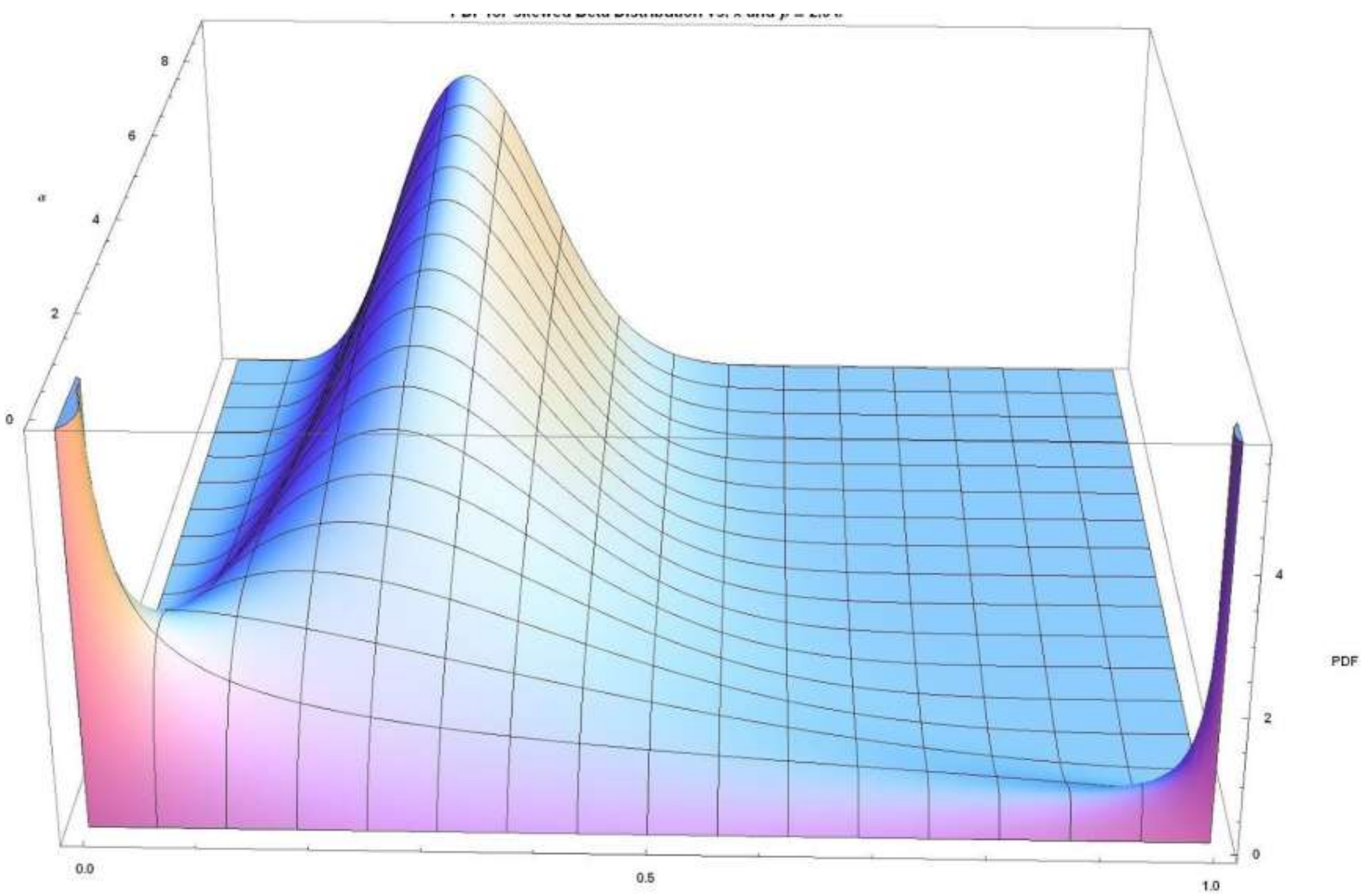
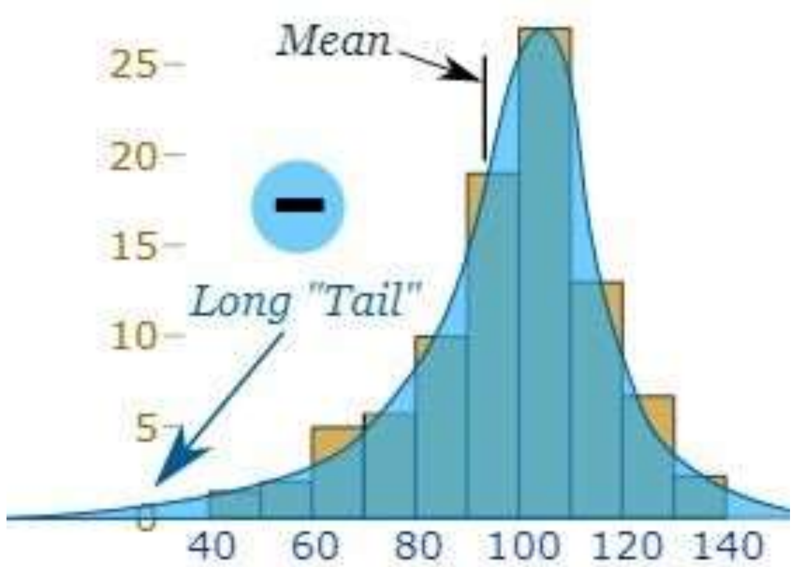
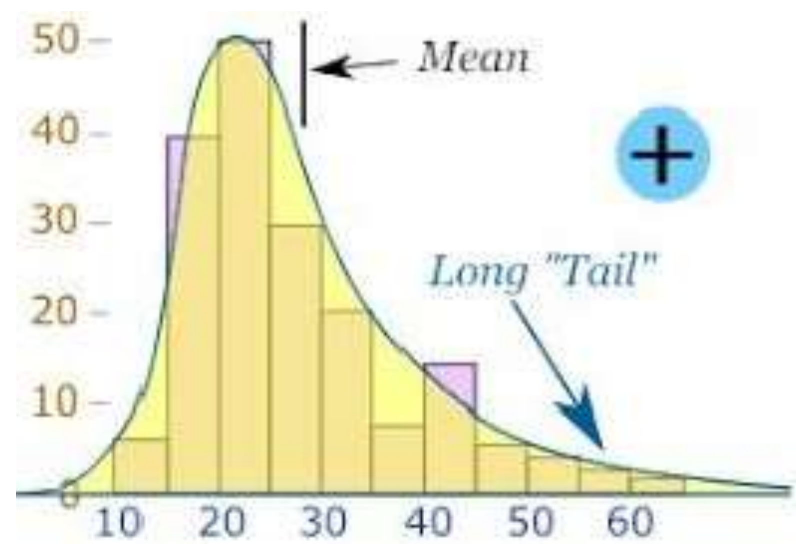


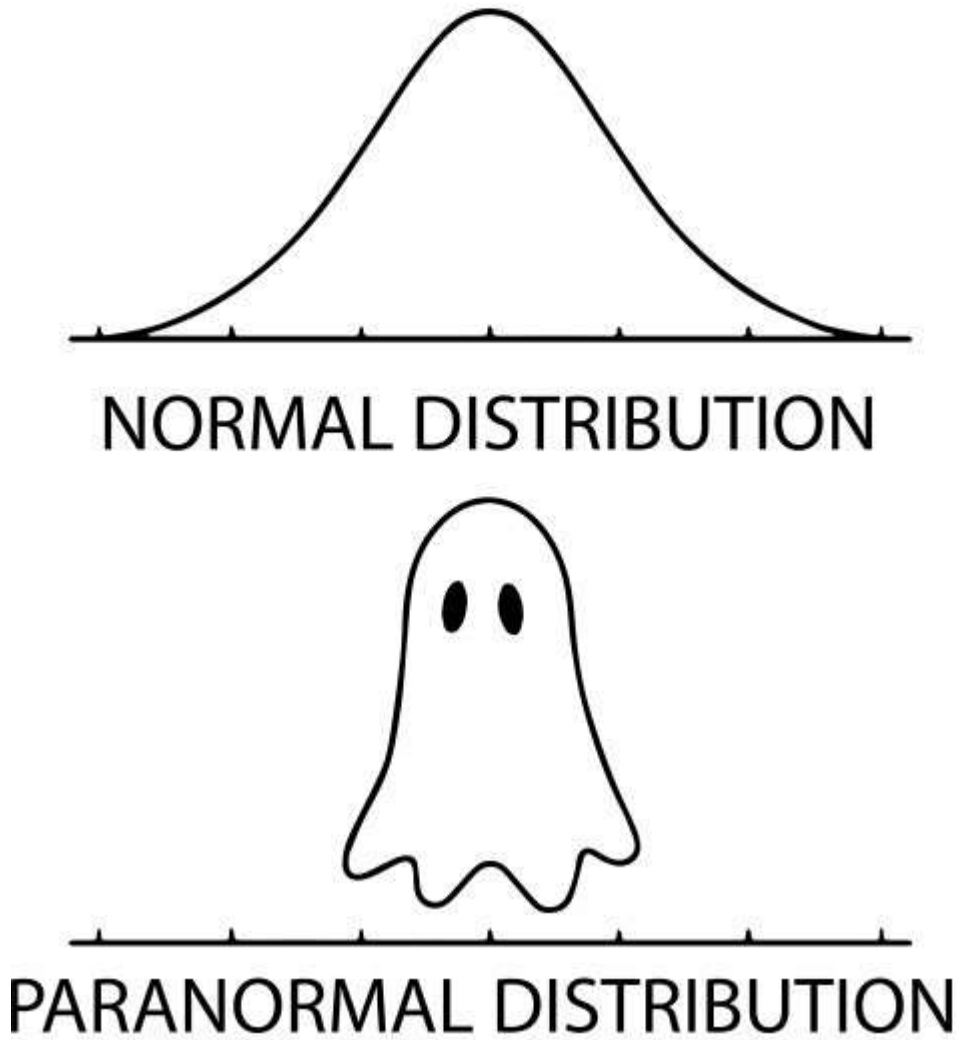




Handling Highly Skewed Data

dataebook

WHY DO WE CARE SO MUCH ABOUT NORMALITY?



Most of the parametric machine learning models like LDA, Linear Regression and any more assume that the data is normally distributed. If this assumption fails the model fails to give accurate predictions.

Everyone Learn statistics but they fail to apply But

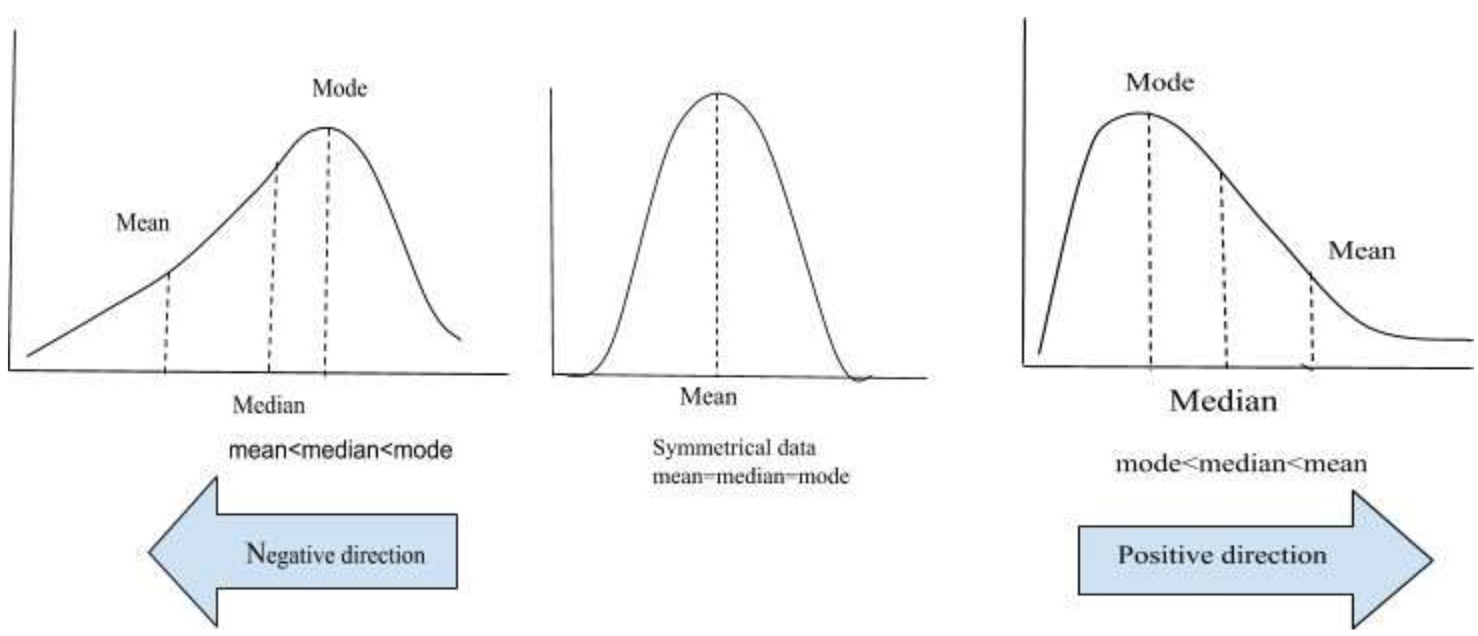
We are here to make it easy #Dataebook

WHAT IS NORMAL DISTRIBUTION?

A probability distribution with the mean 0 and standard deviation of 1 is known as standard normal distribution or Gaussian distribution. A normal distribution is symmetric about the mean and follows a bell shaped curve. And almost 99.7% of the values lie within 3 standard deviation. The mean, median and mode of a normal distribution are equal.

Handling Skewness

Skewness of a distribution is defined as the lack of symmetry. In a symmetrical distribution, the Mean, Median and Mode are equal. The normal distribution has a skewness of 0.
Skewness tell us about distribution of our data.



Skewness is of two types:

Positive skewness: When the tail on the right side of the distribution is longer or fatter, we say the data is positively skewed.

For a positive skewness mean > median > mode.

Negative skewness: When the tail on the left side of the distribution is longer or fatter, we say that the distribution is negatively skewed.

For a negative skewness mean < median < mode.

Let's have clear picture of skewness

Skewness of a distribution is defined as the lack of symmetry. In a symmetrical distribution, the Mean, Median and Mode are equal to each other. The normal distribution has a skewness of 0. Skewness tells us about where most of the values are concentrated on an ascending scale.

Now, the question is when we can say our data is moderately skewed or heavily skewed?

The thumb rule is:

- If the skewness is between -0.5 to +0.5 then we can say data is fairly symmetrical.
- If the skewness is between -1 to -0.5 or 0.5 to 1 then data is moderately skewed.
- And if the skewness is less than -1 and greater than +1 then our data is heavily skewed.

Types of Skewness

- **Positive skewness:** In simple words, if the skewness is greater than 0 then the distribution is positively skewed. The tail on the right side of the distribution will be longer or flatter. If the data is positively skewed than most of values will be concentrated below the average value of the data.
- **Negative skewness:** If the skewness is less than 0 then the distribution is negatively skewed. For negatively skewed data, most of the values will be concentrated above the average value and tail on the left side of the distribution will be longer of flatter.

What does skewness tells us?

To understand this better consider a example.

Consider house prices ranging from 100k to 1,000,000 with the average being 500,000.

If the peak of the distribution is in left side that means our data is positively skewed and most of the houses are being sold at the price less than the average.

If the peak of the distribution is in right side that means our data is negatively skewed and most of the houses are being sold at the price greater than the average.

Skewness of a data indicates the **direction and relative magnitude of a distribution's deviation from the normal distribution.** Skewness considers the extremes of the dataset rather than concentrating only on the average. Investors need to look at the extremes while judging the return from market as they are less likely to depend on the average value to work out.

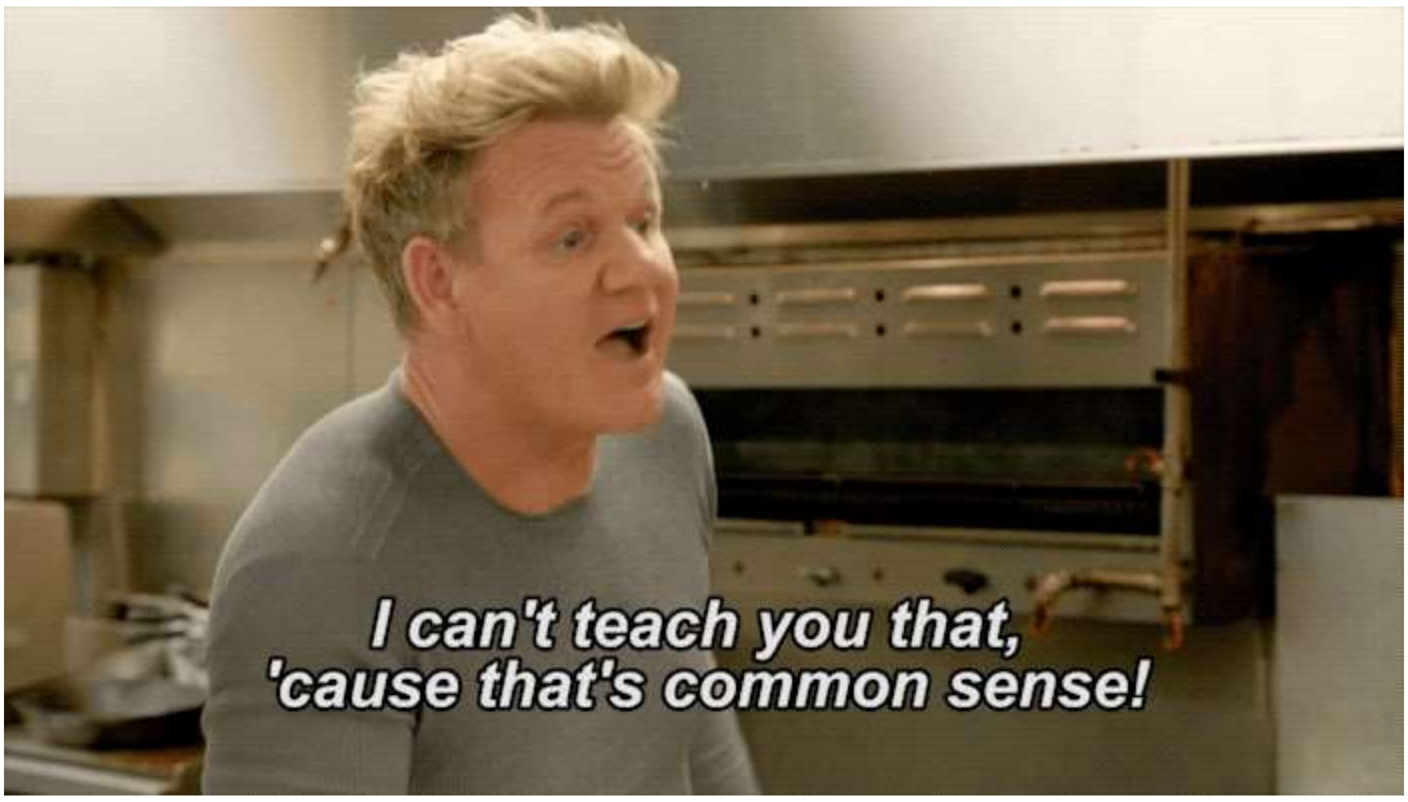
Many **models assume normal distribution** but in reality data points may not be perfectly symmetric. If the data are skewed, then this kind of **model will always underestimate the skewness risk**. The more the data is skewed the less accurate the model will be.

Here,

- skew of raw data is positive and greater than 1, right tail of the data is skewed.
- skew of raw data is negative and less than 1, left tail of the data is skewed.

Still you are not getting

Major key skill while you apply your knowledge is "Commonsense" otherwise it's just an information that you are collecting



TO Experience the same as we have learned & let's explore more to enhance our new way of learning.

All dataset used can be downloaded from this link(https://drive.google.com/open?id=1MaaYBHr5GOcmFn_YQ8mrbYV0j3Y2BVm1 (https://drive.google.com/open?id=1MaaYBHr5GOcmFn_YQ8mrbYV0j3Y2BVm1)

About Dataset

This dataset tells the details about the cars. The car Dataset contains the following attributes:

- index: Unnamed: 0(index values)
- price: The sale price of the vehicle in the ad
- brand: The brand of car
- model: model of the vehicle
- year: The vehicle registration year
- title_status: This feature included binary classification, which are clean title vehicles and salvage insurance
- mileage: miles traveled by vehicle
- color: Color of the vehicle
- vinThe: vehicle identification number is a collection of 17 characters (digits and capital letters)
- lot: A lot number is an identification number assigned to a particular quantity or lot of material from a single manufacturer. For cars, a lot number is combined with a serial number to form the Vehicle Identification Number.
- state: The location in which the car is being available for purchase
- country: The location in which the car is being available for purchase
- condition: Time

In [3]:
```python
import pandas as pd
import numpy as np
```

In [4]:
```python
#reading the dataset
data = pd.read_csv(r'cars_datasets.csv')
```

In [5]:
```python
#look at dataset
data.head()
```

Out[5]:

	Unnamed: 0	price	brand	model	year	title_status	mileage	color	vin	lot	state	coun
0	0	6300	toyota	cruiser	2008	clean vehicle	274117.0	black	jtezu11f88k007763	159348797	new jersey	usa
1	1	2899	ford	se	2011	clean vehicle	190552.0	silver	2fmdk3gc4bbb02217	166951262	tennessee	usa
2	2	5350	dodge	mpv	2018	clean vehicle	39590.0	silver	3c4pdcgg5jt346413	167655728	georgia	usa
3	3	25000	ford	door	2014	clean vehicle	64146.0	blue	1ftfw1et4efc23745	167753855	virginia	usa
4	4	27700	chevrolet	1500	2018	clean vehicle	6654.0	red	3gcpcrec2jg473991	167763266	florida	usa

In [6]:
```python
#Removing the unnamed: 0 column
data = data.drop(['Unnamed: 0'], axis = 1)
```

In [7]:
```python
data.head()
```

Out[7]:

	price	brand	model	year	title_status	mileage	color	vin	lot	state	country	conditio
0	6300	toyota	cruiser	2008	clean vehicle	274117.0	black	jtezu11f88k007763	159348797	new jersey	usa	10 days left
1	2899	ford	se	2011	clean vehicle	190552.0	silver	2fmdk3gc4bbb02217	166951262	tennessee	usa	6 days lef
2	5350	dodge	mpv	2018	clean vehicle	39590.0	silver	3c4pdcgg5jt346413	167655728	georgia	usa	2 days lef
3	25000	ford	door	2014	clean vehicle	64146.0	blue	1ftfw1et4efc23745	167753855	virginia	usa	22 hours left
4	27700	chevrolet	1500	2018	clean vehicle	6654.0	red	3gcpcrec2jg473991	167763266	florida	usa	22 hours left

In [6]:
```python
#Exploring the dataset information
data.info()
```

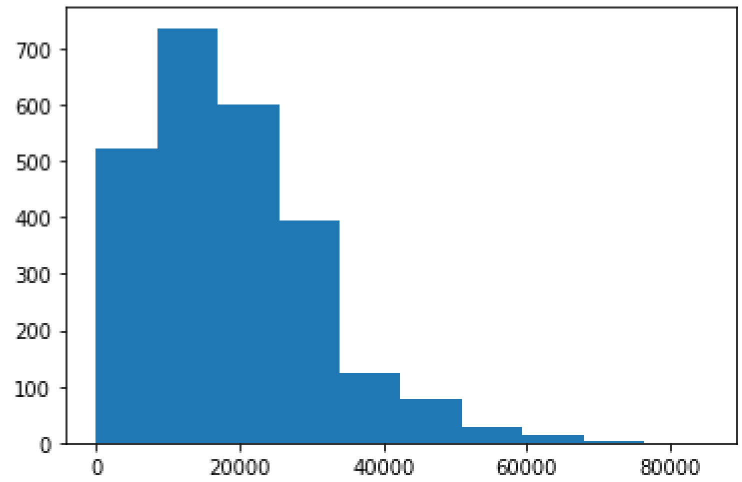
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2499 entries, 0 to 2498
Data columns (total 12 columns):
price           2499 non-null int64
brand           2499 non-null object
model           2499 non-null object
year            2499 non-null int64
title_status    2499 non-null object
mileage         2499 non-null float64
color           2499 non-null object
vin             2499 non-null object
lot             2499 non-null int64
state           2499 non-null object
country         2499 non-null object
condition       2499 non-null object
dtypes: float64(1), int64(3), object(8)
memory usage: 234.4+ KB
```

In [7]:
```python
#checking dimension of dataset
data.shape
```

Out[7]:
```
(2499, 12)
```

In [9]:
```python
#importing library for visualizing dataset and plotting the  histogram for  price attributes
import seaborn as sns
data['price'].hist(grid = False)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee79edb38>
```



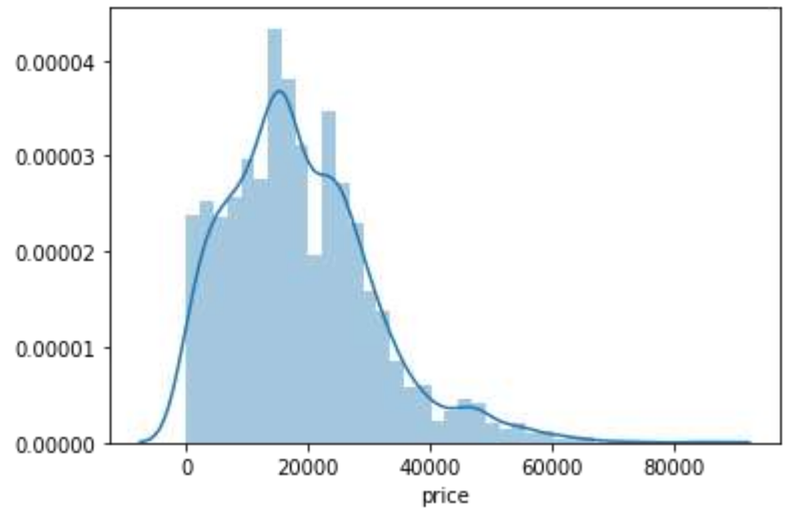
In [10]:
```python
# Checking the skewness of Price column of dataset
data['price'].skew()
```

Out[10]:
```
0.9227307836499805
```

In [11]:
```python
#density plot
sns.distplot(data['price'], hist = True)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8a0a048>
```



Note: As we can see the skewed values lies between the 0.5 to 1 range. so, data is moderately skewed and right skewed(but it's fine to train the model with it). Let's explore another attribute.

In [12]:
```python
# checking the skewness of mileage column of dataset
data['mileage'].skew()
```

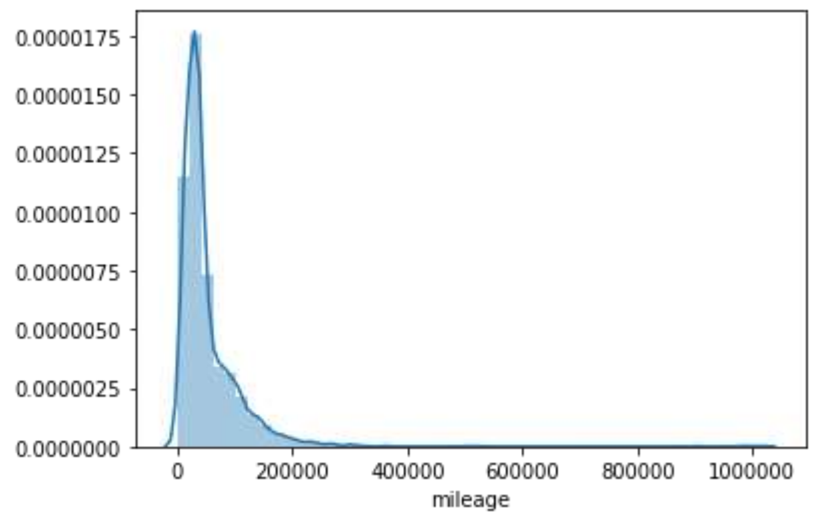
Out[12]:

```
7.0793210165347915
```

In [13]:
```python
sns.distplot(data['mileage'], hist = True)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8b2fa20>
```



Note: As we can see the skewed values lies between -1 and greater than +1 then our data is heavily skewed. so, data is heavily skewed. data['mileage'] is right skewed by looking at the graph and skewed values.

How to handle these skewed data?

Transformation

In data analysis transformation is the replacement of a variable by a function of that variable: for example, replacing a variable x by the square root of x or the logarithm of x. In a stronger sense, a transformation is a replacement that changes the shape of a distribution or relationship.

Steps to do transformation

1. Draw a graph(histogram and density plot) of the data to see how far patterns in data match the simplest ideal patterns.

2. check the range the data. Because Transformations will have little effect if the range is small.

3. check the skewness by statistical methods(decide right and left skewness).

4. apply the methods (explained in detail below) to handle the skewness based on the skewed values.

Reasons for using transformations

There are many reasons for transformation.

```
1. Convenience
2. Reducing skewness
3. Equal spreads
4. Linear relationships
5. Additive relationships
```

1. Convenience: A transformed scale may be as natural as the original scale and more convenient for a specific purpose. for example- percentage rather than the original data.

2. Reducing skewness: A transformation may be used to reduce skewness. A distribution that is symmetric or nearly so is often easier to handle and interpret than a skewed distribution.

```
- To handle the right skewness, we use:
        - logarithms (best for it)
        - roots[square root and cube root] (good)
        - reciprocals (weak)

- To handle left skewness, we use:
        - squares
        - cubes
        - higher powers.
```

3. Equal spreads: A transformation may be used to produce approximately equal spreads, despite marked variations in level, which again makes data easier to **handle and interpret**. Each data set or subset having about the same spread or variability is a condition called **homoscedasticity** and it's opposite is called **heteroscedasticity**

4. Linear relationships: When looking at relationships between variables, it is often far easier to think about patterns that are approximately linear than about patterns that are highly curved. This is vitally important when using linear regression, which amounts to fitting such patterns to data.

5. Additive relationships: Relationships are often easier to analyses when additive rather than (say) multiplicative. So

```
y = a + bx
```

In which two terms **a** and **bx** are added is easier to deal with, than

```
y = ax^b
```

In which two terms **a** and **x^b** are multiplied. Additivity is a vital issue in analysis of variance (in Stata, anova, oneway, etc.).

To Handle Right Skewedness

1. log Transformation

The log transformation is widely used in research to deal with skewed data. It is the best method to handle the right skewed data.

Why log?

- The normal distribution is widely used in basic research studies to model continuous outcomes. Unfortunately, the symmetric bell-shaped distribution often does not adequately describe the observed data from research projects. Quite often data arising in real studies are so skewed that standard statistical analyses of these data yield invalid results.
- Many methods have been developed to test the normality assumption of observed data. When the distribution of the continuous data is non-normal, transformations of data are applied to make the data as **"normal"** as possible and, thus, **increase the validity of the associated statistical analyses.**
- Popular use of the log transformation is to **reduce the variability of data**, especially in data sets that include outlying observations. Again, contrary to this popular belief, log transformation can often **increase – not reduce – the variability of data whether or not there are outliers.**

Why not?

- Using transformations in general and log transformation in particular can be quite problematic. If such an approach is used, the researcher must be mindful about its limitations, particularly when interpreting the relevance of the analysis of transformed data for the hypothesis of interest about the original data.

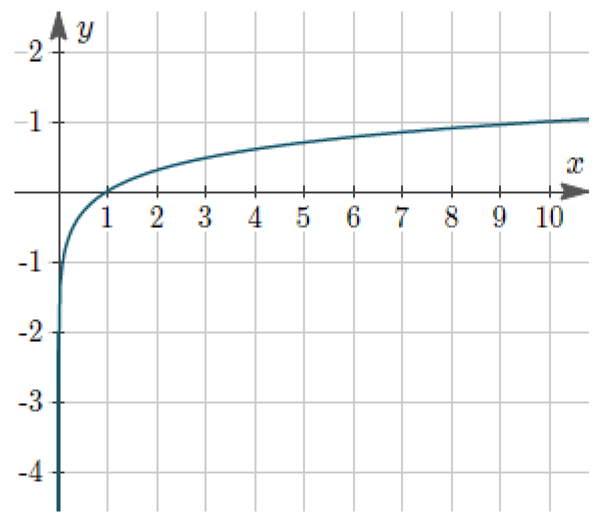
In [12]:
```
#performing the log transformation using numpy
log_mileage = np.log(data['mileage'])
log_mileage.head(15)
```

Out[12]:
```
0     12.521310
1     12.157680
2     10.586332
3     11.068917
4      8.802973
5     10.726807
6     11.912037
7     10.065819
8      9.145375
9     11.057503
10    11.588552
11    10.587846
12    10.039285
13    11.839708
14    11.520467
Name: mileage, dtype: float64
```

In [15]:
```
#checking the skewness after the log-transformation
log_mileage.skew()
```

Out[15]:
```
nan
```

It's giving us **nan** because there are some values as the zero. In log transformation, it deals with only the positive and negative numbers not with zero. The log is range in between (- infinity to infinity) but greater or less than zero. For better understanding you can check the log graph below:



Note: In the graph, the line at zero is deviated towards the positive infinity. so, If you are getting zeros inside the data, refer root Transformation.

2. Root Transformation

2.1 Square root Transfomation

- The square root means **x** to **x^(1/2)** = **sqrt(x)**, is a transformation with a moderate effect on distribution shape. it is **weaker than the logarithm and the cube root**.
- It is also used for reducing right skewness, and also has the advantage that it can be **applied to zero values**.
- Note that the square root of an area has the units of a length. It is commonly applied to counted data, especially if the values are mostly rather small.

In [13]:
```
#calculating the square root for data['mileage'] column
sqrt_mileage = np.sqrt(data['mileage'])
sqrt_mileage.head(15)
```

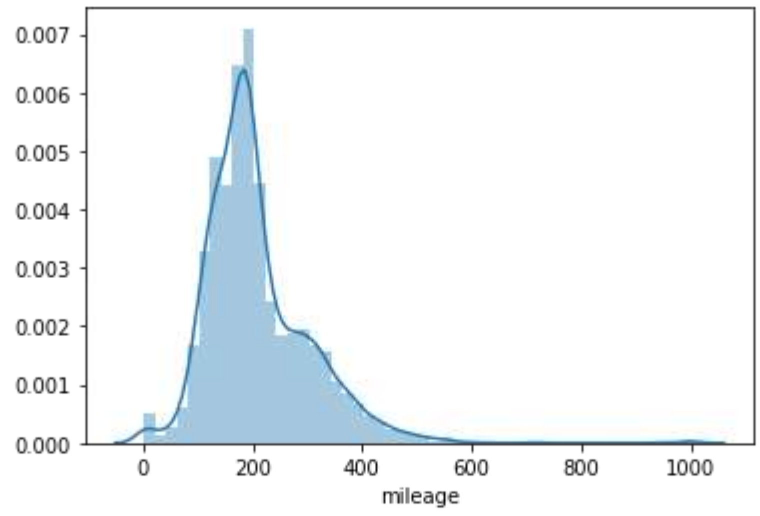
Out[13]:
```
0     523.561840
1     436.522623
2     198.972360
3     253.270606
4      81.572054
5     213.450228
6     386.069942
7     153.378617
8      96.803926
9     251.829307
10    328.414372
11    199.123078
12    151.357193
13    372.357355
14    317.422431
Name: mileage, dtype: float64
```

In [19]:
```
#calculation skewness after calculating the square root & we can observe change in the value of ske
wness
sqrt_mileage.skew()
```

Out[19]:
```
1.6676282633339148
```

In [20]:
```
#visualising by density plot
sns.distplot(sqrt_mileage, hist = True)
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8a5d710>
```



Note: In previous case we got the **nan** because of zero, but the square root transformation has reduced the skewed values **from 7.07 to 1.66.** Which is nearer to zero compare to 7.07.

2.2 cube root Transformation

- The cube root means **x to x^(1/3)**. This is a fairly strong transformation with a substantial effect on distribution shape,
- It is **weaker than the logarithm but stronger than the square root** transformation.
- It is also used for reducing right skewness, and has the advantage that it can be **applied to zero and negative values**. Note that the cube root of a volume has the units of a length. It is commonly applied to rainfall data.

In [14]:
```
#calculating the cube root for the column data['mileage'] column

cube_root_mileage = np.cbrt(data['mileage'])
cube_root_mileage.head(15)
```

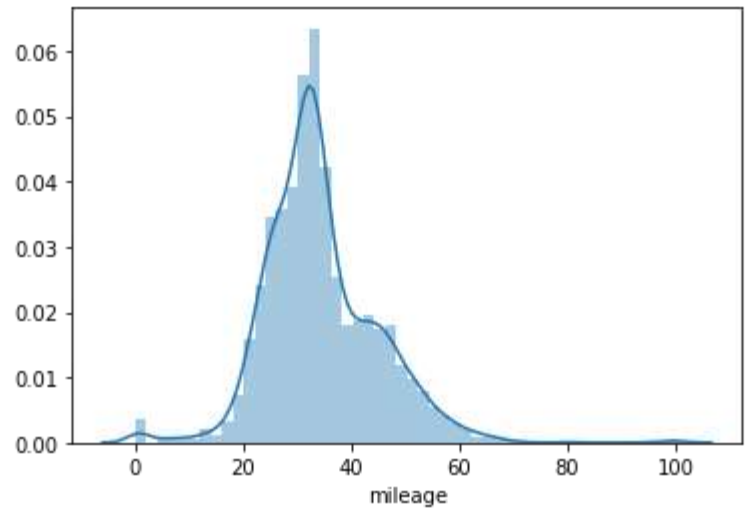
Out[14]:
```
0     64.959896
1     57.544590
2     34.082269
3     40.030394
4     18.808793
5     35.716132
6     53.020521
7     28.653425
8     21.082817
9     39.878381
10    47.600857
11    34.099478
12    28.401114
13    51.757500
14    46.532717
Name: mileage, dtype: float64
```

In [22]:
```
#calculation skewness after calculating the cube root
cube_root_mileage.skew()
```

Out[22]:
```
0.6866069687334178
```

In [23]:
```
#visualising by density plot
sns.distplot(cube_root_mileage, hist = True)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8cce358>
```



Note: In logarithm transformation we got the **nan** because of zero, and in the square root transformation it has reduced the skewed values from **7.07 to 1.66.** but now in cube root transformation the skewed values **reduced to 0.68.** and it is very much near to zero compare to 1.66 and 7.07.

3. Reciprocals Transformation

- The reciprocal, x to 1/x, with its sibling the negative reciprocal, **x to -1/x**, is a very strong transformation with a drastic effect on distribution shape.
- It cannot be applied to zero values. Although it can be applied to **negative values**, it is not useful unless all values are positive.

For Example: we might want to multiply or divide the results of taking the reciprocal by some constant, such as 100 or 1000, to get numbers that are easy to manage, but that it has no effect on skewness or linearity.

In [15]:
```
#calculating the reciprocal for the column data['mileage'] column
recipr_mileage = np.reciprocal(data['mileage'])
recipr_mileage.head(15)
```

```
c:\users\chintan\appdata\local\programs\python\python37\lib\site-packages\pandas\core\series.py:679: RuntimeWarning: divide by z
ero encountered in reciprocal
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[15]:
```
0     0.000004
1     0.000005
2     0.000025
3     0.000016
4     0.000150
5     0.000022
6     0.000007
7     0.000043
8     0.000107
9     0.000016
10    0.000009
11    0.000025
12    0.000044
13    0.000007
14    0.000010
Name: mileage, dtype: float64
```

In [25]:
```
recipr_mileage.skew()
```

Out[25]:
```
nan
```

Note: It's giving output as **nan** because there are some values as the zero. In reciprocal transformation, **it's good deal with negative numbers not with zero.**

To Handle Left skewness

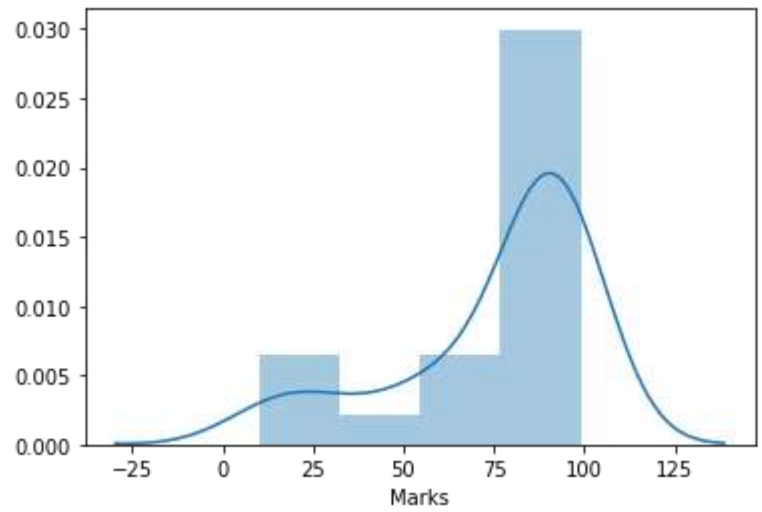
In [16]:
```
#let's create small dataset & have alook on left skewness
Data = [['sameer',10],['pankaj',20],['sam',30],['Hemant',48],['vivek',62],['ram',87],['suman', 93],
['anup',85],['mohit',60],['sandeep',75],['ajeet',84],
        ['yash',90], ['sam', 99], ['deepak', 92],['vikas', 99],['rajiv',95],['sivam', 94],['akash',
89],['vinod',90],['Kundan',87],['Abhay',99]]

df = pd.DataFrame(Data, columns=['Name','Marks'] )  #converting in dataframe
df.head(10)
```

Out[16]:

	Name	Marks
0	sameer	10
1	pankaj	20
2	sam	30
3	Hemant	48
4	vivek	62
5	ram	87
6	suman	93
7	anup	85
8	mohit	60
9	sandeep	75

```
In [27]: #ploting the Density & histogram plot
         import seaborn as sns
         sns.distplot(df['Marks'], hist = True)
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1dee8deb630>
```




```
In [28]: #checking the skewness
         df['Marks'].skew()
```

```
Out[28]: -1.4076657771292151
```

Note: If the skewness is less than -1 and greater than +1 then our data is heavily skewed. Our data is left skewed here; the skewed value is less than -1. Let's try to make it symmetric.

1. squares Transformation

The square, **x to x^2,** has a moderate effect on distribution shape and it could be used to **reduce left skewness.** Squaring usually makes sense only if the variable concerned is zero or positive, given that (-x)^2 and x^2 are identical.

```
In [17]: #calculating the square for the column df['Marks'] column
         Square_marks = np.square(df['Marks'])
         Square_marks.head(10)
```

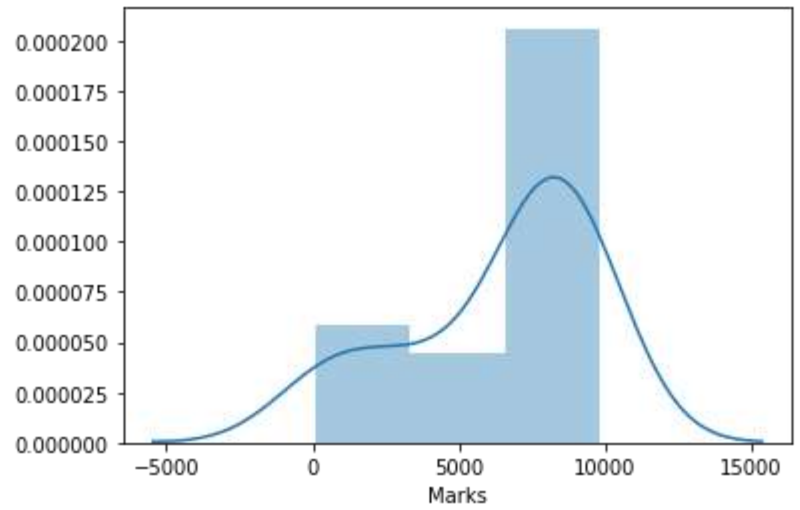
```
Out[17]: 0     100
         1     400
         2     900
         3    2304
         4    3844
         5    7569
         6    8649
         7    7225
         8    3600
         9    5625
         Name: Marks, dtype: int64
```

```
In [30]: #checking the skewness
         Square_marks.skew()
```

```
Out[30]: -0.9341854225868288
```

```
In [31]: #plotting the density and histogram plot
         sns.distplot(Square_marks, hist=True)
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1dee8e72f98>
```



Note: After applying the square Transformation, we are getting the skewed value as 0.93. If the skewed value lies in between -1 to 0.5 then data is moderately skewed. Let’s try some other transformation.

2. Cubes Transformation

The cube, x to x^3, has a better effect on distribution shape than squaring and it could be used to reduce left skewness.

```
In [18]: #calculating the Cubes for the column df['Marks'] column
         cube_marks = np.power(df['Marks'], 3)
         cube_marks.head(10)
```

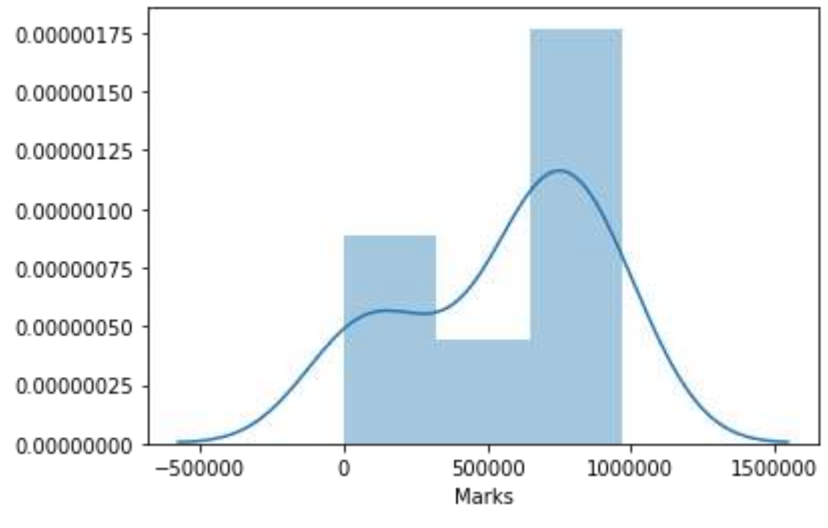
```
Out[18]: 0      1000
         1      8000
         2     27000
         3    110592
         4    238328
         5    658503
         6    804357
         7    614125
         8    216000
         9    421875
         Name: Marks, dtype: int64
```

```
In [33]: #calculating the skewness
         cube_marks.skew()
```

```
Out[33]: -0.6133662709032679
```

```
In [34]: #plotting the density and histogram plot
         sns.distplot(cube_marks, hist= True)
```

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1dee8ee2240>
```



Note: After applying the cube transformation, the skewed value is -0.6 , and If the skewed value lies in between -1 to 0.5 then data is moderately skewed. Let’s try some other transformation.

3. higher powers

When simple transformation like square and cubes doesn't reduce the skewness in the data distribution, we can use higher powers to transform to data. It is only useful in left skewness.

In [20]:
```
#calculating the Higher power(power = 4) for the column df['Marks'] column
higher_power_4 = np.power(df['Marks'], 4)
higher_power_4.head(15)
```

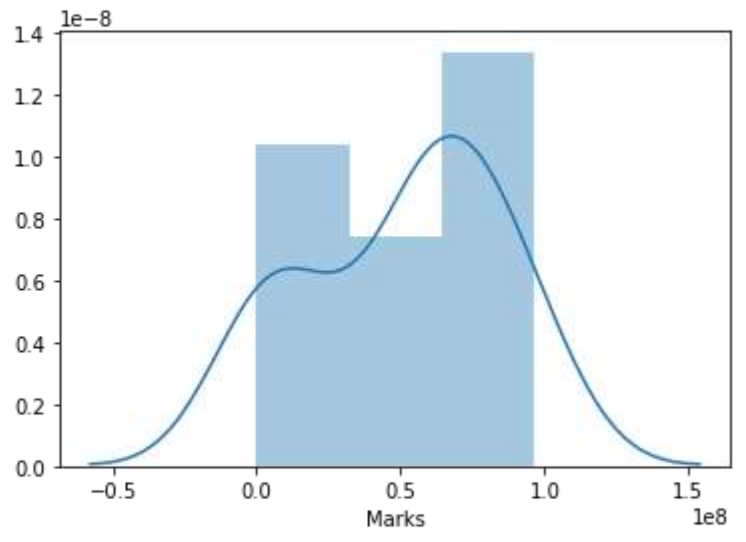
Out[20]:
```
0        10000
1       160000
2       810000
3      5308416
4     14776336
5     57289761
6     74805201
7     52200625
8     12960000
9     31640625
10    49787136
11    65610000
12    96059601
13    71639296
14    96059601
Name: Marks, dtype: int64
```

In [36]:
```
#calculating the skewness
higher_power_4.skew()
```

Out[36]:
```
-0.3563776896040546
```

In [37]:
```
#plotting the density and histogram
sns.distplot(higher_power_4, hist = True)
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8f03668>
```



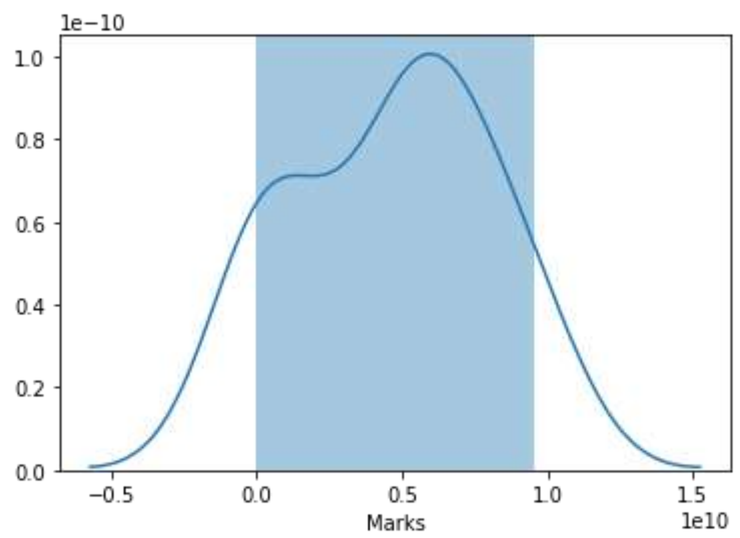
Note: After applying the higher power (power = 4) the skewness is changed from -1.4 to -0.3.If the skewness is between -0.5 to +0.5 then we can say data is fairly symmetrical. So, finally we have got the best result and we got the skew value as -0.3.

Incase if we would not have got this skew value still after applying these many powers, we can increase the power to get better result. You can check out the below for better understanding:

In [38]:
```
#applying the higher power(power = 5) and calculating the skewness
higher_power_5 = np.power(df['Marks'], 5)
higher_power_5.skew()
```

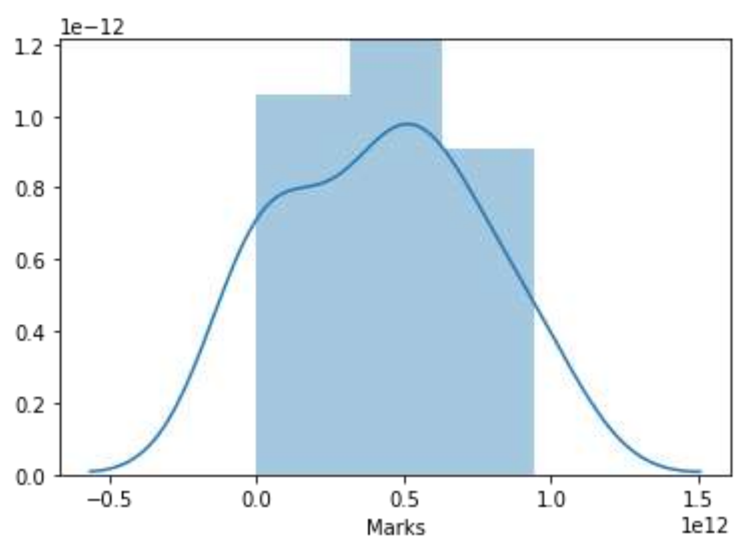
Out[38]:
```
-0.12781688683710232
```

In [39]:
```
#plotting the density and histogram
sns.distplot(higher_power_5, hist = True)
```
Out[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8fb60b8>
```



In [40]:
```
#applying the higher power(power = 5) and calculating the skewness
higher_power_6 = np.power(df['Marks'], 6)
higher_power_6.skew()
```
Out[40]:
```
0.08406219634567366
```

In [41]:
```
#plotting the density and histogram
sns.distplot(higher_power_6, hist= True)
```
Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dee8fb63c8>
```



Note: Finally, we have got the skewed value as 0.08 (almost 0), and we can see the data is not symmetrically distributed.

Let's look into some more examples

In [25]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import skew, skewtest, norm
from scipy.stats import kurtosis
```

In [26]:
```
data1=pd.read_csv('house1.csv')
```

In [27]:
```
data1.head(2)
```
Out[27]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0

2 rows × 81 columns

Let's check the distribution of the "SalePrice"

In [28]:
```
data1['SalePrice'].describe()
```

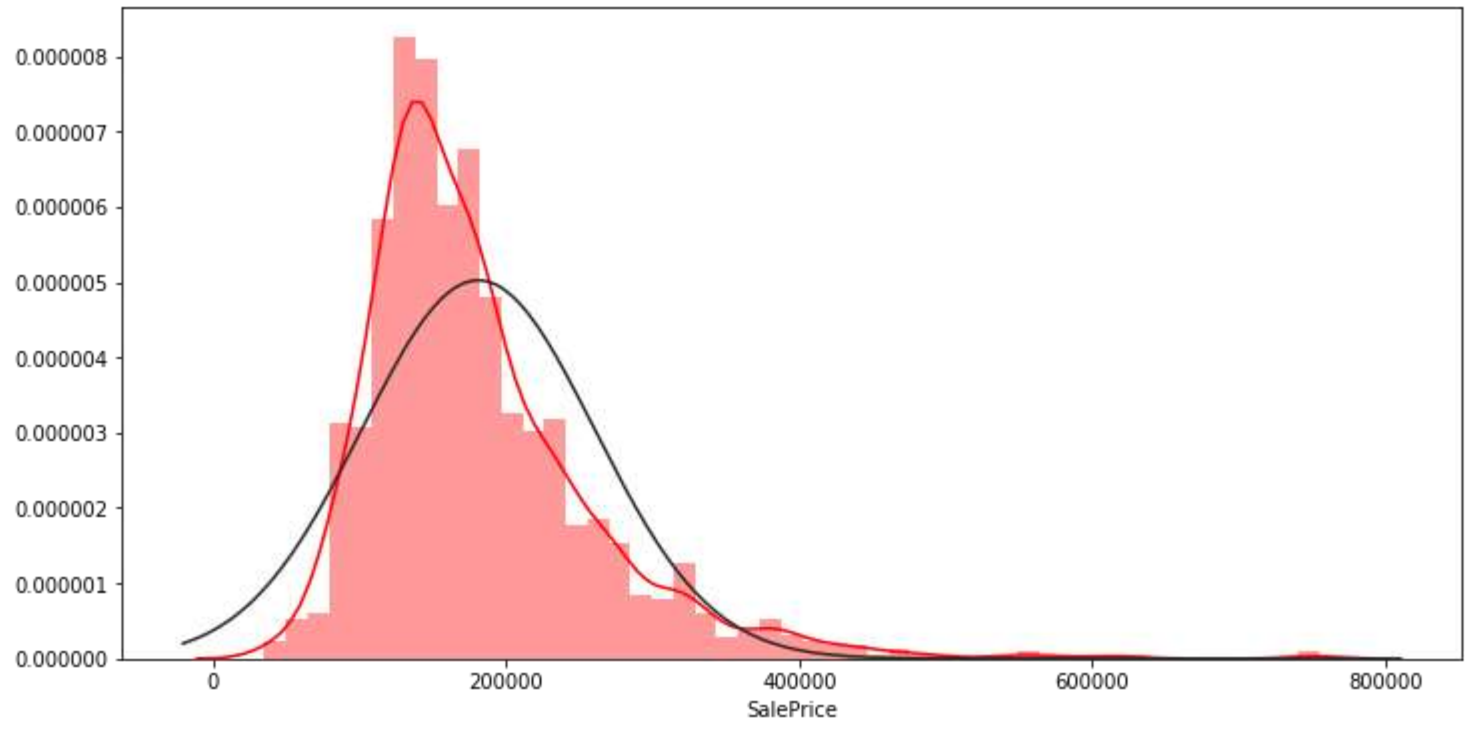
Out[28]:
```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

Here we can see that Mean (180921) is greater than the median(163000) and the maximum is 3.5 times the 75%. (The distribution is positively skewed).

We can say that most of the house prices are below the average.

Let's plot and check

In [29]:
```
#Plot and check the distribution
plt.figure(figsize=(12,6))
sns.distplot(data1['SalePrice'],fit=norm, color ="r")
plt.show()
```



The histogram confirms that our dataset is positively skewed.

Now let's check the measure of skewness and kurtosis

In [30]:
```
print("Skew of raw data: %f" % data1['SalePrice'].skew()) #check skewness
print("Kurtosis of raw data: %f" % kurtosis(data1['SalePrice'],fisher = False)) #check kurtosis
```

```
Skew of raw data: 1.882876
Kurtosis of raw data: 9.509812
```

Let's work with highly kurtosis data.

Note::In this notebook we have focused on skewed dataset & soon how to handle highly skewed dataset will be released soon

Here, skew of raw data is positive and greater than 1, and kurtosis is greater than 3, right tail of the data is skewed. So, our data in this case is positively skewed and leptokurtic.

Note- If we are keeping 'fisher=True', then kurtosis of normal distribution will be 0. Similarly, kurtosis >0 will be leptokurtic and kurtosis < 0 will be Platykurtic

Log Transformation

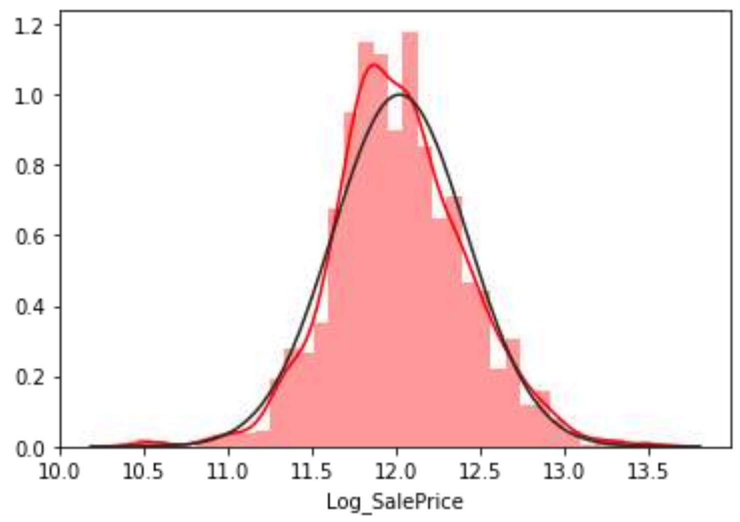
Logarithm is defined only for positive values so we can't apply log transformation on 0 and negative numbers.

Logarithmic transformation is a convenient means of transforming a highly skewed variable into a more normalized dataset. When modeling variables with non-linear relationships, the chances of producing errors may also be skewed negatively. Using the logarithm of one or more variables improves the fit of the model by transforming the distribution of the features to a more normally-shaped bell curve.

In [31]:
```
#Log transformation
data1['Log_SalePrice'] = np.log(data1['SalePrice'])

#check distribution,skewness and kurtosis
sns.distplot(data1['Log_SalePrice'], fit=norm,color ="r")
print("Skew after Log Transformation: %f" % data1['Log_SalePrice'].skew())
print("Kurtosis after Log Transformation: %f" % kurtosis(data1['Log_SalePrice'],fisher = False))
data1['Log_SalePrice'].describe()
```

```
Skew after Log Transformation: 0.121335
Kurtosis after Log Transformation: 3.802656
```

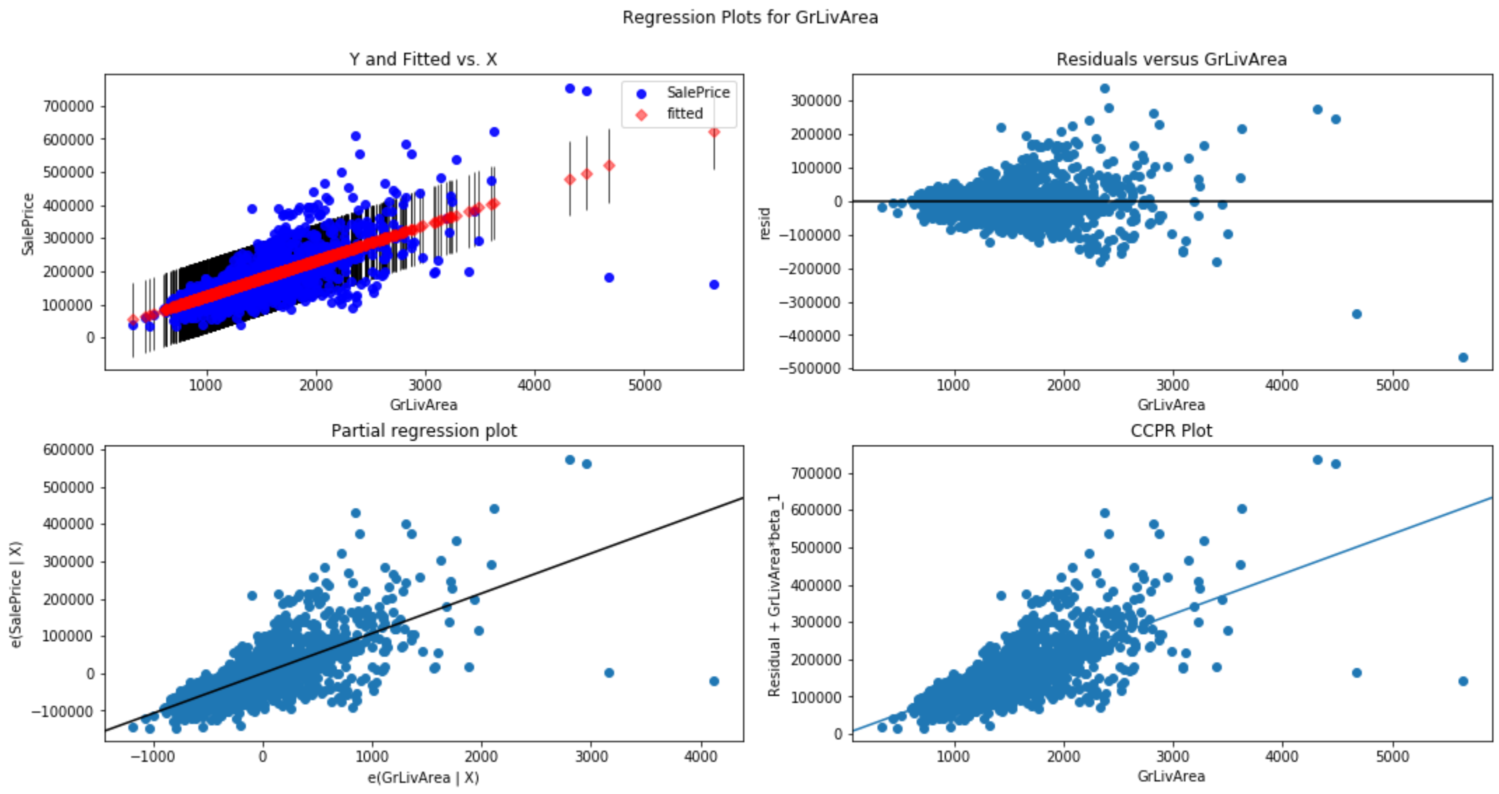
Out[31]:
```
count    1460.000000
mean       12.024051
std         0.399452
min        10.460242
25%        11.775097
50%        12.001505
75%        12.273731
max        13.534473
Name: Log_SalePrice, dtype: float64
```



Now if you look the distribution it is close to normal distribution. We have also reduced the skewness and the kurtosis.

Let's apply a linear regression model and check how well model performs before and after we apply log transformation.

In [33]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols

f = 'SalePrice~GrLivArea'
model = ols(formula=f, data=data1).fit()

fig = plt.figure(figsize =(15,8))
fig = sm.graphics.plot_regress_exog(model, 'GrLivArea', fig=fig)
```



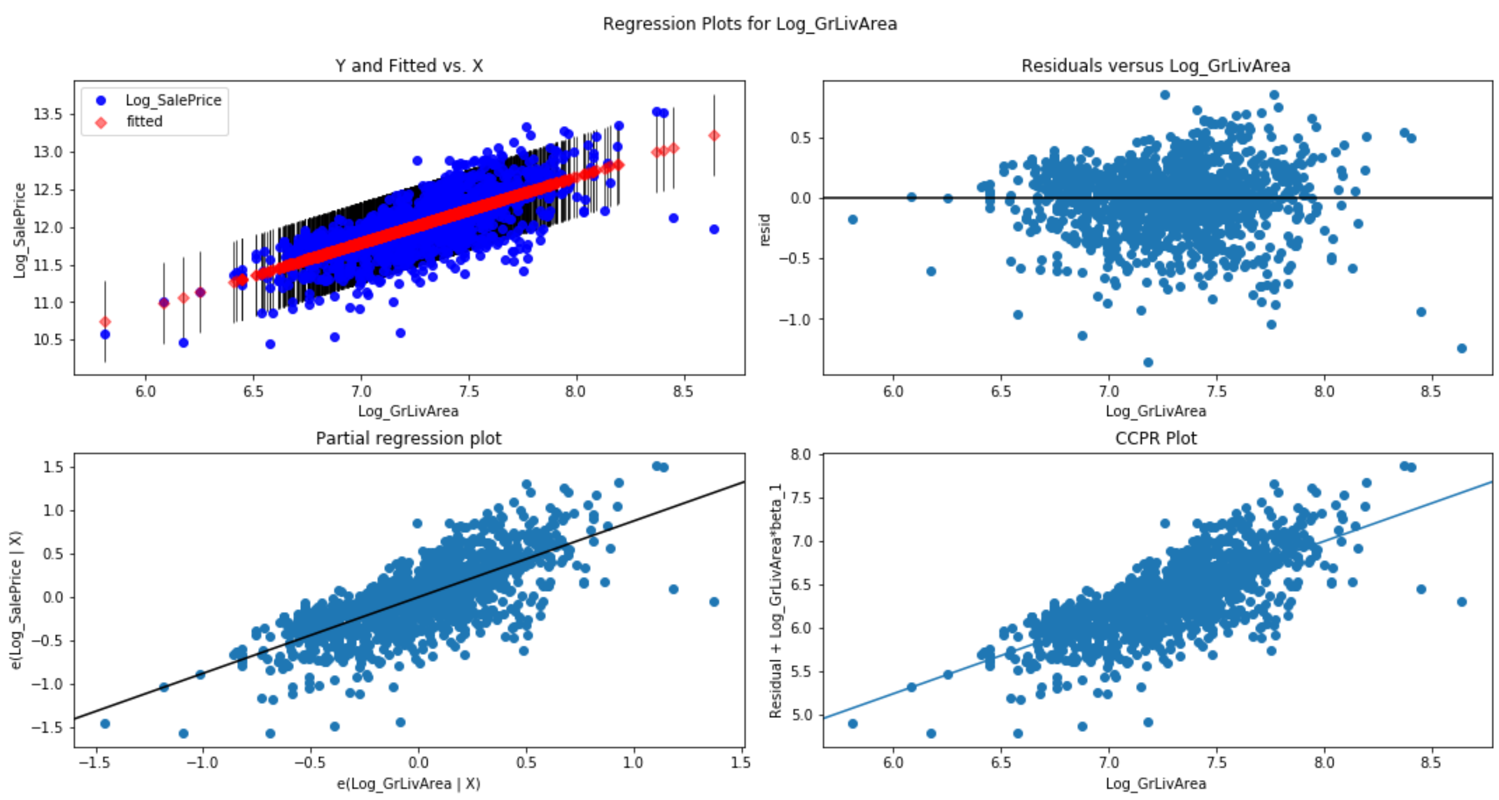
We can notice that it has a cone shape where the data points essentially scatter off as we increase in GrLivArea.

Apply Log on GrLivArea

In [34]:
```
#Log transformation on GrLivArea

data1['Log_GrLivArea'] = np.log(data1['GrLivArea'])
```

Apply Linear Regression on Log transformed SalePrice and GrLivArea

In [35]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols

f = 'Log_SalePrice~Log_GrLivArea'
model = ols(formula=f, data=data1).fit()

fig = plt.figure(figsize =(15,8))
fig = sm.graphics.plot_regress_exog(model, 'Log_GrLivArea', fig=fig)
```



We can now see the relationship as a percent change. By applying the logarithm to the variables, there is a much more distinguished and or adjusted linear regression line through the base of the data points, resulting in a better prediction model.

Finally!!! You have learned a lot & we wish you will be feeling like