In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\C9_Data - C9_Data.csv")
data
```

Out[4]:

	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 [5]:
```
data.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 [7]:
```
df=data[['row_id','user_id','gate_id']]
```

In [8]:
```
x=df[['row_id','user_id']]
y=df['gate_id']
```

Linear Regression

In [9]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [10]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
print(lr.intercept_)
```

```
7.294157482332566
```

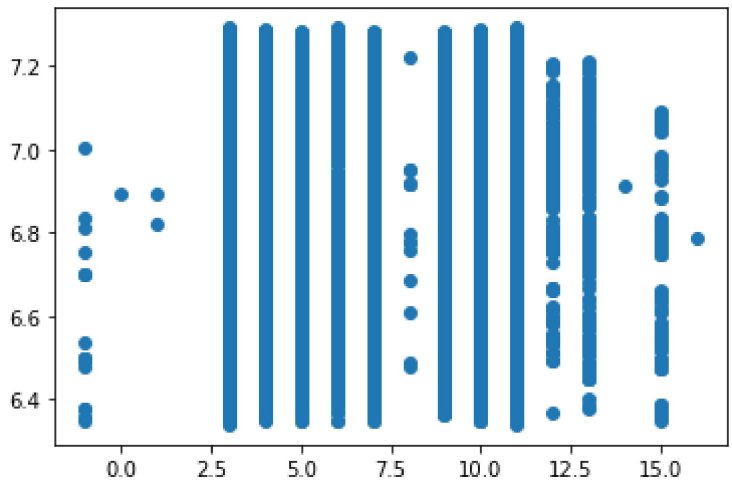
In [12]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[12]:

	Co-efficient
row_id	-0.000006
user_id	-0.012841

In [13]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]:
```
<matplotlib.collections.PathCollection at 0x1ece666deb0>
```



In [14]:
```
print(lr.score(x_test,y_test))
```

```
0.005404512629078484
```

Logistic Regression

In [15]:
```
from sklearn.linear_model import LogisticRegression
```

In [16]:
```
df=data[['row_id','user_id','gate_id']]
```

In [21]:
```
feature_matrix = df.iloc[:,0:4]
target_vector = df['gate_id']
```

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

Out[22]:
```
(37518, 3)
```

In [23]:
```
target_vector.shape
```

Out[23]:
```
(37518,)
```

In [24]:
```
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [26]:
```
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/preprocessing.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[26]:
```
LogisticRegression()
```

In [27]:
```
observation=[[5,7,9]]
```

In [28]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[15]
```

In [29]:
```
logr.classes_
```

Out[29]:
```
array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16],
      dtype=int64)
```

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

Out[30]:
```
0.0
```

In [31]:
```
logr.predict_proba(observation)[0][0]
```

Out[31]:
```
0.0
```

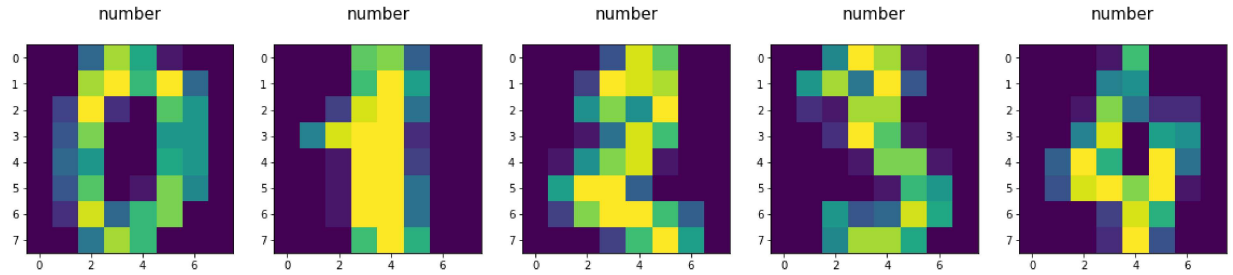
In [32]:
```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [33]:
```
digits = load_digits()
digits
```

```
eprocessing programs made available by NIST were used to extract\nnormalized
bitmaps of handwritten digits from a preprinted form. From a\ntotal of 43 peo
ple, 30 contributed to the training set and different 13\nto the test set. 32
x32 bitmaps are divided into nonoverlapping blocks of\n4x4 and the number of
on pixels are counted in each block. This generates\nan input matrix of 8x8 w
here each element is an integer in the range\n0..16. This reduces dimensional
ity and gives invariance to small\ndistortions.\n\nFor info on NIST preproces
sing routines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick,
J. Geist, P. J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-Based Hand
print Recognition System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  -
C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their\n    App
lications to Handwritten Digit Recognition, MSc Thesis, Institute of\n    Gra
duate Studies in Science and Engineering, Bogazici University.\n  - E. Alpayd
in, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n  - Ken Tang and Po
nnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityr

eduction using relevance weighted LDA. School of\n    Electrical and Electron
ic Engineering Nanyang Technological University.\n    2005.\n  - Claudio Gent
ile. A New Approximate Maximal Margin Classification\n    Algorithm. NIPS. 20
00.\n"}
```

In [34]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:8])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)))
    plt.title("number\n"%label,fontsize=15)
```



In [35]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_siz
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()
logre.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: 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/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[36]:
```
LogisticRegression()
```

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

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

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

```
0.9666666666666667
```