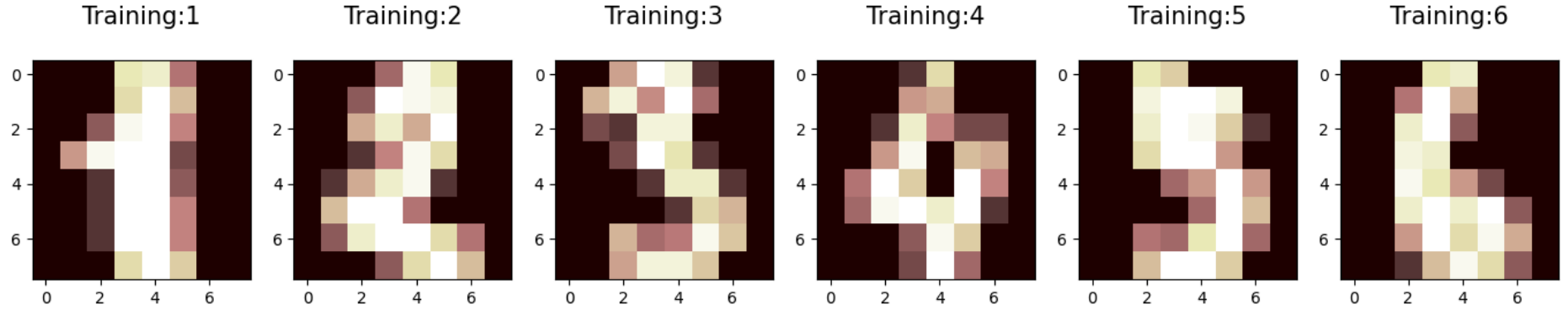
LogisticRegression to convert numbers into pics

In [1]:
```
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline
digits = load_digits()
```

In [2]:
```
print("Image Data shape",digits.data.shape)
print("Label Data shape",digits.target.shape)
```

```
Image Data shape (1797, 64)
Label Data shape (1797,)
```

In [77]:
```
plt.figure(figsize = (20,4))
for index,(image,Label) in enumerate(zip(digits.data[1:7],digits.target[1:7])):
    plt.subplot(1,7,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.pink)
    plt.title('Training:%i\n'%Label,fontsize=15)
```



In [8]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.30,random_state=2)
```

In [9]:
```
print(x_train.shape)
```

```
(1257, 64)
```

In [10]:
```
print(y_test.shape)
```

```
(540,)
```

In [11]:
```
print(x_test.shape)
```

```
(540, 64)
```

In [12]:
```
print(y_train.shape)
```

```
(1257,)
```

In [18]:
```
from sklearn.linear_model import LogisticRegression
LogisticRegr = LogisticRegression(max_iter=10000)
LogisticRegr.fit(x_train,y_train)
```

Out[18]:
```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [17]:
```
print(LogisticRegr.predict(x_test))
```

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

In [16]:
```
score = LogisticRegr.score(x_test,y_test)
print(score)
```

```
0.9537037037037037
```

LogisticRegression

Problem Statement:

To predict the risk of heart disease using Logistic Regression

In [83]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [84]:
```
df=pd.read_csv(r"C:\Users\HP\OneDrive\Documents\gender_submission.csv")
df
```

Out[84]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
5	897	0
6	898	1
7	899	0
8	900	1
9	901	0
10	902	0

In [85]:
```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [86]:
```
print('This Dataframe has %d Rows and %d columns'%(df.shape))
```

```
This Dataframe has 418 Rows and 2 columns
```

In [87]:
```
df.head()
```

Out[87]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [88]:
```
feature_matrix = df.iloc[:,0:2]
target_vector = df.iloc[:,-1]
```

In [89]:
```
print('The features matrix has %d rows and %d column(s)'%(feature_matrix.shape))
print('The target matrix has %d rows and %d column(s)'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The features matrix has 418 rows and 2 column(s)
The target matrix has 418 rows and 1 column(s)
```

In [90]:
```
feature_matrix_standardized = StandardScaler().fit_transform(feature_matrix)
```

In [91]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,
                             class_weight=None,random_state=None,solver='lbfgs',max_iter=100,multi_class='auto',
                             verbose=0,warm_start=False,n_jobs=None,l1_ratio=None)
```

In [92]:
```
Logistic_Regression_Model = algorithm.fit(feature_matrix_standardized,target_vector)
```

In [98]:
```
Observation = [[1,0]]
```

In [99]:
```
predictions=Logistic_Regression_Model.predict(Observation)
print('The Model predicted The observation to belong to class %s'%(predictions))
```

```
The Model predicted The observation to belong to class [0]
```

In [100]:
```
print('The algorithm was trained to predict one of the two classes: %s'%(algorithm.classes_))
```

```
The algorithm was trained to predict one of the two classes: [0 1]
```

In [101]:
```
el says the probability of the observation we passed Belonging to class ['b'] Is %s """%(algorithm.predict_proba(Observation)[0][0]))

el says the probability of the observation we passed Belonging to class ['g'] Is %s """%(algorithm.predict_proba(Observation)[0][1]))
```

```
The Model says the probability of the observation we passed Belonging to class ['b'] Is 0.8238872695984016

The Model says the probability of the observation we passed Belonging to class ['g'] Is 0.17611273040159833
```

In [ ]:
```

```