In [16]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_digits
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
```

Steps:

- Import digits data set from Scikitlearn datasets library. Use `load_digits()` . When loaded, the dataset comes with `data` and `target` values.
- Assign `data` to X and `target` to y
- Check the shape of the data
- Use `np.bincount` to print the number of uniqe elements of the target vriable `y`
- Split data into train and test datasets. Use stratification when splitting. You can set your `random_state` to `42`
- Normalize your dataset. When normalizing, simply divide your dataset by the maximum of the train dataset. To find the maximum, use `max()` . Example: to find the maximum of T, simply find T.max()
- Fit KneigborsClassifier and Logistic Regression on your data and check the score on the test dataset

In [ ]:

```
digits = load_digits()
X=digits.data
y=digits.target
```

In [11]:

```
X.shape
```

Out[11]:

```
(1797, 64)
```

In [12]:

```
y[1:10]
```

Out[12]:

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [17]:

```
np.bincount(y)
```

Out[17]:

```
array([178, 182, 177, 183, 181, 182, 181, 179, 174, 180])
```

In [13]:

```
X_train, X_test, y_train, y_test=train_test_split(X,y, stratify=y,random_state=42 )
```

In [14]:

```
X_max=X_train.max()
X_train=X_train/X_max
X_test=X_test/X_max
```

In [15]:

```
knn = KNeighborsClassifier()
logistic = LogisticRegression(max_iter=1000)

print('KNN score: %f' % knn.fit(X_train, y_train).score(X_test, y_test))
print('LogisticRegression score: %f'
      % logistic.fit(X_train, y_train).score(X_test, y_test))
```

```
KNN score: 0.984444
LogisticRegression score: 0.962222
```

In [ ]:

```

```