```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
```

Build the Linear regression and Logistic regression model on the dataset. Tune the parameters. Visualize the results. Measure the model performance using confusion matrix and ROC curve. Conclude with the summary of your findings. Dataset link: any tumor/cancer related dataset image dataset

```
#load the data
data = pd.read_csv('data.csv')
```

```
#print the data
print(data.shape)
print(data.head(6))
```

```
(569, 33)
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0
5    843786         M        12.45         15.70           82.57      477.1

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430
5          0.12780           0.17000          0.1578              0.08089

   ...  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  ...          17.33           184.60      2019.0            0.1622
1  ...          23.41           158.80      1956.0            0.1238
2  ...          25.53           152.50      1709.0            0.1444
3  ...          26.50            98.87       567.7            0.2098
4  ...          16.67           152.20      1575.0            0.1374
5  ...          23.75           103.40       741.6            0.1791

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364
5             0.5249           0.5355                0.1741          0.3985

   fractal_dimension_worst  Unnamed: 32
```

```
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN
5                  0.12440          NaN

[6 rows x 33 columns]
```

```
data.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
```

```
data['diagnosis'] = data['diagnosis'].apply(lambda x: 0 if x=='M' else 1)
```

```
y = data['diagnosis']
X = data.drop(['id','diagnosis','Unnamed: 32'], axis = 1)
```

```
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)
```

## Linear Regression Model

```
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

```
pred = model.predict(X_test)
y_pred1 = [ 0 if x < 0.5 else 1 for x in pred]
y_pred1
```

```
1,
1,
1,
1,
1,
1,
1,
1,
1,
1,
0,
1,
1,
0,
1,
0,
1,
0,
0,
1,
0,
1,
1,
0,
1,
1,
1,
0,
1,
1,
0,
0,
1,
1,
1,
1,
1,
0,
0,
0,
0,
1,
0,
1,
1,
1,
0,
1,
```

```
       1,
       1,
       0,
       1,
       0,
       1,
       1,
       0,
       1,
       0,
       1,
```

Logistic Regression Model

```
model2 = LogisticRegression()
model2.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818:
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

```
y_pred2 = model2.predict(X_test)
y_pred2
```

```
array([1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1])
```

# Confusion Matrix

Here we will create confusion matrices for both the models

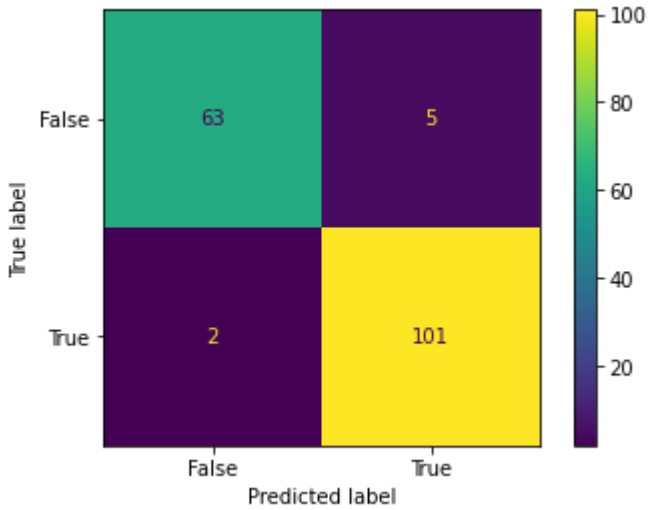
```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
cm1= confusion_matrix(y_test,y_pred1)
cm2= confusion_matrix(y_test,y_pred2)
print("Confusin Matrix: ")
print(cm1)
print(cm2)
```

```
Confusin Matrix:
[[ 63   5]
```

```
 [  2 101]]
[[63  5]
 [ 8 95]]
```

Confusion matrix for Linear Regression Model :

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm1, display_labels
cm_display.plot()
plt.show()
```

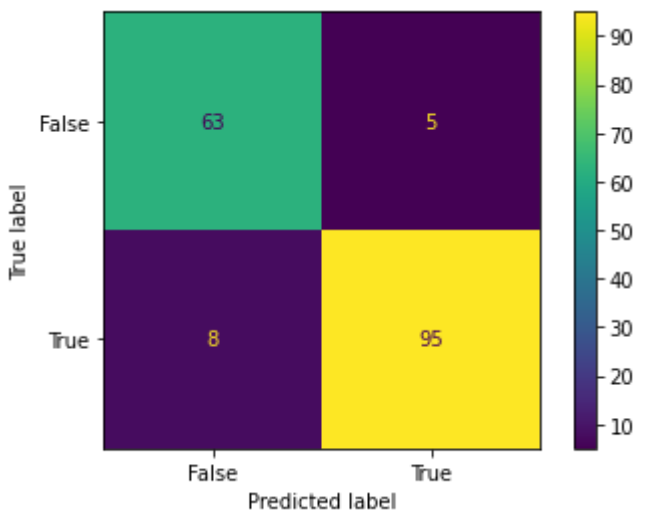



```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred1))
```

```
Accuracy: 0.9590643274853801
```

Confusion matrix for Logistic Regression Model :

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm2, display_labels
cm_display.plot()
plt.show()
```




```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred2))
```

```
Accuracy: 0.9239766081871345
```

# ROC Curve

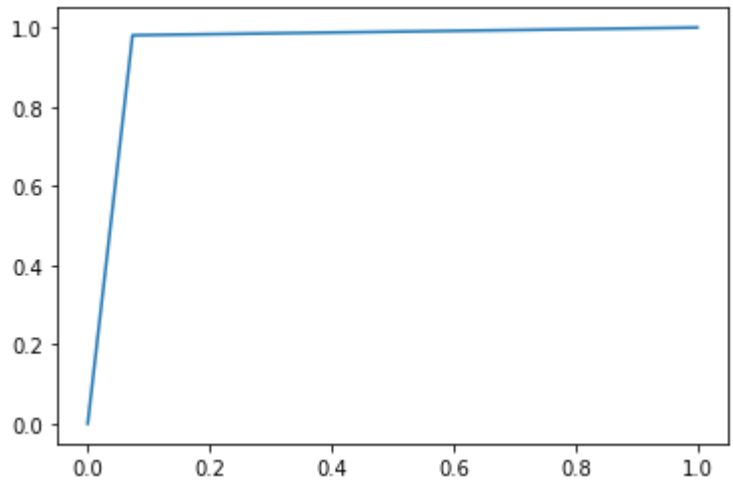
## ROC Curve for Linear Regression Model :

```
from sklearn.metrics import roc_curve, auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(false_positive_rate, true_positive_rate)
roc_auc
```

```
0.9535265562535694
```

```
plt.plot(false_positive_rate,true_positive_rate)
```

```
[<matplotlib.lines.Line2D at 0x7f345d1f3f50>]
```




## ROC Curve for Logistic Regression Model :

```
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(false_positive_rate, true_positive_rate)
roc_auc
```

```
0.9244003426613364
```

```
plt.plot(false_positive_rate,true_positive_rate)
```

```
[<matplotlib.lines.Line2D at 0x7f345d166f10>]
```