Source of the Dataset

- https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

In [1]:
```
import pandas as pd
df = pd.read_csv('cancer.csv')
print(df.shape)
df.head()
```

```
(569, 33)
```

Out[1]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	point
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

In [2]:
```
df.dropna(axis = 1, inplace = True)
df.shape
```

Out[2]:
```
(569, 32)
```

In [3]:
```
df.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	point
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 32 columns

In [4]:
```
# Separate Independent (X) and Dependent(y) features

y = df['diagnosis']
X = df.drop(columns = ['id', 'diagnosis'])
```

In [5]:
```
X.head()
```

Out[5]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_me
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.24
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.18
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.20
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.25
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.18

5 rows × 30 columns

In [6]:
```
# check if the dataset is balanced or imbalanced

y.value_counts()
```

Out[6]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

HoldOut Validation Approach - Train-test split

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy ", accuracy_score(y_pred, y_test))
```

```
Accuracy  0.9239766081871345
```

K Fold Cross Validation

In [8]:
```
from sklearn.model_selection import KFold, cross_val_score
import numpy as np
kfold_validation = KFold(n_splits = 10)
clf = DecisionTreeClassifier()

results = cross_val_score(clf, X, y, cv = kfold_validation)
print(results)
print()
print ("Results = ", np.mean(results), "+/-", np.std(results))
```

```
[0.9122807  0.9122807  0.87719298 0.96491228 0.9122807  0.96491228
 0.89473684 0.98245614 0.92982456 0.89285714]

Results =  0.9243734335839597 +/- 0.03350584048302555
```

Stratified K Fold Cross Validation

In [9]:
```
# For imbalance dataset
from sklearn.model_selection import StratifiedKFold
skfold = StratifiedKFold(n_splits=5)
clf = DecisionTreeClassifier()
results = cross_val_score(clf, X, y, cv = skfold)
print(results)
print()
print ("Results = ", np.mean(results), "+/-", np.std(results))
```

```
[0.90350877 0.90350877 0.92105263 0.94736842 0.90265487]

Results =  0.915618692749573 +/- 0.0173143528579722
```

Leave One Out Cross Validation (LOOCV)

In [10]:
```
# not recommended for large dataset
from sklearn.model_selection import LeaveOneOut
clf = DecisionTreeClassifier()
leave = LeaveOneOut()
results = cross_val_score(clf, X, y, cv = leave)
print(results)
print()
print ("Results = ", np.mean(results), "+/-", np.std(results))
print ("Length of Results = ", len(results))
```

```
[1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.]

Results =  0.9226713532513181 +/- 0.2671121995355125
Length of Results =  569
```

Shuffle Split/ Repeated Random train-test splits

In [11]:
```
from sklearn.model_selection import ShuffleSplit
ssplit = ShuffleSplit(n_splits = 10, test_size = 0.30)
clf = DecisionTreeClassifier()
results = cross_val_score(clf, X, y, cv = ssplit)
print(results)
print()
print ("Results = ", np.mean(results), "+/-", np.std(results))
```

```
[0.92397661 0.94736842 0.88304094 0.94152047 0.92397661 0.94736842
 0.92982456 0.9005848  0.9005848  0.92397661]

Results =  0.9222222222222222 +/- 0.020434392058077736
```

In [ ]: