Objective(s):

- This activity aims to demonstrate how to apply simple linear regression analysis to solve regression problem

Intended Learning Outcomes (ILOs):

- Demonstrate how to solve classification problems using Logistic Regression
- Use the logistic regression model to perform classification

Resources:

Jupyter Notebook Dataset: https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29

Submission Requirements:

- PDF containing initial EDA and Data Wrangling
- PDF showing demonstration of simple linear regression.
- Submit a link to the colab file through the comment section.

```
import pandas as pd
import numpy as np
```

```
data = pd.read_csv('/content/risk_factors_cervical_cancer.csv')
```

```
print(data.head())
```

```
        Age  Number of sexual partners  First sexual intercourse  \
21   41                        3.0                      17.0
22   40                        1.0                      18.0
59   35                        3.0                      17.0
68   35                        3.0                      20.0
78   35                        3.0                      17.0

        Num of pregnancies Smokes  Smokes (years)  Smokes (packs/year)  \
21                 4.0    0.0             0.0                  0.0
22                 1.0    0.0             0.0                  0.0
59                 4.0    0.0             0.0                  0.0
68                 2.0    0.0             0.0                  0.0
78                 6.0    1.0            13.0                  2.6

       Hormonal Contraceptives  Hormonal Contraceptives (years)  IUD  ...  \
21                     1.0                            10.00  0.0  ...
22                     1.0                             0.25  0.0  ...
59                     1.0                             7.00  1.0  ...
68                     0.0                             0.00  1.0  ...
78                     1.0                             7.00  0.0  ...

        STDs: Time since first diagnosis STDs: Time since last diagnosis  \
21                              21.0                             21.0
22                               2.0                              2.0
59                              19.0                             19.0
68                               3.0                              3.0
78                              12.0                             12.0

        Dx:Cancer Dx:CIN Dx:HPV Dx Hinselmann Schiller Citology Biopsy
21          0      0      0  0          0        0        0      0
22          0      0      0  0          0        1        1      1
59          0      0      0  0          0        0        0      0
68          0      0      0  0          0        0        0      0
78          0      0      0  0          0        1        0      0

[5 rows x 36 columns]
```

```
print(data.columns)
```

```
Index(['Age', 'Number of sexual partners', 'First sexual intercourse',
       'Num of pregnancies', 'Smokes', 'Smokes (years)', 'Smokes (packs/year)',
       'Hormonal Contraceptives', 'Hormonal Contraceptives (years)', 'IUD',
       'IUD (years)', 'STDs', 'STDs (number)', 'STDs:condylomatosis',
       'STDs:cervical condylomatosis', 'STDs:vaginal condylomatosis',
       'STDs:vulvo-perineal condylomatosis', 'STDs:syphilis',
       'STDs:pelvic inflammatory disease', 'STDs:genital herpes',
       'STDs:molluscum contagiosum', 'STDs:AIDS', 'STDs:HIV',
       'STDs:Hepatitis B', 'STDs:HPV', 'STDs: Number of diagnosis',
       'STDs: Time since first diagnosis', 'STDs: Time since last diagnosis',
       'Dx:Cancer', 'Dx:CIN', 'Dx:HPV', 'Dx', 'Hinselmann', 'Schiller',
       'Citology', 'Biopsy'],
      dtype='object')
```

```
# Summary statistics and data types
print(data.describe())
```

```
              Age  Number of sexual partners  First sexual intercourse  \
count  59.000000                  59.000000                 59.000000
mean   27.457627                   2.711864                 17.050847
std     8.090697                   1.426967                  2.944450
min    15.000000                   1.000000                 10.000000
25%    20.000000                   2.000000                 15.000000
50%    28.000000                   3.000000                 17.000000
75%    33.500000                   3.000000                 19.000000
```

```
max    49.000000                  7.000000                28.000000

       Num of pregnancies  Smokes (years)  Smokes (packs/year)  \
count           59.000000       59.000000            59.000000
mean             2.525424        2.314135             0.694292
std              1.454552        4.736601             1.603797
min              0.000000        0.000000             0.000000
25%              1.500000        0.000000             0.000000
50%              2.000000        0.000000             0.000000
75%              3.500000        1.266973             0.125000
max              6.000000       16.000000             7.000000

       Hormonal Contraceptives (years)  IUD (years)  STDs (number)  \
count                        59.000000    59.000000      59.000000
mean                          1.948983     0.628475       1.728814
std                           2.996554     1.883241       0.715120
min                           0.000000     0.000000       1.000000
25%                           0.000000     0.000000       1.000000
50%                           0.250000     0.000000       2.000000
75%                           3.000000     0.000000       2.000000
max                          12.000000    10.000000       4.000000

       STDs: Number of diagnosis  STDs: Time since first diagnosis  \
count                  59.000000                         59.000000
mean                    1.050847                          6.101695
std                     0.289097                          6.016342
min                     1.000000                          1.000000
25%                     1.000000                          2.000000
50%                     1.000000                          4.000000
75%                     1.000000                          8.000000
max                     3.000000                         22.000000

       STDs: Time since last diagnosis  Dx:Cancer  Dx:CIN     Dx:HPV  \
count                        59.000000  59.000000    59.0  59.000000
mean                          6.016949   0.016949     0.0   0.016949
std                           6.061443   0.130189     0.0   0.130189
min                           1.000000   0.000000     0.0   0.000000
25%                           1.500000   0.000000     0.0   0.000000
50%                           3.000000   0.000000     0.0   0.000000
75%                           8.000000   0.000000     0.0   0.000000
max                          22.000000   1.000000     0.0   1.000000

              Dx  Hinselmann   Schiller   Citology     Biopsy
count  59.000000   59.000000  59.000000  59.000000  59.000000
mean    0.016949    0.067797   0.186441   0.118644   0.152542
std     0.130189    0.253555   0.392805   0.326145   0.362632
min     0.000000    0.000000   0.000000   0.000000   0.000000
25%     0.000000    0.000000   0.000000   0.000000   0.000000
50%     0.000000    0.000000   0.000000   0.000000   0.000000
75%     0.000000    0.000000   0.000000   0.000000   0.000000
```

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 59 entries, 21 to 831
Data columns (total 36 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Age                                 59 non-null     int64
 1   Number of sexual partners           59 non-null     float64
 2   First sexual intercourse            59 non-null     float64
 3   Num of pregnancies                  59 non-null     float64
 4   Smokes                              59 non-null     object
 5   Smokes (years)                      59 non-null     float64
 6   Smokes (packs/year)                 59 non-null     float64
 7   Hormonal Contraceptives             59 non-null     object
 8   Hormonal Contraceptives (years)     59 non-null     float64
 9   IUD                                 59 non-null     object
 10  IUD (years)                         59 non-null     float64
 11  STDs                                59 non-null     object
 12  STDs (number)                       59 non-null     float64
 13  STDs:condylomatosis                 59 non-null     object
 14  STDs:cervical condylomatosis        59 non-null     object
 15  STDs:vaginal condylomatosis         59 non-null     object
 16  STDs:vulvo-perineal condylomatosis  59 non-null     object
 17  STDs:syphilis                       59 non-null     object
 18  STDs:pelvic inflammatory disease    59 non-null     object
 19  STDs:genital herpes                 59 non-null     object
 20  STDs:molluscum contagiosum          59 non-null     object
 21  STDs:AIDS                           59 non-null     object
 22  STDs:HIV                            59 non-null     object
 23  STDs:Hepatitis B                    59 non-null     object
 24  STDs:HPV                            59 non-null     object
 25  STDs: Number of diagnosis           59 non-null     int64
 26  STDs: Time since first diagnosis    59 non-null     float64
 27  STDs: Time since last diagnosis     59 non-null     float64
 28  Dx:Cancer                           59 non-null     int64
 29  Dx:CIN                              59 non-null     int64
 30  Dx:HPV                              59 non-null     int64
 31  Dx                                  59 non-null     int64
 32  Hinselmann                          59 non-null     int64
 33  Schiller                            59 non-null     int64
 34  Citology                            59 non-null     int64
 35  Biopsy                              59 non-null     int64
dtypes: float64(10), int64(10), object(16)
memory usage: 17.1+ KB
None
```

```
missing_values = data.isnull().sum()
print("Missing values per column:")
print(missing_values)
```

```
Missing values per column:
Age                                 0
Number of sexual partners           0
First sexual intercourse            0
Num of pregnancies                  0
```

```
Smokes                              0
Smokes (years)                      0
Smokes (packs/year)                 0
Hormonal Contraceptives             0
Hormonal Contraceptives (years)     0
IUD                                 0
IUD (years)                         0
STDs                                0
STDs (number)                       0
STDs:condylomatosis                 0
STDs:cervical condylomatosis        0
STDs:vaginal condylomatosis         0
STDs:vulvo-perineal condylomatosis  0
STDs:syphilis                       0
STDs:pelvic inflammatory disease    0
STDs:genital herpes                 0
STDs:molluscum contagiosum          0
STDs:AIDS                           0
STDs:HIV                            0
STDs:Hepatitis B                    0
STDs:HPV                            0
STDs: Number of diagnosis           0
STDs: Time since first diagnosis    0
STDs: Time since last diagnosis     0
Dx:Cancer                           0
Dx:CIN                              0
Dx:HPV                              0
Dx                                  0
Hinselmann                          0
Schiller                            0
Citology                            0
Biopsy                              0
dtype: int64
```

```
# If there are missing values, visualize them using a heatmap
if missing_values.sum() > 0:
    plt.figure(figsize=(12, 8))  # Adjust the size to your preference
    sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
    plt.title('Missing Value Heatmap')
    plt.show()
else:
    print("No missing values detected in the dataset.")
```

```
No missing values detected in the dataset.
```

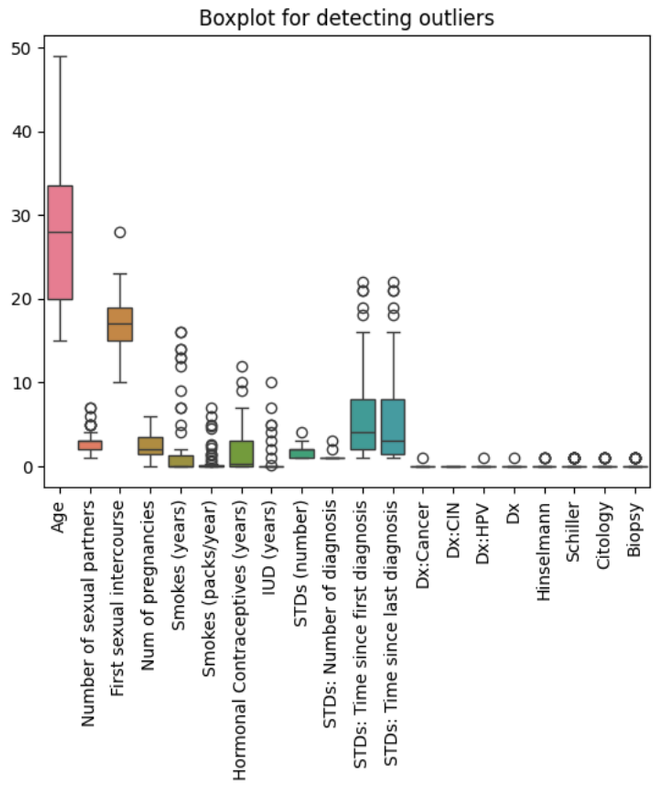
```
# Optional: Use an alternative method to mark and visualize missing values explicitly
if missing_values.sum() > 0:
    # Make a temporary copy of the dataset where NaNs are replaced with a unique value
    temp_data = data.fillna(-999)
    plt.figure(figsize=(12, 8))
    sns.heatmap(temp_data == -999, cbar=False, cmap='viridis')
    plt.title('Missing Value Heatmap - Explicit Null Marking')
    plt.show()
```

```
print("Data types of the dataset:")
print(data.dtypes)
```

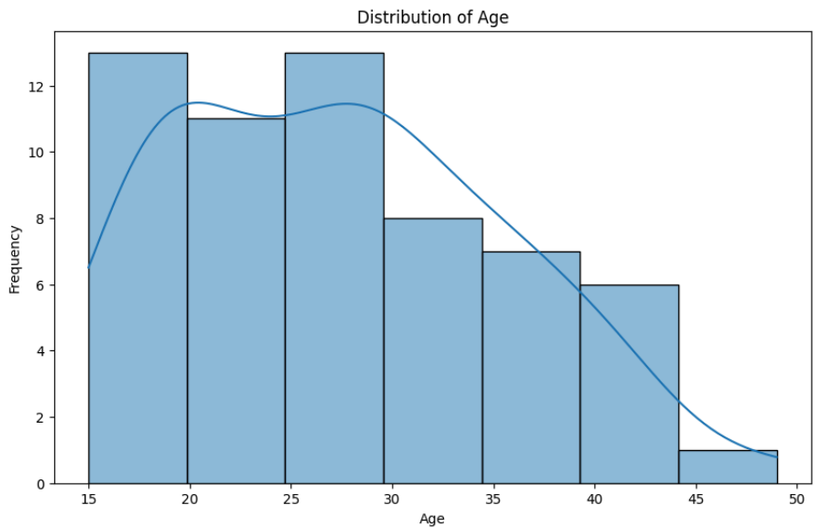
```
Data types of the dataset:
Age                                 int64
Number of sexual partners         float64
First sexual intercourse          float64
Num of pregnancies                float64
Smokes                             object
Smokes (years)                    float64
Smokes (packs/year)               float64
Hormonal Contraceptives            object
Hormonal Contraceptives (years)   float64
IUD                                object
IUD (years)                       float64
STDs                               object
STDs (number)                     float64
STDs:condylomatosis                object
STDs:cervical condylomatosis       object
STDs:vaginal condylomatosis        object
STDs:vulvo-perineal condylomatosis  object
STDs:syphilis                      object
STDs:pelvic inflammatory disease   object
STDs:genital herpes                object
STDs:molluscum contagiosum         object
STDs:AIDS                          object
STDs:HIV                           object
STDs:Hepatitis B                   object
STDs:HPV                           object
STDs: Number of diagnosis           int64
STDs: Time since first diagnosis  float64
STDs: Time since last diagnosis   float64
Dx:Cancer                           int64
Dx:CIN                              int64
Dx:HPV                              int64
Dx                                  int64
Hinselmann                          int64
Schiller                            int64
Citology                            int64
Biopsy                              int64
dtype: object
```

```
for column in data.columns:
    if data[column].dtype == 'float64' or data[column].dtype == 'int64':
        data[column].fillna(data[column].median(), inplace=True)
```

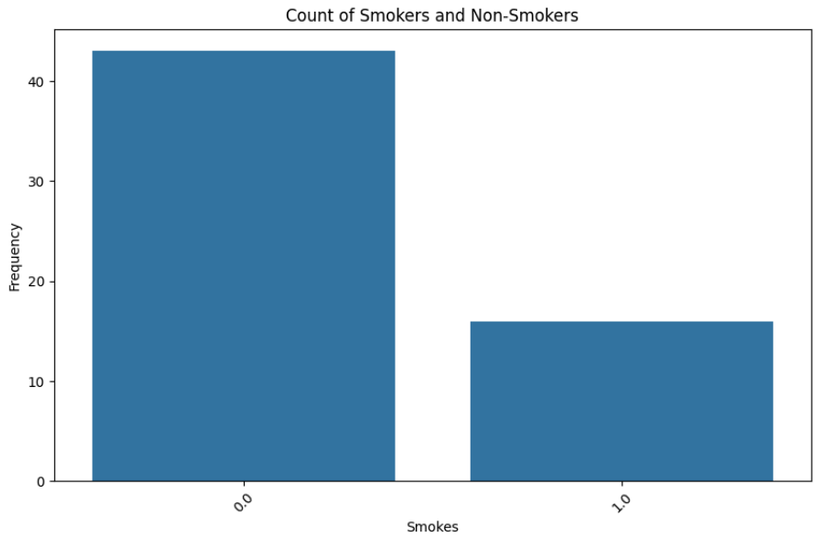
```
sns.boxplot(data=data)
plt.title('Boxplot for detecting outliers')
plt.xticks(rotation=90)
plt.show()
```



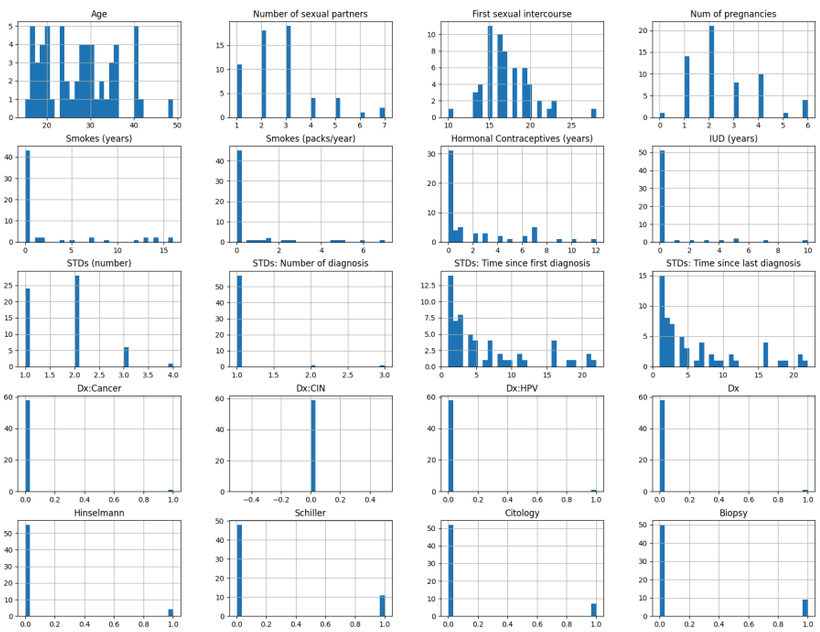

```
plt.figure(figsize=(10, 6))
sns.histplot(data['Age'], kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



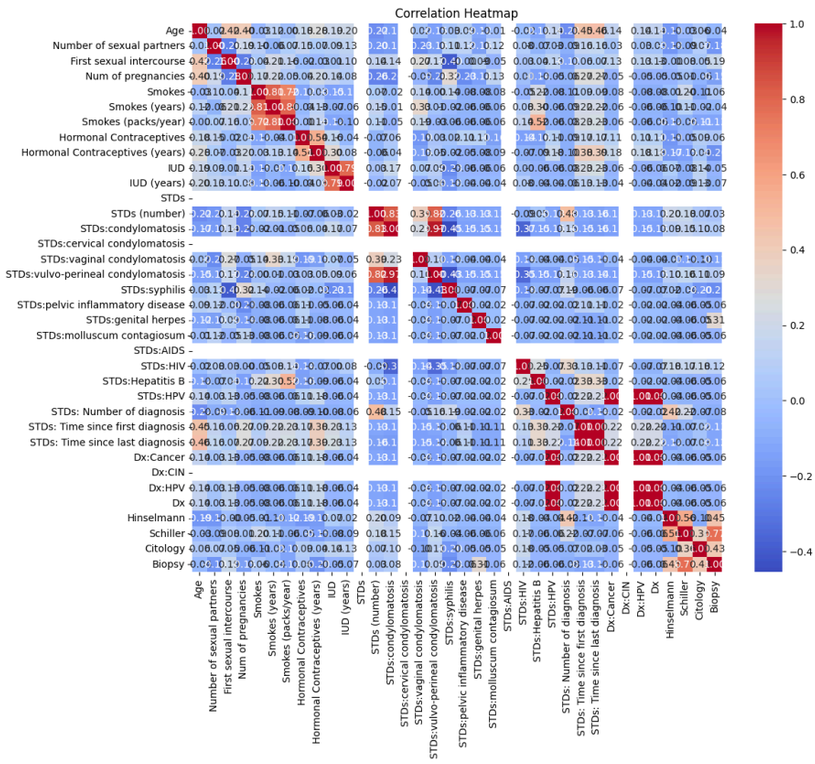

```
plt.figure(figsize=(10, 6))
sns.countplot(x=data['Smokes'].astype(str))
plt.title('Count of Smokers and Non-Smokers')
plt.xlabel('Smokes')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```




```
data.hist(figsize=(20, 15), bins=30)
plt.show()
```



```
plt.figure(figsize=(12, 10))
corr_matrix = data.corr()
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression  # Corrected this line
from sklearn.metrics import mean_squared_error, r2_score
```

```
data['Number of sexual partners'] = pd.to_numeric(data['Number of sexual partners'], errors='coerce')

data = data.dropna(subset=['Number of sexual partners'])

X = data[['Age']]
Y = data['Number of sexual partners']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

model = LinearRegression()
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)

mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 0.48677832650065506
R-squared: -0.16826798360157214
```

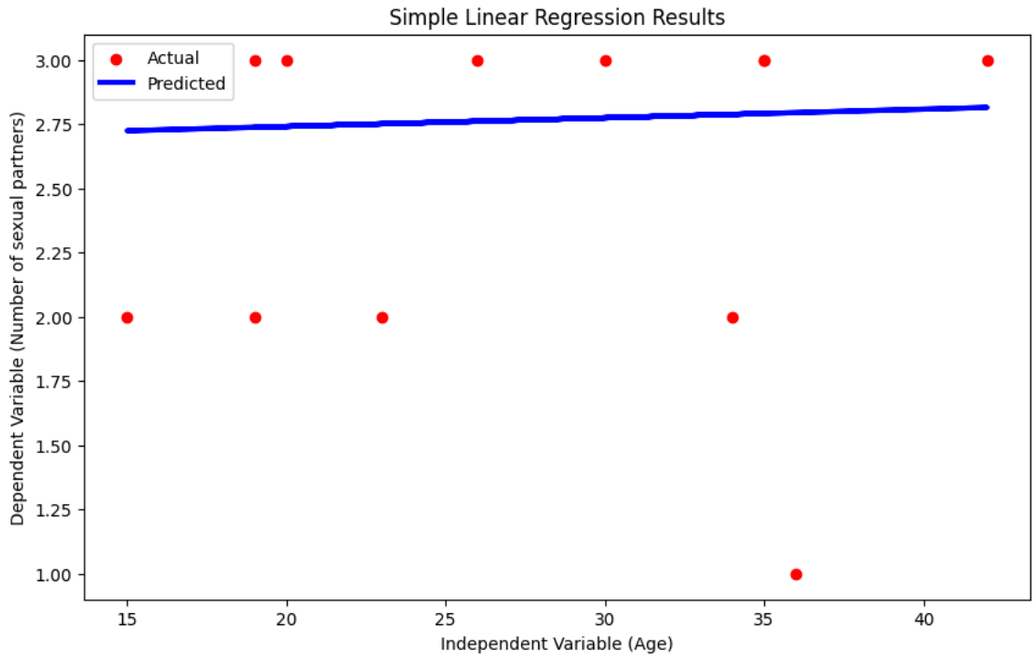
```
print('Coefficient(s):', model.coef_)

mse = mean_squared_error(Y_test, Y_pred)
print('Mean Squared Error: {:.2f}'.format(mse))

r_squared = r2_score(Y_test, Y_pred)
print('Coefficient of Determination: {:.2f}'.format(r_squared))

plt.figure(figsize=(10, 6))
plt.scatter(X_test, Y_test, color='red', label='Actual')
plt.plot(X_test, Y_pred, color='blue', linewidth=3, label='Predicted')
plt.title('Simple Linear Regression Results')
plt.xlabel('Independent Variable (Age)')
plt.ylabel('Dependent Variable (Number of sexual partners)')
plt.legend()
plt.show()
```

```
Coefficient(s): [0.00335662]
Mean Squared Error: 0.49
Coefficient of Determination: -0.17
```




```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
X = data.drop('Biopsy', axis=1)  # Features
Y = data['Biopsy']  # Target variable

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train, Y_train)

Y_pred = logistic_model.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
class_report = classification_report(Y_test, Y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix: \n{conf_matrix}")
print(f"Classification Report: \n{class_report}")
```

```
Accuracy: 0.9166666666666666
Confusion Matrix:
[[11  0]
 [ 1  0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        11
           1       0.00      0.00      0.00         1

    accuracy                           0.92        12
   macro avg       0.46      0.50      0.48        12
weighted avg       0.84      0.92      0.88        12

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Assuming you have a fitted model and a test set (X_test, Y_test)
# model.predict_proba(X_test)[:, 1] would give you the probabilities of the positive class
```

```
Y_probs = logistic_model.predict_proba(X_test)[:, 1]

# Calculate FPR, TPR, and the AUC
fpr, tpr, thresholds = roc_curve(Y_test, Y_probs)
roc_auc = auc(fpr, tpr)

# Plotting the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```