In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Logistic Regression

In [2]:

```
from sklearn.linear_model import LogisticRegression
```

In [3]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\c7_used_cars.csv")
df
```

Out[3]:

	Unnamed: 0	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSiz
0	0	T-Roc	2019	25000	Automatic	13904	Diesel	145	49.6	2
1	1	T-Roc	2019	26883	Automatic	4562	Diesel	145	49.6	2
2	2	T-Roc	2019	20000	Manual	7414	Diesel	145	50.4	2
3	3	T-Roc	2019	33492	Automatic	4825	Petrol	145	32.5	2
4	4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	150	39.8	1
...	...	...	...	...	...	...	...	...	...	
99182	10663	A3	2020	16999	Manual	4018	Petrol	145	49.6	1
99183	10664	A3	2020	16999	Manual	1978	Petrol	150	49.6	1
99184	10665	A3	2020	17199	Manual	609	Petrol	150	49.6	1
99185	10666	Q3	2017	19499	Automatic	8646	Petrol	150	47.9	1
99186	10667	Q3	2016	15999	Manual	11855	Petrol	150	47.9	1

99187 rows × 11 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Unnamed: 0', 'model', 'year', 'price', 'transmission', 'mileage',
       'fuelType', 'tax', 'mpg', 'engineSize', 'Make'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99187 entries, 0 to 99186
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    99187 non-null  int64  
 1   model         99187 non-null  object 
 2   year          99187 non-null  int64  
 3   price         99187 non-null  int64  
 4   transmission  99187 non-null  object 
 5   mileage       99187 non-null  int64  
 6   fuelType      99187 non-null  object 
 7   tax           99187 non-null  int64  
 8   mpg           99187 non-null  float64
 9   engineSize    99187 non-null  float64
 10  Make          99187 non-null  object 
dtypes: float64(2), int64(5), object(4)
memory usage: 8.3+ MB
```

In [6]:

```
feature_matrix=df[['Unnamed: 0', 'year', 'price', 'mileage', 'tax']]
target_vector = df[['tax']]
```

In [7]:

```
feature_matrix.shape
```

Out[7]:

```
(99187, 5)
```

In [8]:

```
target_vector.shape
```

Out[8]:

```
(99187, 1)
```

In [9]:

```
from sklearn.preprocessing import StandardScaler
```

In [10]:

```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [11]:

```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  return f(*args, **kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[11]:

```
LogisticRegression()
```

In [14]:

```
observation=[[1.4,2,3,3.5,4]]
# Take Random Values
```

In [15]:

```
prediction = logr.predict(observation)
print(prediction)
```

```
[300]
```

In [16]:

```
logr.classes_
```

Out[16]:

```
array([  0,  10,  20,  30, 110, 115, 120, 125, 130, 135, 140, 145, 150,
       155, 160, 165, 185, 190, 195, 200, 205, 210, 220, 230, 235, 240,
       245, 250, 255, 260, 265, 270, 280, 290, 295, 300, 305, 315, 325,
       330, 515, 520, 535, 540, 555, 565, 570, 580], dtype=int64)
```

In [17]:

```
logr.predict_proba(observation)[0][0]
```

Out[17]:

```
1.283204141367192e-203
```

In [18]:

```
logr.predict_proba(observation)[0][1]
```

Out[18]:

```
2.7848501207834577e-59
```

Logistic Regression-2

In [19]:

```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [20]:

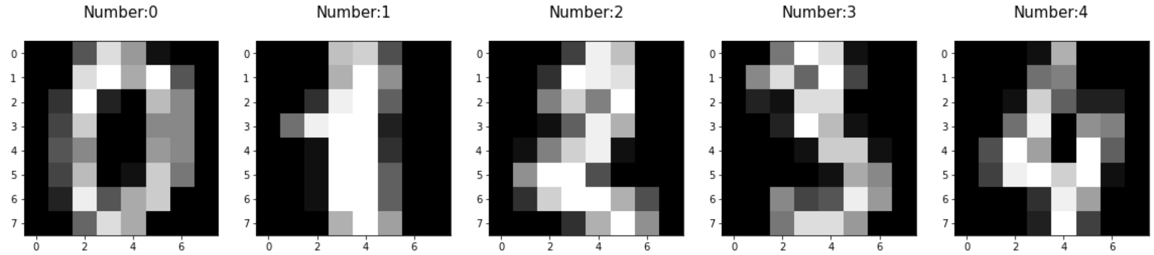
```
digits = load_digits()
digits
```

Out[20]:

```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
```

In [21]:

```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=15)
```



In [22]:

```
x_train,x_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.3
```

In [23]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1257, 64)
(540, 64)
(1257,)
(540,)
```

In [24]:

```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[24]:

```
LogisticRegression(max_iter=10000)
```

In [25]:

```
print(logre.predict(x_test))
```

```
[2 0 4 7 2 1 2 1 2 6 0 7 6 0 6 2 5 8 9 0 5 5 3 8 2 4 1 3 9 7 5 5 6 7 3 6 7
 3 1 2 8 5 5 6 6 4 8 3 3 4 2 0 9 1 4 7 4 8 6 0 7 2 7 2 7 7 3 5 0 7 3 1 3 5
 8 6 8 7 5 0 3 2 0 0 9 6 2 2 3 2 7 5 6 9 6 2 8 7 9 3 5 7 2 9 2 2 1 4 9 0 3
 3 1 4 9 8 1 6 6 8 3 0 0 5 2 1 7 7 1 0 7 1 2 1 0 5 0 4 6 7 0 1 0 9 5 3 1 6
 6 4 7 7 5 5 9 6 8 5 6 6 8 2 6 9 5 3 1 7 0 7 6 4 5 3 1 8 0 4 5 7 9 7 5 8 3
 8 6 5 4 1 5 1 7 6 1 6 2 1 1 3 3 8 9 8 5 6 8 0 3 8 9 9 5 3 0 6 0 4 6 3 3 1
 3 6 5 5 0 8 8 9 9 1 4 4 9 8 0 7 2 5 6 8 2 2 3 6 7 4 0 2 6 2 0 9 0 0 2 4 8
 8 7 5 9 1 8 5 5 7 3 0 3 1 9 8 3 9 1 7 4 7 5 4 1 8 4 9 4 8 9 0 3 8 5 9 2 0
 2 4 4 6 3 0 8 4 6 9 2 2 3 0 5 9 3 1 5 6 8 8 4 3 0 2 1 1 7 9 4 5 1 6 9 7 5
 1 3 7 2 3 5 4 2 9 3 0 6 8 4 4 3 6 1 5 8 9 7 7 4 5 1 1 5 2 3 1 4 3 7 3 1 7
 2 8 9 1 7 4 5 7 2 5 7 2 7 5 5 5 5 9 3 3 8 4 9 7 5 4 2 9 4 9 4 1 8 8 4 5 3
 8 2 8 1 2 6 0 3 7 1 2 0 0 4 6 7 6 4 0 7 4 7 7 0 1 7 4 0 4 3 0 8 0 7 3 8 6
 7 2 1 3 7 3 6 7 5 8 5 3 4 4 7 4 0 6 5 2 2 2 0 1 7 8 4 6 9 6 5 1 0 6 9 4 4
 5 2 9 3 6 7 4 0 9 4 6 8 2 7 4 6 2 0 3 0 1 5 4 7 7 2 3 3 3 9 8 5 7 3 2 9 4
 3 8 3 2 2 1 3 8 8 8 0 7 5 3 2 8 2 1 0 2 3 5]
```

In [26]:

```
print(logre.score(x_test,y_test))
```

```
0.9666666666666667
```