In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
from sklearn.linear_model import LogisticRegression
```

In [36]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\C6_bmi.csv")
df
```

Out[36]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

In [37]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Gender  500 non-null    object
 1   Height  500 non-null    int64 
 2   Weight  500 non-null    int64 
 3   Index   500 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 15.8+ KB
```

In [38]:

```
df.columns
```

Out[38]:

```
Index(['Gender', 'Height', 'Weight', 'Index'], dtype='object')
```

In [44]:

```
feature_matrix = df1[[ 'Height', 'Weight', 'Index']]
target_vector = df1[['Weight']]
```

In [45]:

```
feature_matrix.shape
```

Out[45]:

```
(50, 3)
```

In [46]:

```
target_vector.shape
```

Out[46]:

```
(50, 1)
```

In [47]:

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

In [48]:

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

In [49]:

```
log = LogisticRegression()
log.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)
```

Out[49]:

```
LogisticRegression()
```

In [50]:

```
observations = [[1,2,3]]
```

In [51]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[139]
```

In [52]:

```
log.classes_
```

Out[52]:

```
array([ 51,  52,  56,  61,  62,  64,  65,  67,  72,  76,  79,  80,  81,
        87,  90,  92,  95,  96,  97, 101, 103, 104, 107, 108, 110, 111,
       114, 118, 120, 121, 122, 126, 129, 131, 132, 139, 145, 149, 152,
       153, 159], dtype=int64)
```

In [53]:

```
log.predict_proba(observations)[0][0]
```

Out[53]:

```
3.431239878111352e-05
```

In [54]:

```
log.predict_proba(observations)[0][1]
```

Out[54]:

```
2.0161123095969065e-05
```

Logic regression 2

In [55]:

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

In [56]:

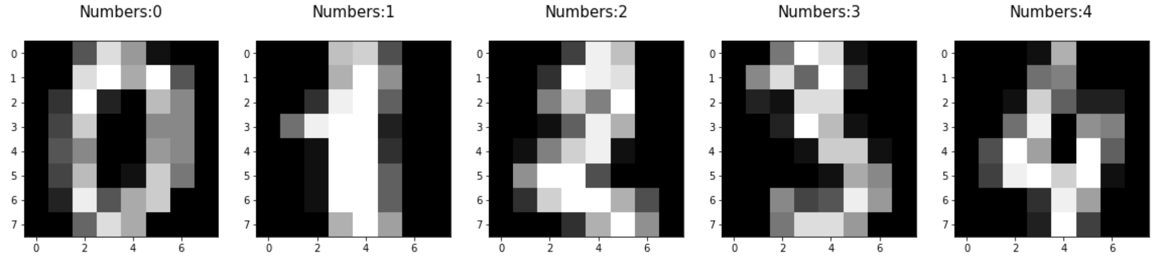
```
digits = load_digits()
digits
```

Out[56]:

```
{'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 [57]:

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



In [58]:

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

In [59]:

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

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

In [60]:

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

Out[60]:

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

In [61]:

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

```
0.9648148148148148
```

In [ ]:

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
from sklearn.linear_model import LogisticRegression
```

In [11]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\c7_used_cars.csv")
df1 = df.head(50)
df1
```

Out[11]:

	Unnamed: 0	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	0	T-Roc	2019	25000	Automatic	13904	Diesel	145	49.6	2.0
1	1	T-Roc	2019	26883	Automatic	4562	Diesel	145	49.6	2.0
2	2	T-Roc	2019	20000	Manual	7414	Diesel	145	50.4	2.0
3	3	T-Roc	2019	33492	Automatic	4825	Petrol	145	32.5	2.0
4	4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	150	39.8	1.5
5	5	T-Roc	2020	31895	Manual	10	Petrol	145	42.2	1.5
6	6	T-Roc	2020	27895	Manual	10	Petrol	145	42.2	1.5
7	7	T-Roc	2020	39495	Semi-Auto	10	Petrol	145	32.5	2.0
8	8	T-Roc	2019	21995	Manual	10	Petrol	145	44.1	1.0
9	9	T-Roc	2019	23285	Manual	10	Petrol	145	42.2	1.5
10	10	T-Roc	2019	23985	Semi-Auto	10	Petrol	145	39.8	1.5
11	11	T-Roc	2019	23585	Manual	10	Petrol	145	42.2	1.5
12	12	T-Roc	2020	25785	Semi-Auto	10	Petrol	145	39.8	1.5
13	13	T-Roc	2019	23995	Semi-Auto	1069	Petrol	145	39.8	1.5
14	14	T-Roc	2018	17495	Manual	21645	Petrol	145	53.3	1.5
15	15	T-Roc	2018	21495	Manual	16972	Petrol	145	53.3	1.5
16	16	T-Roc	2018	18995	Manual	15100	Petrol	145	44.1	1.0
17	17	T-Roc	2018	18995	Manual	1380	Petrol	150	55.4	1.0
18	18	T-Roc	2019	33785	Semi-Auto	2500	Petrol	145	32.5	2.0
19	19	T-Roc	2020	23790	Semi-Auto	5000	Petrol	145	39.8	1.5
20	20	T-Roc	2019	19995	Manual	12300	Petrol	145	42.2	1.5
21	21	T-Roc	2020	21690	Manual	2500	Petrol	145	44.1	1.0
22	22	T-Roc	2019	23290	Manual	5540	Petrol	145	42.2	1.5
23	23	T-Roc	2019	24790	Manual	3500	Petrol	145	42.2	1.5
24	24	T-Roc	2018	19995	Manual	16251	Petrol	145	53.3	1.5
25	25	T-Roc	2019	20290	Manual	5127	Petrol	145	42.2	1.5
26	26	T-Roc	2019	33990	Semi-Auto	3500	Petrol	145	32.5	2.0
27	27	T-Roc	2018	17685	Manual	22584	Petrol	145	44.1	1.0
28	28	T-Roc	2020	24495	Manual	1955	Diesel	150	50.4	2.0
29	29	T-Roc	2018	17995	Manual	10456	Petrol	150	55.4	1.0
30	30	T-Roc	2020	28290	Semi-Auto	2500	Petrol	145	39.8	1.5
31	31	T-Roc	2018	17495	Manual	6908	Petrol	145	55.4	1.0
32	32	T-Roc	2019	22990	Manual	7216	Diesel	145	50.4	1.6
33	33	T-Roc	2019	22995	Manual	4003	Diesel	145	46.3	2.0
34	34	T-Roc	2019	21498	Manual	5	Diesel	150	50.4	1.6
35	35	T-Roc	2018	17995	Manual	14837	Petrol	145	53.3	1.5

	Unnamed: 0	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
36	36	T-Roc	2018	17995	Manual	14337	Petrol	145	53.3	1.5
37	37	T-Roc	2019	22230	Manual	3392	Diesel	145	50.4	2.0
38	38	T-Roc	2018	18695	Manual	6753	Petrol	145	53.3	1.5
39	39	T-Roc	2019	25990	Semi-Auto	2423	Diesel	145	49.6	2.0
40	40	T-Roc	2018	18695	Manual	6753	Petrol	145	53.3	1.5
41	41	T-Roc	2019	23999	Manual	4224	Petrol	145	42.2	1.5
42	42	T-Roc	2019	24490	Manual	6144	Diesel	145	50.4	2.0
43	43	T-Roc	2019	24490	Manual	5345	Diesel	145	50.4	2.0
44	44	T-Roc	2019	25990	Semi-Auto	4233	Diesel	145	49.6	2.0
45	45	T-Roc	2018	17995	Manual	14837	Petrol	145	53.3	1.5
46	46	T-Roc	2018	17995	Manual	14337	Petrol	145	53.3	1.5
47	47	T-Roc	2019	23999	Semi-Auto	5027	Petrol	145	39.8	1.5
48	48	T-Roc	2019	22230	Manual	3392	Diesel	145	50.4	2.0
49	49	T-Roc	2019	23999	Manual	2520	Petrol	145	42.2	1.5

In [12]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    50 non-null     int64  
 1   model         50 non-null     object 
 2   year          50 non-null     int64  
 3   price         50 non-null     int64  
 4   transmission  50 non-null     object 
 5   mileage       50 non-null     int64  
 6   fuelType      50 non-null     object 
 7   tax           50 non-null     int64  
 8   mpg           50 non-null     float64
 9   engineSize    50 non-null     float64
 10  Make          50 non-null     object 
dtypes: float64(2), int64(5), object(4)
memory usage: 4.4+ KB
```

In [13]:

```
df1.columns
```

Out[13]:

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

In [14]:

```
feature_matrix = df1[['Unnamed: 0', 'year', 'price', 'mileage', 'tax', 'mpg', 'engineSize
target_vector = df1[['price']]
```

In [15]:

```
feature_matrix.shape
```

Out[15]:

```
(50, 7)
```

In [16]:

```
target_vector.shape
```

Out[16]:

```
(50, 1)
```

In [17]:

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

In [18]:

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

In [19]:

```
log = LogisticRegression()
log.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)
```

Out[19]:

```
LogisticRegression()
```

In [20]:

```
observations = [[1,2,3,4,5,6,7]]
```

In [21]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[24490]
```

In [22]:

```
log.classes_
```

Out[22]:

```
array([17495, 17685, 17995, 18695, 18995, 19995, 20000, 20290, 21495,
       21498, 21690, 21995, 22230, 22900, 22990, 22995, 23285, 23290,
       23585, 23790, 23985, 23995, 23999, 24490, 24495, 24790, 25000,
       25785, 25990, 26883, 27895, 28290, 31895, 33492, 33785, 33990,
       39495], dtype=int64)
```

In [23]:

```
log.predict_proba(observations)[0][0]
```

Out[23]:

```
5.087365887249748e-07
```

In [24]:

```
log.predict_proba(observations)[0][1]
```

Out[24]:

```
2.6163073278563958e-08
```

Logic regression 2

In [25]:

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

In [26]:

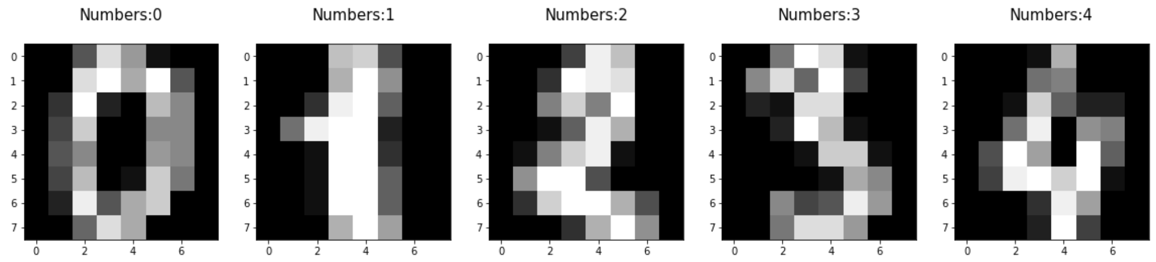
```
digits = load_digits()
digits
```

Out[26]:

```
{'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 [27]:

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



In [28]:

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

In [29]:

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

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

In [30]:

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

Out[30]:

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

In [31]:

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

```
0.9648148148148148
```

In [ ]:

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\C8_loan-train.csv")
df
xf = pd.read_csv(r"C:\Users\user\Desktop\C8_loan-test.csv")
xf
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001015	Male	Yes	0	Graduate	No	5720	
1	LP001022	Male	Yes	1	Graduate	No	3076	
2	LP001031	Male	Yes	2	Graduate	No	5000	
3	LP001035	Male	Yes	2	Graduate	No	2340	
4	LP001051	Male	No	0	Not Graduate	No	3276	
...	...	...	...	...	...	...	...	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	
363	LP002975	Male	Yes	0	Graduate	No	4158	
364	LP002980	Male	No	0	Graduate	No	3250	
365	LP002986	Male	Yes	0	Graduate	No	5000	
366	LP002989	Male	No	0	Graduate	Yes	9200	

367 rows × 12 columns

In [4]:

```
df.fillna("39.0")
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [5]:

```
df0 = df.head(20)
df1 = df0.fillna("30.0")
df1
xf0 = xf.head(20)
xf1 = xf0.fillna("25.5")
xf1
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
0	LP001015	Male	Yes	0	Graduate	No	5720	
1	LP001022	Male	Yes	1	Graduate	No	3076	
2	LP001031	Male	Yes	2	Graduate	No	5000	
3	LP001035	Male	Yes	2	Graduate	No	2340	
4	LP001051	Male	No	0	Not Graduate	No	3276	
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	
6	LP001055	Female	No	1	Not Graduate	No	2226	
7	LP001056	Male	Yes	2	Not Graduate	No	3881	
8	LP001059	Male	Yes	2	Graduate	25.5	13633	
9	LP001067	Male	No	0	Not Graduate	No	2400	
10	LP001078	Male	No	0	Not Graduate	No	3091	
11	LP001082	Male	Yes	1	Graduate	25.5	2185	
12	LP001083	Male	No	3+	Graduate	No	4166	
13	LP001094	Male	Yes	2	Graduate	25.5	12173	
14	LP001096	Female	No	0	Graduate	No	4666	
15	LP001099	Male	No	1	Graduate	No	5667	
16	LP001105	Male	Yes	2	Graduate	No	4583	
17	LP001107	Male	Yes	3+	Graduate	No	3786	
18	LP001108	Male	Yes	0	Graduate	No	9226	
19	LP001115	Male	No	0	Graduate	No	1300	

In [6]:

```
df1.info()
xf1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            20 non-null     object 
 1   Gender             20 non-null     object 
 2   Married            20 non-null     object 
 3   Dependents         20 non-null     object 
 4   Education          20 non-null     object 
 5   Self_Employed      20 non-null     object 
 6   ApplicantIncome    20 non-null     int64  
 7   CoapplicantIncome  20 non-null     float64
 8   LoanAmount         20 non-null     object 
 9   Loan_Amount_Term   20 non-null     object 
 10  Credit_History     20 non-null     object 
 11  Property_Area      20 non-null     object 
 12  Loan_Status        20 non-null     object 
dtypes: float64(1), int64(1), object(11)
memory usage: 2.2+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            20 non-null     object 
 1   Gender             20 non-null     object 
 2   Married            20 non-null     object 
 3   Dependents         20 non-null     object 
 4   Education          20 non-null     object 
 5   Self_Employed      20 non-null     object 
 6   ApplicantIncome    20 non-null     int64  
 7   CoapplicantIncome  20 non-null     int64  
 8   LoanAmount         20 non-null     float64
 9   Loan_Amount_Term   20 non-null     float64
 10  Credit_History     20 non-null     object 
 11  Property_Area      20 non-null     object 
dtypes: float64(2), int64(2), object(8)
memory usage: 2.0+ KB
```

In [7]:

```
df1.describe()
xf1.describe()
```

Out[7]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
count	20.000000	20.000000	20.000000	20.000000
mean	4728.000000	1390.950000	141.100000	336.000000
std	3312.898397	2004.777438	66.004705	59.683375
min	1300.000000	0.000000	40.000000	180.000000
25%	2385.000000	0.000000	100.000000	360.000000
50%	3833.500000	166.500000	126.000000	360.000000
75%	5166.750000	2436.500000	163.000000	360.000000
max	13633.000000	7916.000000	300.000000	360.000000

In [8]:

```
df1.columns
xf1.columns
```

Out[8]:

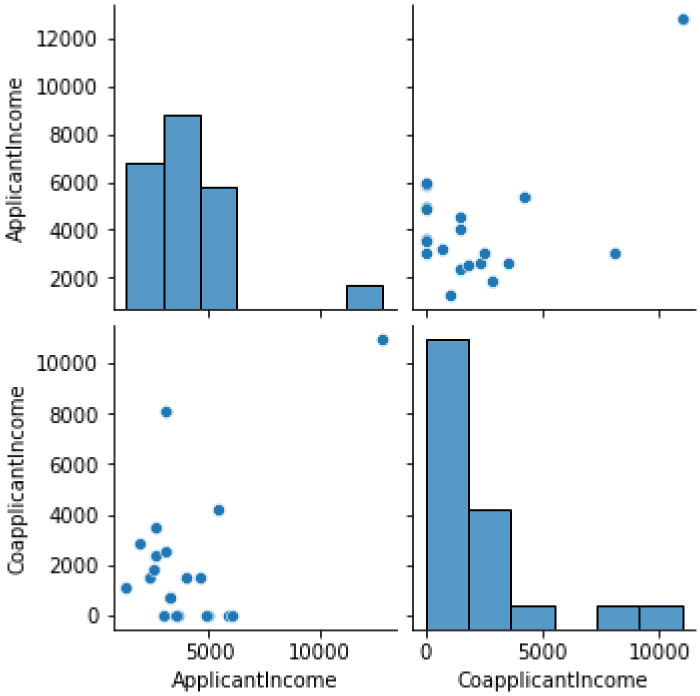
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [9]:

```
sb.pairplot(df1)
sb.pairplot(xf1)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x1d56b02f9a0>
```



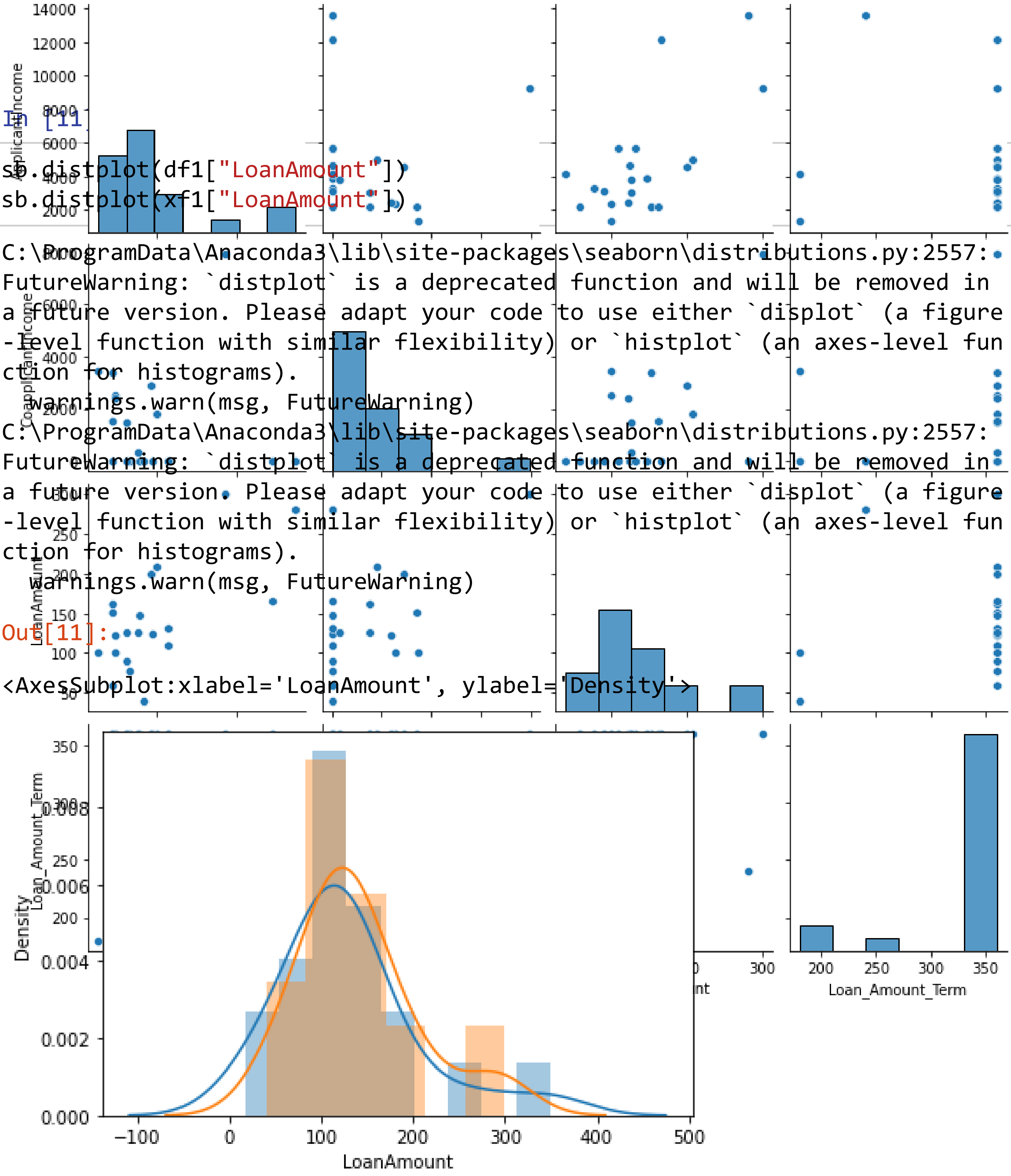
In [11]:

```
sb.distplot(df1["LoanAmount"])
sb.distplot(xf1["LoanAmount"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]:

```
<AxesSubplot:xlabel='LoanAmount', ylabel='Density'>
```

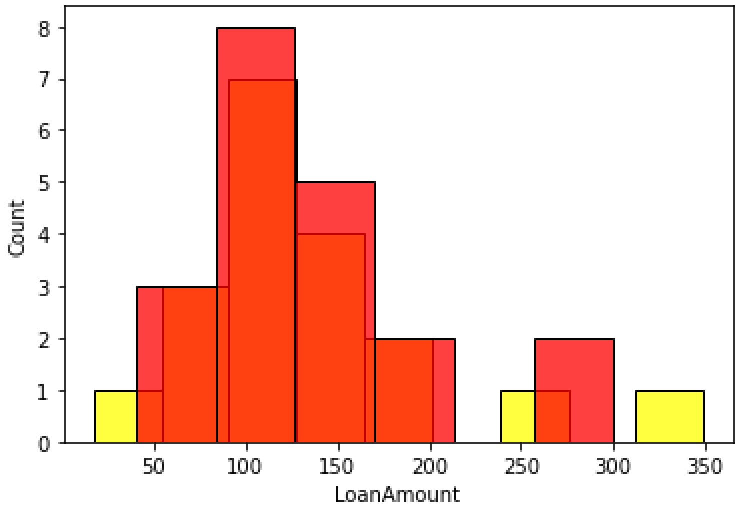


In [12]:

```
sb.histplot(df0["LoanAmount"], color = 'yellow')
sb.histplot(xf0["LoanAmount"],color = 'red')
```

Out[12]:

```
<AxesSubplot:xlabel='LoanAmount', ylabel='Count'>
```



In [13]:

```
df2 = df1[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term']]
df2
xf2 = xf1[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term']]
xf2
```

Out[13]:

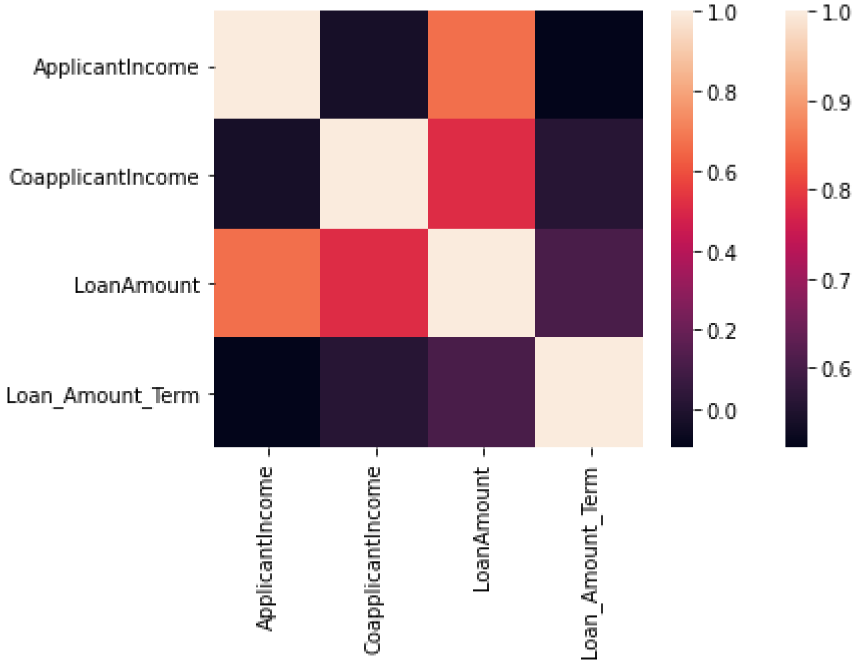
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	5720	0	110.0	360.0
1	3076	1500	126.0	360.0
2	5000	1800	208.0	360.0
3	2340	2546	100.0	360.0
4	3276	0	78.0	360.0
5	2165	3422	152.0	360.0
6	2226	0	59.0	360.0
7	3881	0	147.0	360.0
8	13633	0	280.0	240.0
9	2400	2400	123.0	360.0
10	3091	0	90.0	360.0
11	2185	1516	162.0	360.0
12	4166	0	40.0	180.0
13	12173	0	166.0	360.0
14	4666	0	124.0	360.0
15	5667	0	131.0	360.0
16	4583	2916	200.0	360.0
17	3786	333	126.0	360.0
18	9226	7916	300.0	360.0
19	1300	3470	100.0	180.0

In [14]:

```
sb.heatmap(df2.corr())
sb.heatmap(xf2.corr())
```

Out[14]:

```
<AxesSubplot:>
```



In [16]:

```
x = df2[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term']]
y = xf2[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term']]
```

In [17]:

```
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)
```

REGRESSION

LINEAR REGRESSION

In [18]:

```
from sklearn.linear_model import LinearRegression
```

In [19]:

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

Out[19]:

```
LinearRegression()
```

In [20]:

```
print(lr.intercept_)
```

```
[ 518.19673612 3089.18320968   82.12168522  188.32436315]
```

In [21]:

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

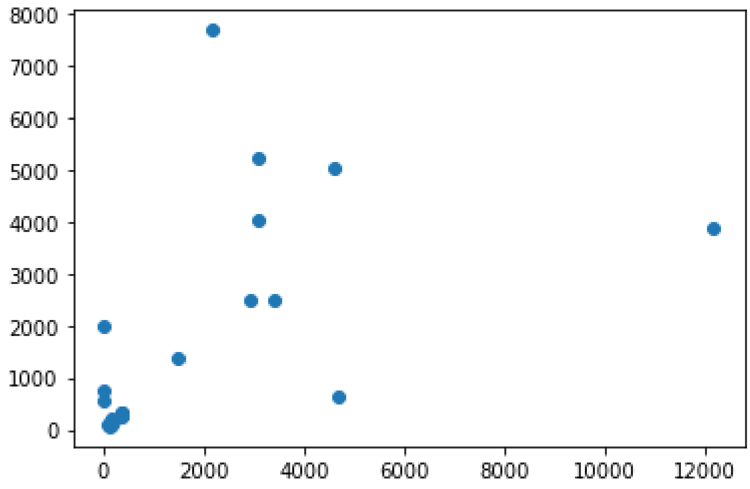
```
-0.30041062799103213
```

In [22]:

```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x1d56dde6f40>
```



lasso and ridge regression

In [23]:

```
lr.score(x_test,y_test)
```

Out[23]:

```
-0.30041062799103213
```

In [24]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[25]:

```
0.3498203226447659
```

In [26]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[26]:

```
0.3498060155009918
```

elasticnet

In [27]:

```
from sklearn.linear_model import ElasticNet
e = ElasticNet()
e.fit(x_train,y_train)
```

Out[27]:

```
ElasticNet()
```

In [28]:

```
print(e.coef_)
```

```
[[-1.36057183e-01 -9.19455661e-01  3.63535214e+01  5.74318976e+00]
 [ 7.77999483e-02 -3.09457272e-01  1.33619367e+01 -9.17478469e+00]
 [-2.23600361e-03 -2.63435366e-02  1.01344545e+00 -2.29546041e-02]
 [ 1.33577283e-02 -8.92564080e-03 -2.00762556e-01  3.76470671e-01]]
```

In [29]:

```
print(e.intercept_)
```

```
[ 519.03586808 3089.22557998   82.13670048  188.332421   ]
```

In [30]:

```
predictions = e.predict(x_test)
predictions
```

Out[30]:

```
array([[4052.32873305,  753.97840373,  119.21853697,  346.30526558],
       [ 650.9613752 , 1980.39587618,   65.09707108,  237.7543812 ],
       [5043.49192301, 2503.23953707,  169.93147105,  306.64351731],
       [7697.91669057, 2476.89979274,  221.81306563,  305.16508533],
       [5229.74571816, 1386.5265841 ,  153.62040221,  345.92285775],
       [3867.51758766,  574.86852515,  110.44686503,  300.37798181]])
```

In [31]:

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

```
-0.29975698893124114
```

Mean Absolute Error

In [32]:

```
from sklearn import metrics
```

In [33]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 1116.607854239524
```

Mean Squared Error

In [34]:

```
print("Mean Squared Error:", metrics.mean_squared_error(y_test,predictions))
```

```
Mean Squared Error: 5309401.758014495
```

Root Mean Squared Error

In [35]:

```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
Root Mean Squared Error 2304.2139132499165
```

In [36]:

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

In [46]:

```
feature_matrix = df2[['ApplicantIncome', 'CoapplicantIncome']]
target_vector = df2[['CoapplicantIncome']]
```

In [47]:

```
feature_matrix.shape
```

Out[47]:

```
(20, 2)
```

In [48]:

```
target_vector.shape
```

Out[48]:

```
(20, 1)
```

In [49]:

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

In [50]:

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

In [51]:

```
log = LogisticRegression()
log.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)
```

Out[51]:

```
LogisticRegression()
```

In [53]:

```
observations = df2[['ApplicantIncome', 'CoapplicantIncome']]
```

In [54]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[10968. 10968. 10968. 10968. 10968. 10968. 10968. 10968. 10968. 10968.
 10968. 10968. 10968. 10968. 10968. 10968. 10968. 10968. 10968. 10968.]
```

In [55]:

```
log.classes_
```

Out[55]:

```
array([    0.,   700.,  1086.,  1508.,  1516.,  1526.,  1840.,  2358.,
        2504.,  2840.,  3500.,  4196.,  8106., 10968.])
```

In [56]:

```
log.predict_proba(observations)[0][0]
```

Out[56]:

```
8.317905704449431e-19
```

In [57]:

```
log.predict_proba(observations)[0][1]
```

Out[57]:

```
0.0
```

logic regression 2

In [59]:

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

In [60]:

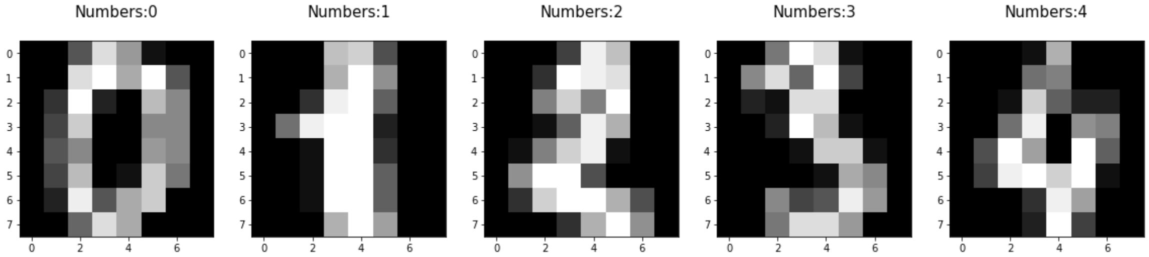
```
digits = load_digits()
digits
```

Out[60]:

```
{'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 [61]:

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



In [62]:

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

In [63]:

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

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

In [64]:

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

Out[64]:

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

In [65]:

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

```
0.9611111111111111
```

In [ ]:

```
```

In [11]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
from sklearn.linear_model import LogisticRegression
import seaborn as sb
```

In [6]:

```
df1 = pd.read_csv(r"C:\Users\user\Desktop\C9_Data.csv")
df = df1.head(50)
df
```

Out[6]:

	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
5	5	18	2022-07-29 09:10:34	10
6	6	18	2022-07-29 09:32:47	11
7	7	18	2022-07-29 09:33:12	4
8	8	18	2022-07-29 09:33:13	4
9	9	1	2022-07-29 09:33:16	7
10	10	18	2022-07-29 09:33:23	9
11	11	18	2022-07-29 09:33:23	9
12	12	18	2022-07-29 09:33:41	5
13	13	18	2022-07-29 09:33:42	5
14	14	18	2022-07-29 09:34:04	10
15	15	1	2022-07-29 09:34:18	9
16	16	1	2022-07-29 09:34:18	9
17	17	1	2022-07-29 09:34:32	5
18	18	1	2022-07-29 09:34:33	5
19	19	1	2022-07-29 09:35:00	10
20	20	3	2022-07-29 09:40:40	7
21	21	3	2022-07-29 09:42:49	9
22	22	3	2022-07-29 09:42:49	9
23	23	3	2022-07-29 09:43:01	5
24	24	3	2022-07-29 09:43:03	5
25	25	3	2022-07-29 09:43:29	10
26	26	6	2022-07-29 09:53:22	7
27	27	29	2022-07-29 09:53:44	7
28	28	29	2022-07-29 09:53:46	7
29	29	6	2022-07-29 09:54:25	9
30	30	6	2022-07-29 09:54:25	9
31	31	6	2022-07-29 09:54:35	5
32	32	6	2022-07-29 09:54:37	5
33	33	6	2022-07-29 09:54:57	10
34	34	29	2022-07-29 09:56:04	9
35	35	29	2022-07-29 09:56:04	9
36	36	29	2022-07-29 09:56:12	5

	row_id	user_id	timestamp	gate_id
37	37	29	2022-07-29 09:56:14	5
38	38	18	2022-07-29 09:56:31	12
39	39	18	2022-07-29 09:56:33	12
40	40	29	2022-07-29 09:56:41	10
41	41	55	2022-07-29 10:09:23	7
42	42	55	2022-07-29 10:10:28	3
43	43	55	2022-07-29 10:10:30	3
44	44	55	2022-07-29 10:10:50	10
45	45	24	2022-07-29 10:12:52	15
46	46	24	2022-07-29 10:15:52	3
47	47	24	2022-07-29 10:15:55	3
48	48	24	2022-07-29 10:16:19	10
49	49	24	2022-07-29 10:17:48	11

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   row_id     50 non-null     int64 
 1   user_id    50 non-null     int64 
 2   timestamp  50 non-null     object
 3   gate_id    50 non-null     int64 
dtypes: int64(3), object(1)
memory usage: 1.7+ KB
```

In [8]:

```
df.describe()
```

Out[8]:

	row_id	user_id	gate_id
count	50.00000	50.000000	50.000000
mean	24.50000	17.820000	7.560000
std	14.57738	14.603173	2.785971
min	0.00000	1.000000	3.000000
25%	12.25000	6.000000	5.000000
50%	24.50000	18.000000	8.000000
75%	36.75000	24.000000	9.750000
max	49.00000	55.000000	15.000000

In [9]:

```
df.columns
```

Out[9]:

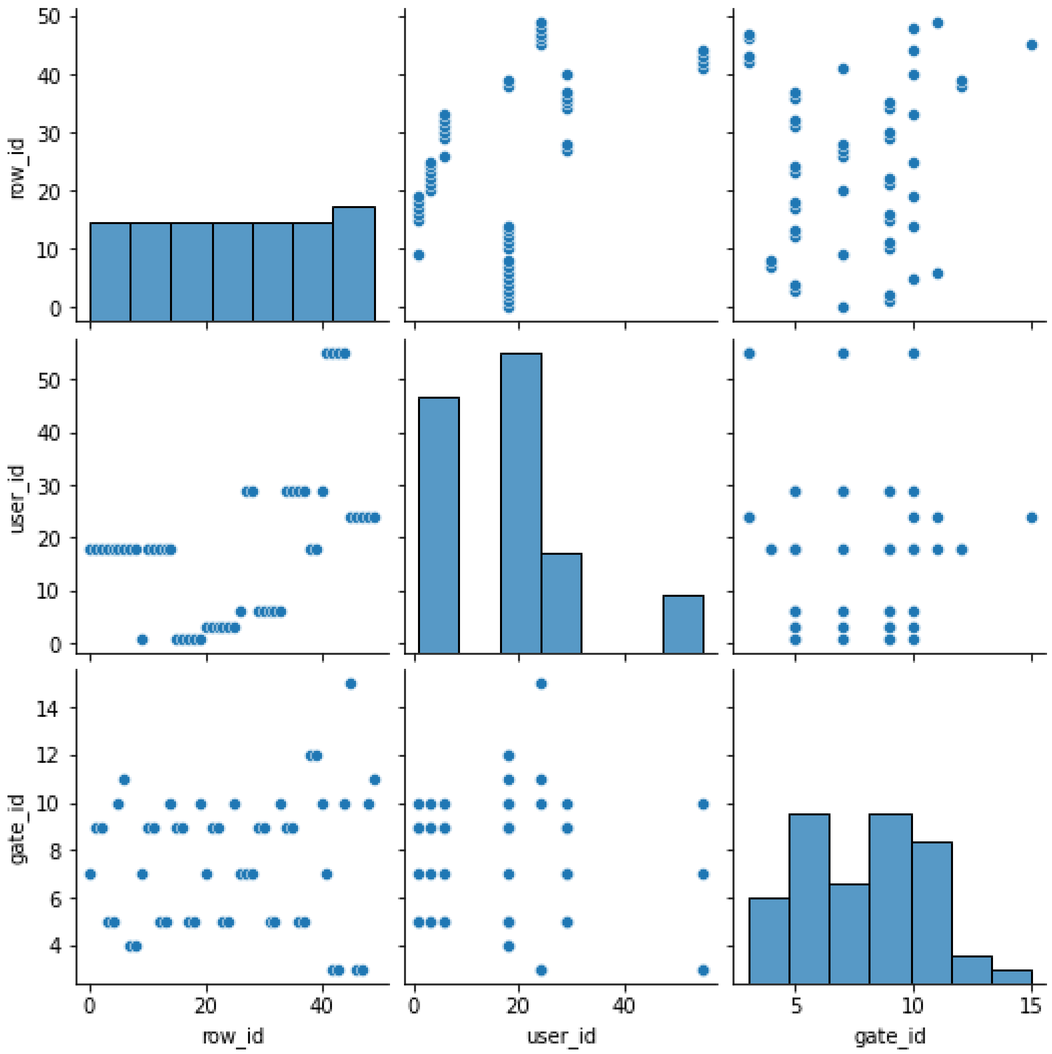
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

In [12]:

```
sb.pairplot(df)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x20da1389cd0>
```



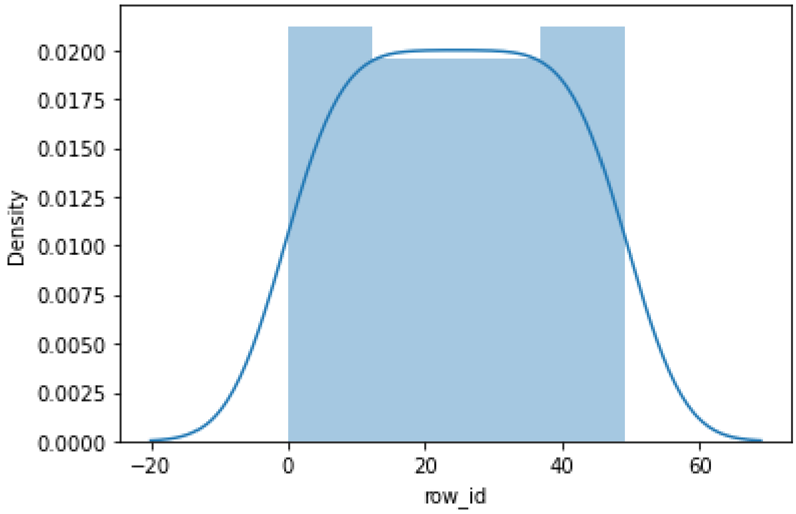
In [13]:

```
sb.distplot(df["row_id"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:

```
<AxesSubplot:xlabel='row_id', ylabel='Density'>
```

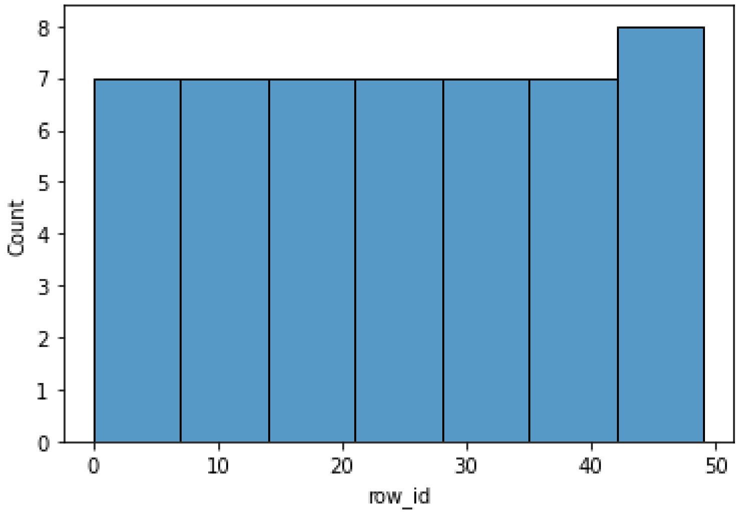


In [14]:

```
sb.histplot(df["row_id"])
```

Out[14]:

```
<AxesSubplot:xlabel='row_id', ylabel='Count'>
```



In [15]:

```
df1=df[['row_id', 'user_id', 'gate_id']]
df1
```

Out[15]:

	row_id	user_id	gate_id
0	0	18	7
1	1	18	9
2	2	18	9
3	3	18	5
4	4	18	5
5	5	18	10
6	6	18	11
7	7	18	4
8	8	18	4
9	9	1	7
10	10	18	9
11	11	18	9
12	12	18	5
13	13	18	5
14	14	18	10
15	15	1	9
16	16	1	9
17	17	1	5
18	18	1	5
19	19	1	10
20	20	3	7
21	21	3	9
22	22	3	9
23	23	3	5
24	24	3	5
25	25	3	10
26	26	6	7
27	27	29	7
28	28	29	7
29	29	6	9
30	30	6	9
31	31	6	5
32	32	6	5
33	33	6	10
34	34	29	9
35	35	29	9
36	36	29	5

	row_id	user_id	gate_id
37	37	29	5
38	38	18	12
39	39	18	12
40	40	29	10
41	41	55	7
42	42	55	3
43	43	55	3
44	44	55	10
45	45	24	15
46	46	24	3
47	47	24	3
48	48	24	10
49	49	24	11

In [18]:

```
x = df1[['row_id', 'user_id', 'gate_id']]
y = df1['row_id']
```

In [19]:

```
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 [20]:

```
from sklearn.linear_model import LinearRegression

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

Out[20]:

```
LinearRegression()
```

In [21]:

```
print(lr.intercept_)
```

```
3.552713678800501e-15
```

In [22]:

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

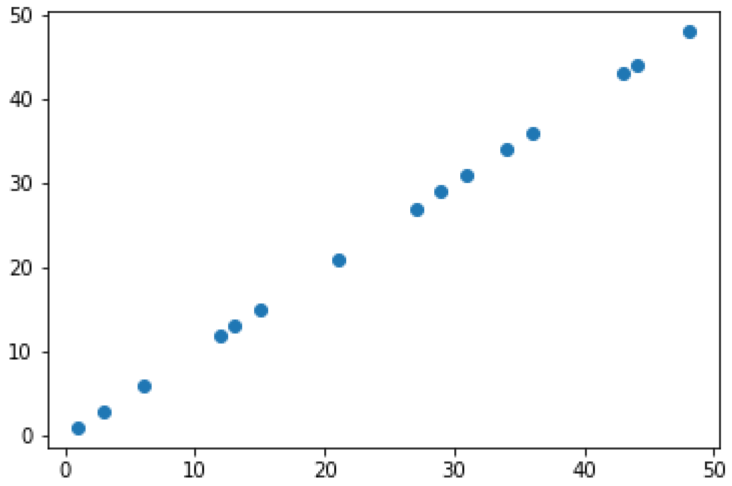
```
1.0
```

In [23]:

```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[23]:

```
<matplotlib.collections.PathCollection at 0x20da259ddf0>
```



In [24]:

```
lr.score(x_test,y_test)
```

Out[24]:

```
1.0
```

In [25]:

```
lr.score(x_train,y_train)
```

Out[25]:

```
1.0
```

In [26]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [27]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[27]:

```
0.9999975268576631
```

In [28]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[28]:

```
0.9975686744103146
```

In [29]:

```
from sklearn.linear_model import ElasticNet
e = ElasticNet()
e.fit(x_train,y_train)
```

Out[29]:

```
ElasticNet()
```

In [30]:

```
print(e.intercept_)
print(e.coef_)
```

```
0.12114106288533932
[0.99508128 0.         0.        ]
```

In [31]:

```
predictions = e.predict(x_test)
predictions
```

Out[31]:

```
array([21.01784793, 47.88504247, 15.04736025, 12.06211642,  3.1063849 ,
       33.95390456, 43.90471735,  1.11622234,  6.09162874, 30.96866072,
       28.97849816, 42.90963607, 13.05719769, 26.98833561, 35.94406712])
```

In [32]:

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

```
0.9999757860638582
```

In [33]:

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

In [36]:

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

In [37]:

```
feature_matrix.shape
```

Out[37]:

```
(50, 3)
```

In [38]:

```
target_vector.shape
```

Out[38]:

```
(50, 1)
```

In [39]:

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

In [40]:

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

In [41]:

```
log = LogisticRegression()
log.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)
```

Out[41]:

```
LogisticRegression()
```

In [51]:

```
observations = [[1,2,3]]
```

In [52]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[18]
```

In [53]:

```
log.classes_
```

Out[53]:

```
array([ 1,  3,  6, 18, 24, 29, 55], dtype=int64)
```

In [54]:

```
log.predict_proba(observations)[0][0]
```

Out[54]:

```
0.00015476222083038137
```

logic regression2

In [20]:

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

In [21]:

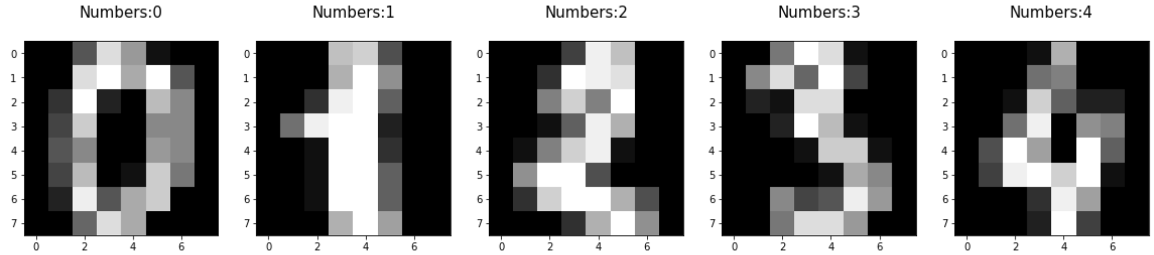
```
digits = load_digits()
digits
```

Out[21]:

```
{'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 [22]:

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



In [23]:

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

In [24]:

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

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

In [25]:

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

Out[25]:

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

In [26]:

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

```
0.9629629629629629
```

In [ ]: