```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score
```

Double-click (or enter) to edit

```
df=pd.read_csv('/content/breast_cancer.csv')
```

```
df.head()
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Chro
0	5	1	1	1	2	1	
1	5	4	4	5	7	10	
2	3	1	1	1	2	2	
3	6	8	8	1	3	4	
4	4	1	1	3	2	1	

Next steps: Generate code with df | New interactive sheet

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683 entries, 0 to 682
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Clump Thickness              683 non-null    int64
 1   Uniformity of Cell Size      683 non-null    int64
```

```
 2   Uniformity of Cell Shape     683 non-null    int64
 3   Marginal Adhesion            683 non-null    int64
 4   Single Epithelial Cell Size  683 non-null    int64
 5   Bare Nuclei                  683 non-null    int64
 6   Bland Chromatin              683 non-null    int64
 7   Normal Nucleoli              683 non-null    int64
 8   Mitoses                      683 non-null    int64
 9   Class                        683 non-null    int64
dtypes: int64(10)
memory usage: 53.5 KB
```

Double-click (or enter) to edit

```
df.describe()
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Ba Nucl
count	683.000000	683.000000	683.000000	683.000000	683.000000	683.0000
mean	4.442167	3.150805	3.215227	2.830161	3.234261	3.5446
std	2.820761	3.065145	2.988581	2.864562	2.223085	3.6438
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000
25%	2.000000	1.000000	1.000000	1.000000	2.000000	1.0000
50%	4.000000	1.000000	1.000000	1.000000	2.000000	1.0000
75%	6.000000	5.000000	5.000000	4.000000	4.000000	6.0000
max	10.000000	10.000000	10.000000	10.000000	10.000000	10.0000

```
df.shape
```

```
(683, 10)
```

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

	0
Clump Thickness	0
Uniformity of Cell Size	0
Uniformity of Cell Shape	0
Marginal Adhesion	0
Single Epithelial Cell Size	0
Bare Nuclei	0
Bland Chromatin	0

Bland Chromatin	0
Normal Nucleoli	0
Mitoses	0
Class	0

dtype: int64

```
print(df['Class'].value_counts())
```

```
Class
2    444
4    239
Name: count, dtype: int64
```

```
df['Class']=df['Class'].map({2:0,4:1})
```

```
#splitting into features and target
X=df.drop('Class',axis=1)
Y=df['Class']
```

```
X_train,X_test,y_train,y_test=train_test_split(X, Y, test_size=0.2,ra
```

```
print("Training set shape",X_train.shape,y_train.shape)
print("Testing  set shape",X_test.shape,y_test.shape)
```

```
Training set shape (546, 9) (546,)
Testing  set shape (137, 9) (137,)
```

```
#computes the mean and standard deviation of training data then scale
#transform=applies the same scaling leaned from training data to the
#after scaling each feature will have mean=0 and sdt=1
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

```
model=LogisticRegression(random_state=2)
model.fit(X_train,y_train)
```

```
▾ LogisticRegression i ?
LogisticRegression(random_state=2)
```

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

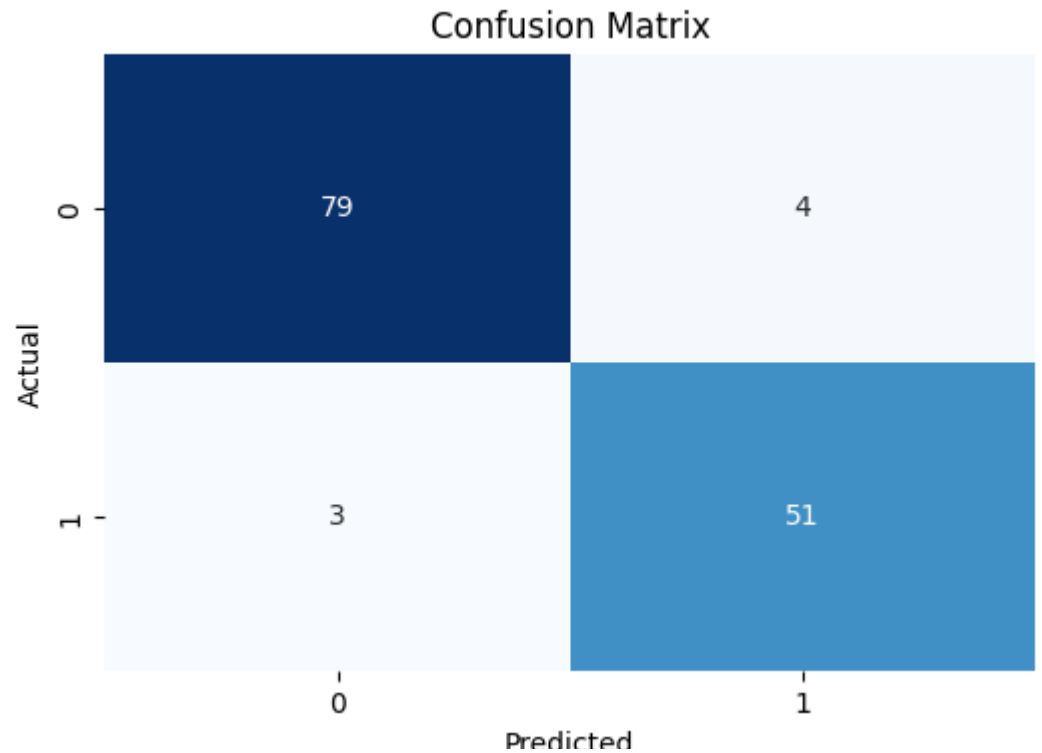
```
from sklearn.metrics import accuracy_score, confusion_matrix, classif
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_p
```

```
Accuracy: 0.948905109489051

Confusion Matrix:
 [[79  4]
 [ 3 51]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.96        83
           1       0.93      0.94      0.94        54

    accuracy                           0.95       137
   macro avg       0.95      0.95      0.95       137
weighted avg       0.95      0.95      0.95       137
```

```
cm = confusion_matrix(y_test, y_pred)
```

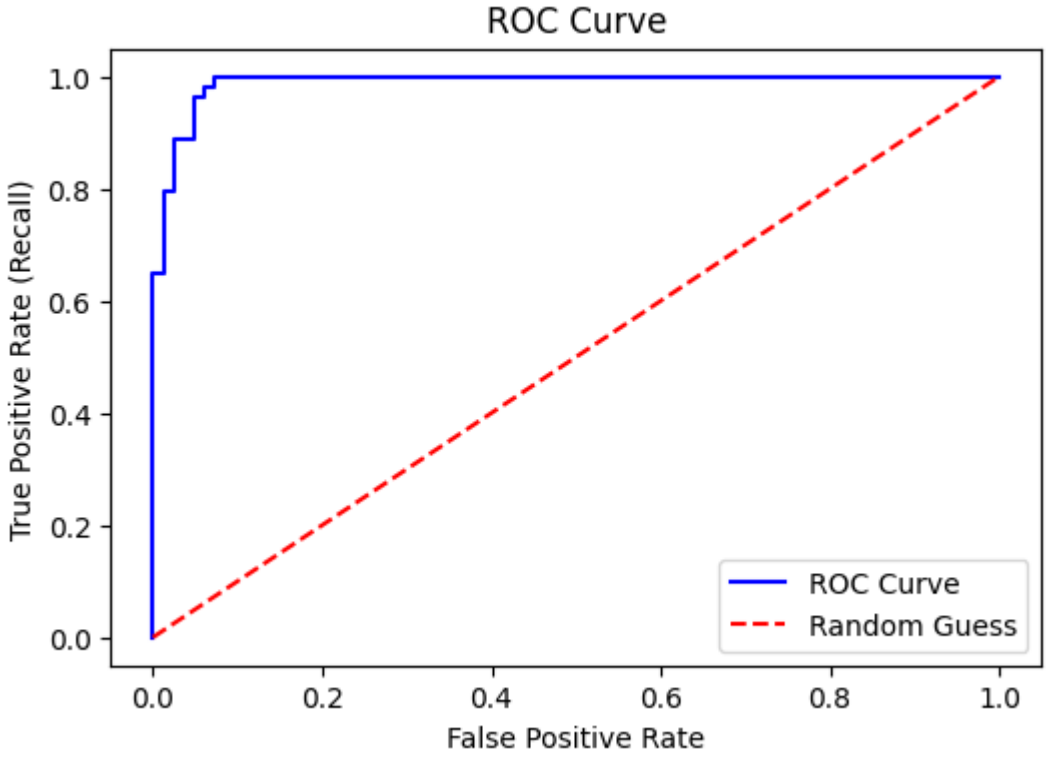
```
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



```
y_prob = model.predict_proba(X_test)[:, 1]
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
```

```
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, color='blue', label="ROC Curve")
plt.plot([0,1], [0,1], color='red', linestyle="--", label="Random Guess
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate (Recall)")
plt.title("ROC Curve")
plt.legend()
plt.show()
```




```
auc = roc_auc_score(y_test, y_prob)
print("AUC Score:", auc)
```

```
AUC Score: 0.9899598393574298
```