## Cross Validation

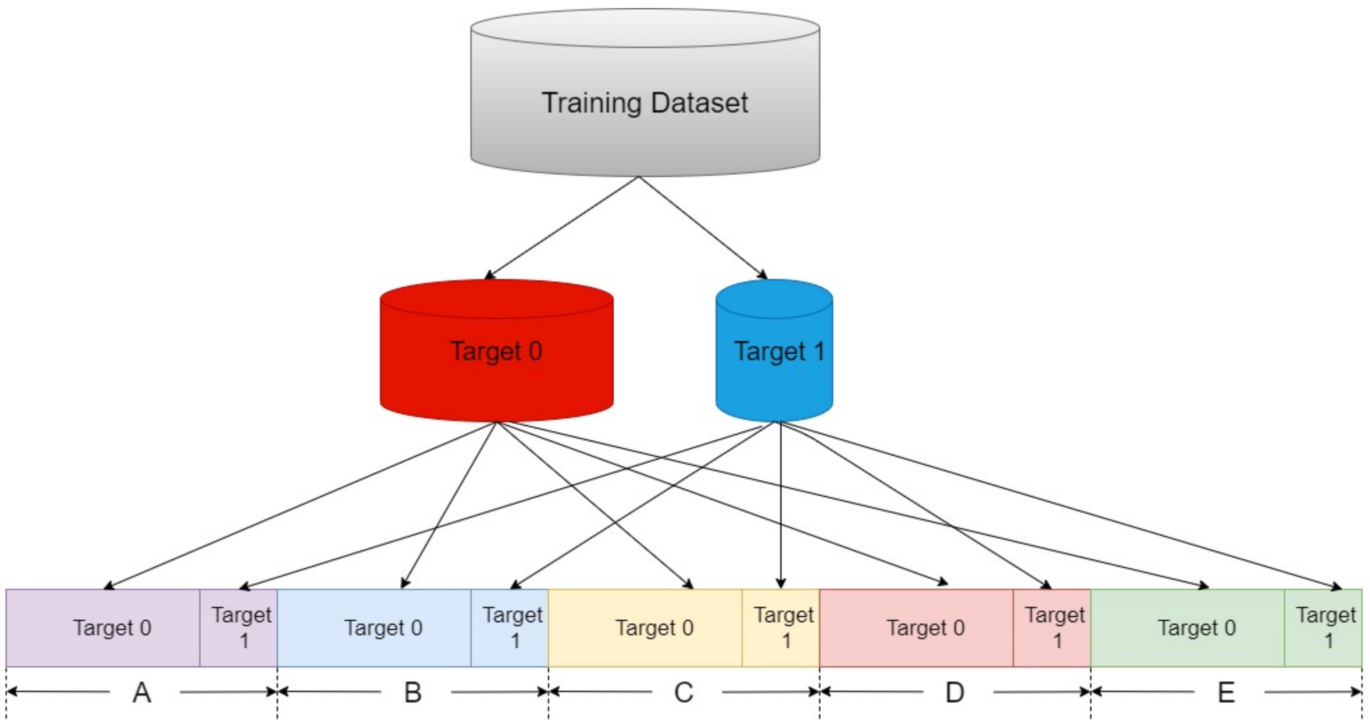
- Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting, ie, failing to generalize a pattern.
- The three steps involved in cross-validation are as follows :

```
1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.
```

***Here I Only Dicusss about Stratified k-fold cross validation***

## Stratified k-fold cross validation

- Stratified k-fold cross-validation is same as just k-fold cross-validation, But in Stratified k-fold cross-validation, it does stratified sampling instead of random sampling.
- One obvious problem with normal KFold, is that each in each fold the distribution of classes in the validation set, will be not be same. This is a big problem with imbalanced datasets.
- To overcome this problem we will use Stratified-KFold Validation. StratifiedKFold ensures that each of the splits have same proportion of examples of each class.
- StratifiedKFold is a variation of KFold. First, StratifiedKFold shuffles your data, after that splits the data into n_splits parts and Done. Now, it will use each part as a test set. Note that it only and always shuffles data one time before splitting.



## Demonstration

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix, roc_auc_score ,roc_curve,auc
from sklearn.model_selection import StratifiedKFold
import warnings
warnings.simplefilter('ignore')
```

In [2]:

```
cancer = load_breast_cancer()
df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
df['target'] = pd.Series(cancer.target)
df.head()
```

Out[2]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mear symmetr |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.241 |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.181 |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.206 |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.259 |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.180 |

5 rows × 31 columns

In [3]:

```
X = df.drop('target',axis=1)
y = df['target'].astype('category')
```

## Use manual train_test_split

In [4]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

In [5]:

```
lr_manual = LogisticRegression()
lr_manual.fit(X_train,y_train)
```

Out[5]:

```
LogisticRegression()
```

In [6]:

```
confusion_matrix(y_test,lr_manual.predict(X_test))
```

Out[6]:

```
array([[39,  5],
       [ 0, 70]], dtype=int64)
```

## Use StratifiedKFold

In [7]:

```
kf = StratifiedKFold(n_splits=5,shuffle=True,random_state=45)
pred_test_full =0
cv_score =[]
i=1
for train_index,test_index in kf.split(X,y):
    print('{} of KFold {}'.format(i,kf.n_splits))

    ### Training Set
    xtr,xvl = X.iloc[train_index],X.iloc[test_index]

    ### Validation Set
    ytr,yvl = y.iloc[train_index],y.iloc[test_index]

    #model
    lr = LogisticRegression(C=2)
    lr.fit(xtr,ytr)
    score = roc_auc_score(yvl,lr.predict(xvl))
    print('ROC AUC score:',score)
    cv_score.append(score)

#     pred_test = lr.predict_proba(x_test)[:,1]
#     pred_test_full +=pred_test
    i+=1
```

```
1 of KFold 5
ROC AUC score: 0.9415329184408779
2 of KFold 5
ROC AUC score: 0.932361611529643
3 of KFold 5
ROC AUC score: 0.9523809523809523
4 of KFold 5
ROC AUC score: 0.9692460317460316
5 of KFold 5
ROC AUC score: 0.9312541918175722
```

In [8]:

```
print('Confusion matrix\n',confusion_matrix(yvl,lr.predict(xvl)))
print('Cv',cv_score,'\nMean cv Score',np.mean(cv_score))
```

```
Confusion matrix
 [[38  4]
 [ 3 68]]
Cv [0.9415329184408779, 0.932361611529643, 0.9523809523809523, 0.96924603174
60316, 0.9312541918175722]
Mean cv Score 0.9453551411830154
```

**here I use logistic regression for demonstrate the k-fold. you can use any algorithm.**