In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv(r"c7_used_cars")
```

In [4]:
```
df
```

Out[4]:

| | Unnamed: 0 | model | year | price | transmission | mileage | fuelType | tax | mpg | engineSize | Make |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | T-Roc | 2019 | 25000 | Automatic | 13904 | Diesel | 145 | 49.6 | 2.0 | VW |
| **1** | 1 | T-Roc | 2019 | 26883 | Automatic | 4562 | Diesel | 145 | 49.6 | 2.0 | VW |
| **2** | 2 | T-Roc | 2019 | 20000 | Manual | 7414 | Diesel | 145 | 50.4 | 2.0 | VW |
| **3** | 3 | T-Roc | 2019 | 33492 | Automatic | 4825 | Petrol | 145 | 32.5 | 2.0 | VW |
| **4** | 4 | T-Roc | 2019 | 22900 | Semi-Auto | 6500 | Petrol | 150 | 39.8 | 1.5 | VW |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99182** | 10663 | A3 | 2020 | 16999 | Manual | 4018 | Petrol | 145 | 49.6 | 1.0 | Audi |
| **99183** | 10664 | A3 | 2020 | 16999 | Manual | 1978 | Petrol | 150 | 49.6 | 1.0 | Audi |
| **99184** | 10665 | A3 | 2020 | 17199 | Manual | 609 | Petrol | 150 | 49.6 | 1.0 | Audi |
| **99185** | 10666 | Q3 | 2017 | 19499 | Automatic | 8646 | Petrol | 150 | 47.9 | 1.4 | Audi |
| **99186** | 10667 | Q3 | 2016 | 15999 | Manual | 11855 | Petrol | 150 | 47.9 | 1.4 | Audi |

99187 rows × 11 columns

In [5]:
```
1 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]:
```
1 df.dropna(inplace=True)
```

In [7]:
```
1 df.columns
```

Out[7]:
```
Index(['Unnamed: 0', 'model', 'year', 'price', 'transmission', 'mileage',
       'fuelType', 'tax', 'mpg', 'engineSize', 'Make'],
      dtype='object')
```

In [8]:
```
1 df
```

Out[8]:

| | Unnamed: 0 | model | year | price | transmission | mileage | fuelType | tax | mpg | engineSize | Make |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | T-Roc | 2019 | 25000 | Automatic | 13904 | Diesel | 145 | 49.6 | 2.0 | VW |
| **1** | 1 | T-Roc | 2019 | 26883 | Automatic | 4562 | Diesel | 145 | 49.6 | 2.0 | VW |
| **2** | 2 | T-Roc | 2019 | 20000 | Manual | 7414 | Diesel | 145 | 50.4 | 2.0 | VW |
| **3** | 3 | T-Roc | 2019 | 33492 | Automatic | 4825 | Petrol | 145 | 32.5 | 2.0 | VW |
| **4** | 4 | T-Roc | 2019 | 22900 | Semi-Auto | 6500 | Petrol | 150 | 39.8 | 1.5 | VW |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99182** | 10663 | A3 | 2020 | 16999 | Manual | 4018 | Petrol | 145 | 49.6 | 1.0 | Audi |
| **99183** | 10664 | A3 | 2020 | 16999 | Manual | 1978 | Petrol | 150 | 49.6 | 1.0 | Audi |
| **99184** | 10665 | A3 | 2020 | 17199 | Manual | 609 | Petrol | 150 | 49.6 | 1.0 | Audi |
| **99185** | 10666 | Q3 | 2017 | 19499 | Automatic | 8646 | Petrol | 150 | 47.9 | 1.4 | Audi |
| **99186** | 10667 | Q3 | 2016 | 15999 | Manual | 11855 | Petrol | 150 | 47.9 | 1.4 | Audi |

99187 rows × 11 columns

In [9]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [10]:
```
1 logr =LogisticRegression()
```

In [11]:
```
1 feature_matrix=df[['year', 'price','mileage','tax', 'mpg', 'engineSize']]
2 target_vector=df['transmission']
```

In [12]:
```
1 feature_matrix.shape
```

Out[12]: (99187, 6)

In [13]:
```
1 target_vector.shape
```

Out[13]: (99187,)

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [16]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[16]:
```
LogisticRegression()
```

In [17]:
```
observation=[[1,2,3,4,5,6]]
```

In [18]:
```
prediction = logr.predict(observation)
print(prediction)
```

```
['Automatic']
```

In [19]:
```
logr.classes_
```

Out[19]:
```
array(['Automatic', 'Manual', 'Other', 'Semi-Auto'], dtype=object)
```

In [20]:
```
logr.predict_proba(observation)[0][1]
```

Out[20]:
```
3.835443675028171e-05
```

In [21]:
```
logr.predict_proba(observation)[0][0]
```

Out[21]:
```
0.739174836321085
```

# Linear regression 2

In [22]:

```
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
```

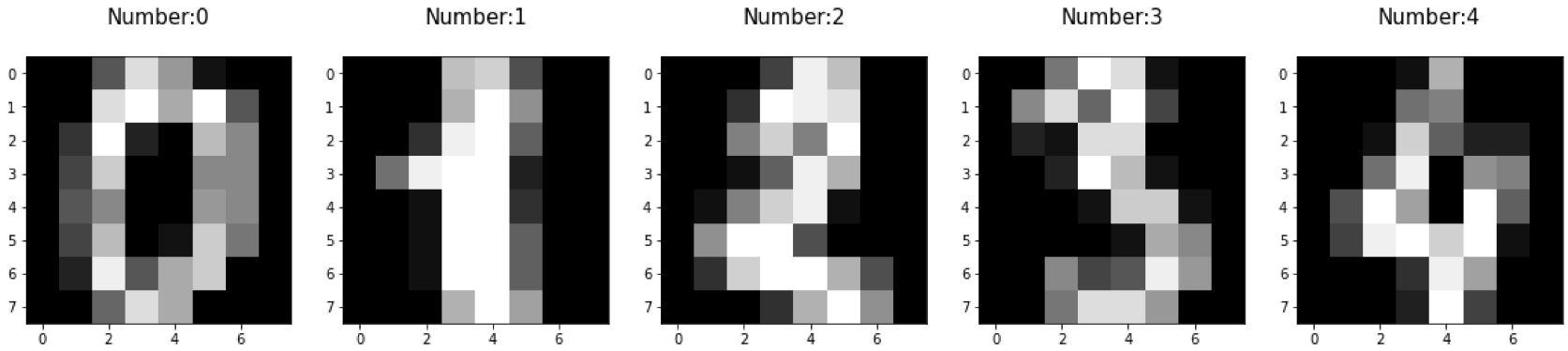
In [23]:

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

Out[23]:

```
{'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',
  'pixel_1_1',
```

In [24]:
```
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 [25]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [26]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [27]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[27]:
```
LogisticRegression(max_iter=10000)
```

In [28]:
```
print(logre.predict(x_test))
```

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

In [29]:
```
print(logre.score(x_test,y_test))
```

```
0.9555555555555556
```