Linear Regression

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

In [2]:
```
df=pd.read_csv(r"C9_Data")
```

In [3]:
```
df
```

Out[3]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
...	...	...	...	...
37513	37513	6	2022-12-31 20:38:56	11
37514	37514	6	2022-12-31 20:39:22	6
37515	37515	6	2022-12-31 20:39:23	6
37516	37516	6	2022-12-31 20:39:31	9
37517	37517	6	2022-12-31 20:39:31	9

37518 rows × 4 columns

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37518 entries, 0 to 37517
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   row_id     37518 non-null  int64 
 1   user_id    37518 non-null  int64 
 2   timestamp  37518 non-null  object
 3   gate_id    37518 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 1.1+ MB
```

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

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

Out[7]:
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

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

Out[8]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
...	...	...	...	...
37513	37513	6	2022-12-31 20:38:56	11
37514	37514	6	2022-12-31 20:39:22	6
37515	37515	6	2022-12-31 20:39:23	6
37516	37516	6	2022-12-31 20:39:31	9
37517	37517	6	2022-12-31 20:39:31	9

37518 rows × 4 columns

In [10]:
```
1 x=df[['row_id', 'user_id','gate_id']]
2 y=df[['gate_id']]
```

In [11]:
```
1 from sklearn.model_selection import train_test_split
```

In [12]:
```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
1 from sklearn.linear_model import LinearRegression
2
3 lr=LinearRegression()
4 lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
1 print(lr.intercept_)
```

```
[-1.0658141e-13]
```

In [15]:
```
1 print(lr.score(x_test,y_test))
```

```
1.0
```

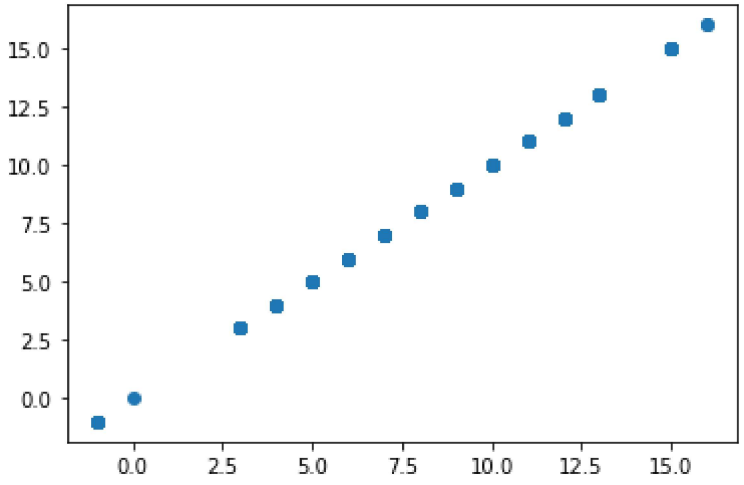
In [16]:
```
1 coeff=pd.DataFrame(lr.coef_)
2 coeff
```

Out[16]:

	0	1	2
0	5.196102e-18	5.450160e-17	1.0

In [17]:
```
1 pred = lr.predict(x_test)
2 plt.scatter(y_test,pred)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x22fd27f0c70>
```



Logistic Regression

In [18]:
```
from sklearn.linear_model import LogisticRegression
```

In [19]:
```
logr =LogisticRegression()
```

In [20]:
```
feature_matrix=df[['row_id', 'user_id']]
target_vector=df['gate_id']
```

In [21]:
```
feature_matrix.shape
```

Out[21]: (37518, 2)

In [22]:
```
target_vector.shape
```

Out[22]: (37518,)

In [23]:
```
from sklearn.preprocessing import StandardScaler
```

In [24]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [25]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
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 in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/prep
rocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/
stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[25]: LogisticRegression()

In [26]:
```
observation=[[1,2]]
```

In [27]:
```
prediction = logr.predict(observation)
print(prediction)
```

```
[3]
```

In [28]:
```
logr.classes_
```

Out[28]:
```
array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16],
      dtype=int64)
```

In [29]:
```
logr.predict_proba(observation)[0][1]
```

Out[29]:
```
2.4322107532317633e-05
```

In [30]:
```
logr.predict_proba(observation)[0][0]
```

Out[30]:
```
0.005365176788164149
```

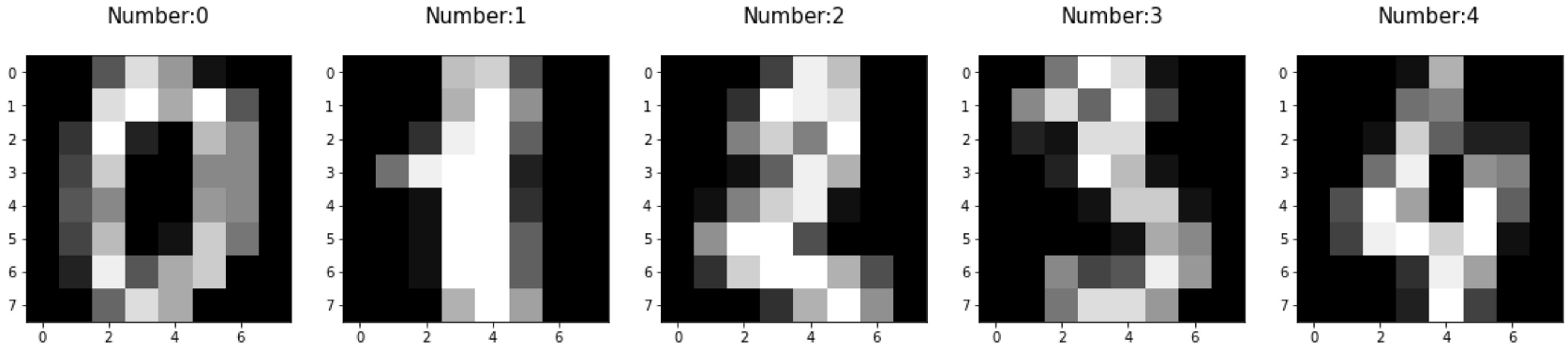
Linear regression 2

In [31]:
```
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 [32]:
```
digits =load_digits()
digits
```

Out[32]:
```
{'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 [33]:
```
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 [34]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [35]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [36]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[36]:
```
LogisticRegression(max_iter=10000)
```

In [37]:
```
print(logre.predict(x_test))
```

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

In [38]:
```
print(logre.score(x_test,y_test))
```

```
0.9629629629629629
```