```
import pandas as pd
df=pd.read_csv("/content/Breast_Cancer_Dataset.csv")
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_me
0	842302	M	17.99	10.38	122.80	1001
1	842517	M	20.57	17.77	132.90	1326
2	84300903	M	19.69	21.25	130.00	1203
3	84348301	M	11.42	20.38	77.58	386
4	84358402	M	20.29	14.34	135.10	1297
...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479
565	926682	M	20.13	28.25	131.20	1261
566	926954	M	16.60	28.08	108.30	858
567	927241	M	20.60	29.33	140.10	1265
568	92751	B	7.76	24.54	47.92	181

569 rows × 33 columns

```
df.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['diagnosis'] = le.fit_transform(df['diagnosis'])
```

```
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_me
0	842302	1	17.99	10.38	122.80	1001
1	842517	1	20.57	17.77	132.90	1326
2	84300903	1	19.69	21.25	130.00	1203
3	84348301	1	11.42	20.38	77.58	386
4	84358402	1	20.29	14.34	135.10	1297
...	...	...	...	...	...	
564	926424	1	21.56	22.39	142.00	1479
565	926682	1	20.13	28.25	131.20	1261
566	926954	1	16.60	28.08	108.30	858
567	927241	1	20.60	29.33	140.10	1265
568	92751	0	7.76	24.54	47.92	181

569 rows × 33 columns

```
x=df.drop(["diagnosis",'Unnamed: 32'],axis=1)
y=df.diagnosis
```

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64
```

```
x
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothn
0	842302	17.99	10.38	122.80	1001.0	
1	842517	20.57	17.77	132.90	1326.0	
2	84300903	19.69	21.25	130.00	1203.0	
3	84348301	11.42	20.38	77.58	386.1	
4	84358402	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	
564	926424	21.56	22.39	142.00	1479.0	
565	926682	20.13	28.25	131.20	1261.0	
566	926954	16.60	28.08	108.30	858.1	
567	927241	20.60	29.33	140.10	1265.0	
568	92751	7.76	24.54	47.92	181.0	

569 rows × 31 columns

```
from sklearn.model_selection import train_test_split, KFold
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_sta
kf=KFold(n_splits=5)
print("Data is splitinto following number of folds:")
kf.get_n_splits(x,y)
```

```
Data is splitinto following number of folds:
5
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
lr_model = LogisticRegression()
lr_model.fit(x_train, y_train)
lr_predictions = lr_model.predict(x_test)
lr_accuracy = accuracy_score(y_test, lr_predictions)
print("Logistic Regression Accuracy:", lr_accuracy*100)
```

```
Logistic Regression Accuracy: 63.1578947368421
```

```
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
nb_predictions = nb_model.predict(x_test)
nb_accuracy = accuracy_score(y_test, nb_predictions)
print("Naive Bayes Accuracy:", nb_accuracy*100)
```

```
Naive Bayes Accuracy: 63.1578947368421
```

```
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)
dt_predictions = dt_model.predict(x_test)
dt_accuracy = accuracy_score(y_test, dt_predictions)
print("Decision Tree Accuracy:",dt_accuracy*100)
```

```
Decision Tree Accuracy: 94.73684210526315
```

```
knn_model = KNeighborsClassifier()
knn_model.fit(x_train, y_train)
knn_predictions = knn_model.predict(x_test)
knn_accuracy = accuracy_score(y_test, knn_predictions)
print("K-NN Accuracy:",knn_accuracy*100)
```

```
K-NN Accuracy: 83.33333333333334
```

```
# SVM
svm_model = SVC()
svm_model.fit(x_train, y_train)
svm_predictions = svm_model.predict(x_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("SVM Accuracy:",svm_accuracy*100)
```

```
SVM Accuracy: 63.1578947368421
```

```
# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)
rf_predictions = rf_model.predict(x_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Random Forest Accuracy:",rf_accuracy*100)
```

```
Random Forest Accuracy: 95.6140350877193
```

```
pd.Series({"Logistic Regression Accuracy:": lr_accuracy*100,

      "Naive Bayes Accuracy:": nb_accuracy*100,

      "Decision Tree Accuracy:":dt_accuracy*100,

      "K-NN Accuracy:":knn_accuracy*100,

      "SVM Accuracy:":svm_accuracy*100,

      "Random Forest Accuracy:":rf_accuracy*100})
```

```
Logistic Regression Accuracy:    63.157895
Naive Bayes Accuracy:            63.157895
Decision Tree Accuracy:          94.736842
K-NN Accuracy:                   83.333333
SVM Accuracy:                    63.157895
Random Forest Accuracy:          95.614035
dtype: float64
```

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_test)
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test,y_pred)
cf
```

```
array([[70,  2],
       [17, 25]])
```

```
import seaborn as sns
import matplotlib.pyplot as plt
ax = sns.heatmap(cf,annot=True)

ax.set_title("Confusion Matrix with labels")
ax.set_xlabel("Predicted Values")
ax.set_ylabel("Actual Values")

ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

plt.show()
```