# LightGBM implementation in Python

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

# Read dataset

In [2]:

```
df=pd.read_csv(r"C:\Users\spoin\Downloads\Breast_cancer_data.csv")
df.head()
```

Out[2]:

| | mean_radius | mean_texture | mean_perimeter | mean_area | mean_smoothness | diagnosis |
|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0 |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0 |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0 |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0 |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0 |

# View summary of dataset

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   mean_radius      569 non-null    float64
 1   mean_texture     569 non-null    float64
 2   mean_perimeter   569 non-null    float64
 3   mean_area        569 non-null    float64
 4   mean_smoothness  569 non-null    float64
 5   diagnosis        569 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 26.8 KB
```

# Check the distribution of target variable

- target variable is diagnosis
- check the distribution of the target variable.

In [4]:

```
df['diagnosis'].value_counts()
```

Out[4]:

```
1    357
0    212
Name: diagnosis, dtype: int64
```

- The target variable is diagnosis. It contains 2 values - 0 and 1.
- 0 is for Negative prediction and 1 for Positive prediction.
- We can see that the problem is binary classification task.

# separate dependent and independent variable

In [5]:

```
X=df.iloc[:,:-1].values
y=df['diagnosis']
```

# Split dataset into training and test set

In [6]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.3,random_state=0)
```

# LightGBM Model Development and Training

In [7]:

```
import lightgbm as lgb
clf = lgb.LGBMClassifier()
clf.fit(X_train, y_train)
```

```
[LightGBM] [Info] Number of positive: 249, number of negative: 149
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000151 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 665
[LightGBM] [Info] Number of data points in the train set: 398, number o
f used features: 5
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.625628 -> initscore=
0.513507
[LightGBM] [Info] Start training from score 0.513507
[LightGBM] [Warning] No further splits with positive gain, best gain: -
inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -
inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -
inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -
inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -
```

# Model Prediction

In [8]:

```
y_pred=clf.predict(X_test)
```

# View Accuracy

In [9]:

```
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_pred,y_test)
print('LightGBM Model accuracy score: {0:0.4f}'.format(accuracy))
```

```
LightGBM Model accuracy score: 0.9298
```

# Check for Overfitting

In [10]:

```
bias=clf.score(X_train,y_train)
varience=clf.score(X_test,y_test)
print(bias)
print(varience)
```

```
1.0
0.9298245614035088
```

- The training and test set accuracy are quite comparable. So, we cannot say there is overfitting.

# Confusion-matrix

In [11]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix\n\n', cm)
print('\nTrue Positives(TP) = ', cm[0,0])
print('\nTrue Negatives(TN) = ', cm[1,1])
print('\nFalse Positives(FP) = ', cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[ 55   8]
 [  4 104]]

True Positives(TP) =  55

True Negatives(TN) =  104

False Positives(FP) =  8

False Negatives(FN) =  4
```

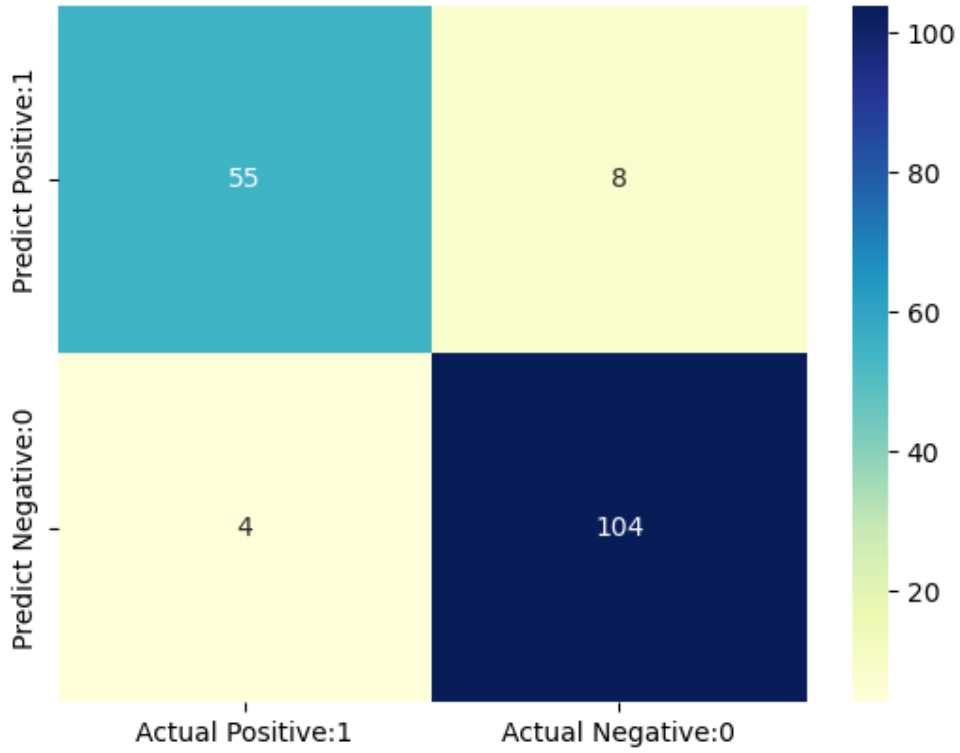
In [12]:

```
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                  index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

Out[12]:

```
<Axes: >
```




# Classification Metrices

In [13]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.87      0.90        63
           1       0.93      0.96      0.95       108

    accuracy                           0.93       171
   macro avg       0.93      0.92      0.92       171
weighted avg       0.93      0.93      0.93       171
```