

Random Forest Classification

Import Libraries

```
In [ ]: # import libraries
import numpy as np
import pandas as pd
```

Load Dataset

```
In [ ]: #Load dataset
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
```

```
In [ ]: data.data
```

```
Out[3]: array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
                1.189e-01],
               [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
                8.902e-02],
               [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
                8.758e-02],
               ...,
               [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
                7.820e-02],
               [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
                1.240e-01],
               [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
                7.039e-02]])
```

```
In [ ]: data.feature_names
```

```
Out[4]: array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',  
              'mean smoothness', 'mean compactness', 'mean concavity',  
              'mean concave points', 'mean symmetry', 'mean fractal dimension',  
              'radius error', 'texture error', 'perimeter error', 'area error',  
              'smoothness error', 'compactness error', 'concavity error',  
              'concave points error', 'symmetry error',  
              'fractal dimension error', 'worst radius', 'worst texture',  
              'worst perimeter', 'worst area', 'worst smoothness',  
              'worst compactness', 'worst concavity', 'worst concave points',  
              'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
In [ ]: data.target
```

```
Out[5]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
               0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
               1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
               1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
               1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
               0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
               1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
               0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
               1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
               1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
               0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
               0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
               1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
               1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 0, 1, 0, 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, 0, 0, 0, 0, 0, 0, 1])
```

```
In [ ]: data.target_names
```

```
Out[6]: array(['malignant', 'benign'], dtype='<U9')
```

```
In [ ]: # create dataframe
df = pd.DataFrame(np.c_[data.data, data.target], columns=[list(data.feature_names)+['target']])
df.head()
```

Out[7]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	r
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	

```
In [ ]: df.tail()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	

```
In [ ]: df.shape
```

```
Out[9]: (569, 31)
```

Split Data

```
In [ ]: X = df.iloc[:, 0:-1]  
y = df.iloc[:, -1]
```

```
In [ ]: 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, random_state=2020)  
  
print('Shape of X_train = ', X_train.shape)  
print('Shape of y_train = ', y_train.shape)  
print('Shape of X_test = ', X_test.shape)  
print('Shape of y_test = ', y_test.shape)
```

```
Shape of X_train = (455, 30)
```

```
Shape of y_train = (455,)
```

```
Shape of X_test = (114, 30)
```

```
Shape of y_test = (114,)
```

Train Random Forest Classification Model

```
In [ ]: from sklearn.ensemble import RandomForestClassifier
```

```
In [ ]: classifier = RandomForestClassifier(n_estimators=100, criterion='gini')
        classifier.fit(X_train, y_train)
```

```
Out[13]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)
```

```
In [ ]: classifier.score(X_test, y_test)
```

```
Out[14]: 0.956140350877193
```

Predict Cancer

```
In [ ]: patient1 = [17.99,  
10.38,  
122.8,  
1001.0,  
0.1184,  
0.2776,  
0.3001,  
0.1471,  
0.2419,  
0.07871,  
1.095,  
0.9053,  
8.589,  
153.4,  
0.006399,  
0.04904,  
0.05373,  
0.01587,  
0.03003,  
0.006193,  
25.38,  
17.33,  
184.6,  
2019.0,  
0.1622,  
0.6656,  
0.7119,  
0.2654,  
0.4601,  
0.1189]
```

```
In [ ]: patient1 = np.array([patient1])
patient1
```

```
Out[16]: array([[1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01, 2.776e-01,
                3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00, 9.053e-01,
                8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02, 1.587e-02,
                3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02, 2.019e+03,
                1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01, 1.189e-01]])
```

```
In [ ]: classifier.predict(patient1)
```

```
Out[17]: array([0.])
```

```
In [ ]: data.target_names
```

```
Out[18]: array(['malignant', 'benign'], dtype='<U9')
```

```
In [ ]: pred = classifier.predict(patient1)
```

```
In [ ]: if pred[0] == 0:
        print('Patient has Cancer (malignant tumor)')
        else:
        print('Patient has no Cancer (malignant benign)')
```

Patient has Cancer (malignant tumor)

Save Model

Save Model using Pickle

```
In [ ]: import pickle
```



```
In [ ]: pickle.dump(classifier, open('model_save', 'wb'))
```

```
In [ ]: model = pickle.load(open('model_save', 'rb'))
```

```
In [ ]: model.predict(patient1)[0]
```

```
Out[37]: 0.0
```

Save Model using Joblib

```
In [ ]: import joblib
```

```
In [ ]: joblib.dump(classifier, 'model_save2')
```

```
Out[40]: ['model_save2']
```

```
In [ ]: model2 = joblib.load('model_save2')
```

```
In [ ]: model2.predict(patient1)
```

```
Out[42]: array([0.])
```

Ab milenge next tutorial me,Tab tak ke liye SIKHATE SIKHATE kuch IMPLEMENT karte raho, Thank You.....:-)