In [3]:

```
import pandas as pd
```

In [19]:

```
!pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.0.2-cp38-cp38-manylinux_2_17_x86_64.manyl
inux2014_x86_64.whl (26.7 MB)
     |████████████████████████████████| 26.7 MB 87 kB/s
Requirement already satisfied: numpy>=1.14.6 in ./lib/python3.8/site-p
ackages (from scikit-learn) (1.22.0)
Collecting scipy>=1.1.0
  Downloading scipy-1.7.3-cp38-cp38-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl (39.3 MB)
     |████████████████████████████████| 39.3 MB 2.5 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.0.0-py3-none-any.whl (14 kB)
Collecting joblib>=0.11
  Downloading joblib-1.1.0-py2.py3-none-any.whl (306 kB)
     |████████████████████████████████| 306 kB 2.8 MB/s
Installing collected packages: threadpoolctl, scipy, joblib, scikit-le
arn
Successfully installed joblib-1.1.0 scikit-learn-1.0.2 scipy-1.7.3 thr
eadpoolctl-3.0.0
```

Dataset used: https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho (https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho)

In [4]:

```
df = pd.read_csv (r'/home/vedant/Downloads/Car details v3.csv')
```

In [5]:

```
print(df)
```

```
                             name  year  selling_price  km_driven
fuel  \
0           Maruti Swift Dzire VDI  2014         450000     145500  Di
esel
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000  Di
esel
2         Honda City 2017-2020 EXi  2006         158000     140000  Pe
trol
3        Hyundai i20 Sportz Diesel  2010         225000     127000  Di
esel
4           Maruti Swift VXI BSIII  2007         130000     120000  Pe
trol
...                            ...   ...            ...        ...
...
8123            Hyundai i20 Magna  2013         320000     110000  Pe
trol
8124         Hyundai Verna CRDi SX  2007         135000     119000  Di
esel
8125        Maruti Swift Dzire ZDi  2009         382000     120000  Di
esel
8126               Tata Indigo CR4  2013         290000      25000  Di
esel
8127               Tata Indigo CR4  2013         290000      25000  Di
esel

     seller_type transmission                 owner     mileage   engi
ne  \
0     Individual       Manual           First Owner   23.4 kmpl  1248
CC
1     Individual       Manual          Second Owner  21.14 kmpl  1498
CC
2     Individual       Manual           Third Owner   17.7 kmpl  1497
CC
3     Individual       Manual           First Owner   23.0 kmpl  1396
CC
4     Individual       Manual           First Owner   16.1 kmpl  1298
CC
...          ...          ...                   ...         ...
...
8123  Individual       Manual           First Owner   18.5 kmpl  1197
CC
8124  Individual       Manual  Fourth & Above Owner   16.8 kmpl  1493
CC
8125  Individual       Manual           First Owner   19.3 kmpl  1248
CC
8126  Individual       Manual           First Owner  23.57 kmpl  1396
CC
8127  Individual       Manual           First Owner  23.57 kmpl  1396
CC

       max_power                    torque  seats
0         74 bhp             190Nm@ 2000rpm    5.0
1     103.52 bhp        250Nm@ 1500-2500rpm    5.0
2         78 bhp      12.7@ 2,700(kgm@ rpm)    5.0
3         90 bhp   22.4 kgm at 1750-2750rpm    5.0
4       88.2 bhp      11.5@ 4,500(kgm@ rpm)    5.0
...          ...                         ...    ...
```

```
8123   82.85 bhp            113.7Nm@ 4000rpm    5.0
8124     110 bhp  24@ 1,900-2,750(kgm@ rpm)    5.0
8125    73.9 bhp              190Nm@ 2000rpm    5.0
8126      70 bhp         140Nm@ 1800-3000rpm    5.0
8127      70 bhp         140Nm@ 1800-3000rpm    5.0

[8128 rows x 13 columns]
```

In [25]:

```
df.info()
#The info() method prints information about the DataFrame.
#The information contains the number of columns, column labels, column data types,
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           8128 non-null   object 
 1   year           8128 non-null   int64  
 2   selling_price  8128 non-null   int64  
 3   km_driven      8128 non-null   int64  
 4   fuel           8128 non-null   object 
 5   seller_type    8128 non-null   object 
 6   transmission   8128 non-null   object 
 7   owner          8128 non-null   object 
 8   mileage        7907 non-null   object 
 9   engine         7907 non-null   object 
 10  max_power      7913 non-null   object 
 11  torque         7906 non-null   object 
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

In [26]:

```
df.isnull()

# it prints True if the value is NULL
```

Out[26]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
8123	False	False	False	False	False	False	False	False	False
8124	False	False	False	False	False	False	False	False	False
8125	False	False	False	False	False	False	False	False	False
8126	False	False	False	False	False	False	False	False	False
8127	False	False	False	False	False	False	False	False	False

8128 rows × 13 columns

In [8]:

```
df.iloc[0]
```

Out[8]:

```
name             Maruti Swift Dzire VDI
year                               2014
selling_price                    450000
km_driven                        145500
fuel                             Diesel
seller_type                  Individual
transmission                     Manual
owner                       First Owner
mileage                       23.4 kmpl
engine                          1248 CC
max_power                        74 bhp
torque                  190Nm@ 2000rpm
seats                              5.0
Name: 0, dtype: object
```

In [9]:

```
df.iloc[1]
```

Out[9]:

```
name             Skoda Rapid 1.5 TDI Ambition
year                                     2014
selling_price                          370000
km_driven                              120000
fuel                                   Diesel
seller_type                       Individual
transmission                          Manual
owner                            Second Owner
mileage                            21.14 kmpl
engine                                1498 CC
max_power                           103.52 bhp
torque                    250Nm@ 1500-2500rpm
seats                                     5.0
Name: 1, dtype: object
```

In [11]:

```
df['name'].head(3)
```

Out[11]:

```
0          Maruti Swift Dzire VDI
1    Skoda Rapid 1.5 TDI Ambition
2        Honda City 2017-2020 EXi
Name: name, dtype: object
```

In [16]:

```
cat = pd.Categorical(df.seats)
```

```
Data includes the text columns, which are repetitive. Features like gender,
country, and codes are always repetitive.
These are the examples for categorical data.
Categorical variables can take on only a limited, and usually fixed number of
possible values.
Besides the fixed length, categorical data might have an order but cannot perform
numerical operation.
```

In [17]:

```
print(cat)
```

```
[5.0, 5.0, 5.0, 5.0, 5.0, ..., 5.0, 5.0, 5.0, 5.0, 5.0]
Length: 8128
Categories (9, float64): [2.0, 4.0, 5.0, 6.0, ..., 8.0, 9.0, 10.0, 14.
0]

/home/vedant/jupyternotebook/jupyterenv/lib/python3.8/site-packages/pa
ndas/io/formats/format.py:1429: FutureWarning: Index.ravel returning n
darray is deprecated; in a future version this will return a view on s
elf.
  for val, m in zip(values.ravel(), mask.ravel())
```

quantitative variables are variables measured on a numeric scale. Height, weight, response time, subjective rating
of pain, temperature, and score on an exam are all examples of quantitative variables.

We need to use an Encoder. One of the most used and popular ones are LabelEncoder
Both are provided as parts of sklearn library.

In [20]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
```

In [23]:

```
x = df.seats
y = labelencoder.fit_transform(x)
print(y)
```

```
[2 2 2 ... 2 2 2]
```

In [24]:

```
print(cat[60])
```

```
5.0
```

In [ ]: