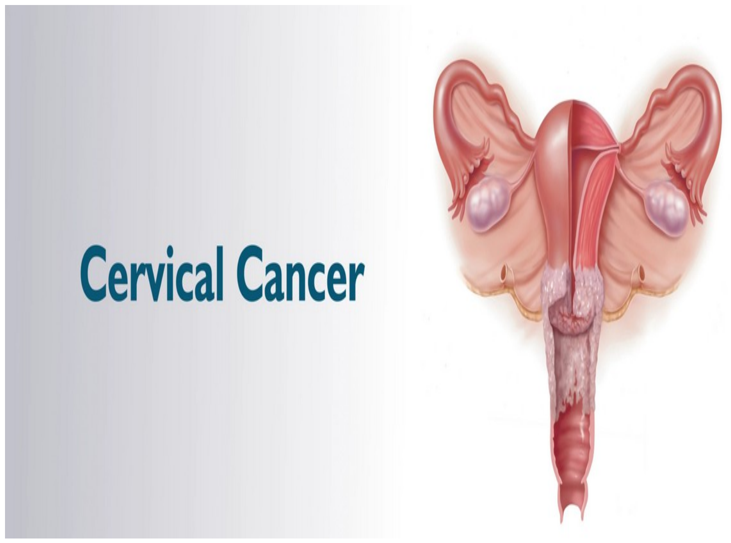
CERVICAL CANCER PREDICTION USING MACHINE LEARNING TECHNIQUES



INTRODUCTION

Cervical cancer is a significant health concern globally, particularly in low- and middle-income countries where regular screening and early detection are less accessible. The disease is primarily caused by persistent infection with high-risk human papillomavirus (HPV) types, making it a preventable and treatable condition if detected early. However, many women still face the risk of late-stage diagnosis, leading to higher mortality rates. The integration of machine learning models in the prediction and early diagnosis of cervical cancer can potentially reduce these risks by identifying high-risk individuals based on various factors and enabling timely intervention.

AIM & OBJECTIVES

The aim of this project is to develop a predictive model that can accurately assess the risk of cervical cancer in individuals..The use of machine learning algorithms, the project seeks to improve the early detection of cervical cancer, thereby aiding in the prevention and management of this disease.

Objectives

1. Data Understanding: I explored and understood the key variables
2. Data Preprocessing: I cleaned the dataset and handled missing values,Converted specified columns to binary values..
3. Exploratory Data Analysis: I analyzed the data to identify patterns and relationships.
4. Feature Engineering: I selected the important features for prediction Model Development: I built machine learning models to predict cervical cancer risk.

5. Model Evaluation:I evaluated the model performance using accuracy and other metrics.
6. Google Colab Implementation: I implemented the entire process in Google Colab.

DATASET

Dataset Features

1. **Age**: The age of the individual.
2. **Number of sexual partners**: The total number of sexual partners the individual has had.
3. **First sexual intercourse**: The age at which the individual had their first sexual intercourse.
4. **Num of pregnancies**: The total number of pregnancies the individual has had.
5. **Smokes**: Indicates whether the individual smokes (yes/no).
6. **Smokes (years)**: The number of years the individual has been smoking.
7. **Smokes (packs/year)**: The number of packs of cigarettes smoked per year.
8. **Hormonal Contraceptives**: Indicates whether the individual uses hormonal contraceptives (yes/no).
9. **Hormonal Contraceptives (years)**: The number of years the individual has been using hormonal contraceptives.
10. **IUD**: Indicates whether the individual has used an intrauterine device (IUD) (yes/no).
11. **IUD (years)**: The number of years the individual has used an intrauterine device (IUD).
12. **STDs**: Indicates whether the individual has had any sexually transmitted diseases (STDs) (yes/no).
13. **STDs (number)**: The total number of different STDs the individual has had.
14. **STDs: condylomatosis**: Indicates whether the individual has had condylomatosis (yes/no).
15. **STDs: cervical condylomatosis**: Indicates whether the individual has had cervical condylomatosis (yes/no).
16. **STDs: vaginal condylomatosis**: Indicates whether the individual has had vaginal condylomatosis (yes/no).
17. **STDs: vulvo-perineal condylomatosis**: Indicates whether the individual has had vulvo-perineal condylomatosis (yes/no).
18. **STDs: syphilis**: Indicates whether the individual has had syphilis (yes/no).
19. **STDs: pelvic inflammatory disease**: Indicates whether the individual has had pelvic inflammatory disease (yes/no).
20. **STDs: genital herpes**: Indicates whether the individual has had genital herpes (yes/no).
21. **STDs: molluscum contagiosum**: Indicates whether the individual has had molluscum contagiosum (yes/no).
22. **STDs: AIDS**: Indicates whether the individual has had AIDS (yes/no).
23. **STDs: HIV**: Indicates whether the individual has had HIV (yes/no).
24. **STDs: Hepatitis B**: Indicates whether the individual has had Hepatitis B (yes/no).
25. **STDs: HPV**: Indicates whether the individual has had human papillomavirus (HPV) (yes/no).
26. **STDs: Number of diagnosis**: The total number of STD diagnoses the individual has had.

27. **STDs: Time since first diagnosis**: The time in years since the individual's first STD diagnosis.
28. **STDs: Time since last diagnosis**: The time in years since the individual's last STD diagnosis.
29. **Dx: Cancer**: Indicates whether the individual has been diagnosed with cancer (yes/no).
30. **Dx: CIN**: Indicates whether the individual has been diagnosed with cervical intraepithelial neoplasia (CIN) (yes/no).
31. **Dx: HPV**: Indicates whether the individual has been diagnosed with HPV (yes/no).
32. **Dx**: General diagnosis (yes/no).
33. **Hinselmann**: A result from a Hinselmann test (used in cervical cancer detection) (yes/no).
34. **Schiller**: A result from a Schiller test (used in cervical cancer detection) (yes/no).
35. **Citology**: A result from a cytology test (Pap smear) (yes/no).
36. **Biopsy**: A result from a biopsy (yes/no).

MODELS

1. Random Forest (RF): 100%
2. Support Vector Machine (SVM): 98.20%
3. Logistic Regression (LR): Mean Squared Error: 0.012662802197635, R-squared: 0.45835446243889166
4. AdaBoost: 98.80%
5. CatBoost: 100%

Among these models, Random Forest and Catboost achieved the highest accuracy, making it the most effective for predicting cervical cancer risk in this case.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/content/cervical_cancer dataset.csv')

df.head()

{"type":"dataframe","variable_name":"df"}

df.shape

(835, 36)

df.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')
```

```
df.dtypes
```

```
Age                                     int64
Number of sexual partners             float64
First sexual intercourse              float64
Num of pregnancies                    float64
Smokes                                float64
Smokes (years)                        float64
Smokes (packs/year)                   float64
Hormonal Contraceptives               float64
Hormonal Contraceptives (years)       float64
IUD                                   float64
IUD (years)                           float64
STDs                                  float64
STDs (number)                         float64
STDs:condylomatosis                   float64
STDs:cervical condylomatosis          float64
STDs:vaginal condylomatosis           float64
STDs:vulvo-perineal condylomatosis    float64
STDs:syphilis                         float64
STDs:pelvic inflammatory disease      float64
STDs:genital herpes                   float64
STDs:molluscum contagiosum            float64
STDs:AIDS                             float64
STDs:HIV                              float64
STDs:Hepatitis B                      float64
STDs:HPV                              float64
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
```

```
df.describe()
```

```
{"type":"dataframe"}
```

```
df.duplicated().sum()
```

```
0
```

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

```
Age                                     0
Number of sexual partners              25
First sexual intercourse                7
Num of pregnancies                     56
Smokes                                 13
Smokes (years)                         13
Smokes (packs/year)                    13
Hormonal Contraceptives               103
Hormonal Contraceptives (years)       103
IUD                                   112
IUD (years)                           112
STDs                                  100
STDs (number)                         100
STDs:condylomatosis                   100
STDs:cervical condylomatosis          100
STDs:vaginal condylomatosis           100
STDs:vulvo-perineal condylomatosis    100
STDs:syphilis                         100
STDs:pelvic inflammatory disease      100
STDs:genital herpes                   100
STDs:molluscum contagiosum            100
STDs:AIDS                             100
STDs:HIV                              100
STDs:Hepatitis B                      100
STDs:HPV                              100
STDs: Number of diagnosis               0
STDs: Time since first diagnosis      764
STDs: Time since last diagnosis       764
Dx:Cancer                               0
Dx:CIN                                  0
Dx:HPV                                  0
Dx                                      0
Hinselmann                              0
Schiller                                0
Citology                                0
Biopsy                                  0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 835 entries, 0 to 834
Data columns (total 36 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Age                                 835 non-null    int64  
 1   Number of sexual partners           810 non-null    float64
 2   First sexual intercourse            828 non-null    float64
 3   Num of pregnancies                  779 non-null    float64
 4   Smokes                              822 non-null    float64
 5   Smokes (years)                      822 non-null    float64
 6   Smokes (packs/year)                 822 non-null    float64
 7   Hormonal Contraceptives             732 non-null    float64
 8   Hormonal Contraceptives (years)     732 non-null    float64
 9   IUD                                 723 non-null    float64
 10  IUD (years)                         723 non-null    float64
 11  STDs                                735 non-null    float64
 12  STDs (number)                       735 non-null    float64
 13  STDs:condylomatosis                 735 non-null    float64
 14  STDs:cervical condylomatosis        735 non-null    float64
 15  STDs:vaginal condylomatosis         735 non-null    float64
 16  STDs:vulvo-perineal condylomatosis  735 non-null    float64
 17  STDs:syphilis                       735 non-null    float64
 18  STDs:pelvic inflammatory disease    735 non-null    float64
 19  STDs:genital herpes                 735 non-null    float64
 20  STDs:molluscum contagiosum          735 non-null    float64
 21  STDs:AIDS                           735 non-null    float64
 22  STDs:HIV                            735 non-null    float64
 23  STDs:Hepatitis B                    735 non-null    float64
 24  STDs:HPV                            735 non-null    float64
 25  STDs: Number of diagnosis           835 non-null    int64  
 26  STDs: Time since first diagnosis    71 non-null     float64
 27  STDs: Time since last diagnosis     71 non-null     float64
 28  Dx:Cancer                           835 non-null    int64  
 29  Dx:CIN                              835 non-null    int64  
 30  Dx:HPV                              835 non-null    int64  
 31  Dx                                  835 non-null    int64  
 32  Hinselmann                          835 non-null    int64  
 33  Schiller                            835 non-null    int64  
 34  Citology                            835 non-null    int64  
 35  Biopsy                              835 non-null    int64  
dtypes: float64(26), int64(10)
memory usage: 235.0 KB
```

```
df.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')


float_cols = df.select_dtypes(include=['float64']).columns
def round_value(x):
    return round(x, 2)

for col in float_cols:
    df[col] = df[col].map(round_value)

df.head()

{"type":"dataframe","variable_name":"df"}

def count_decimal_places(value):
    if isinstance(value, float):
        s = str(value)
        if '.' in s:
            return len(s.split('.')[1])
    return 0

decimal_places = df.applymap(count_decimal_places).max()

print(decimal_places)

Age                                      0
Number of sexual partners                1
First sexual intercourse                 1
Num of pregnancies                       1
Smokes                                   1
Smokes (years)                           2
Smokes (packs/year)                      2
Hormonal Contraceptives                  1
Hormonal Contraceptives (years)          2
IUD                                      1
IUD (years)                              2
STDs                                     1
STDs (number)                         1
STDs:condylomatosis                   1
STDs:cervical condylomatosis          1
STDs:vaginal condylomatosis           1
STDs:vulvo-perineal condylomatosis    1
STDs:syphilis                         1
STDs:pelvic inflammatory disease      1
STDs:genital herpes                   1
STDs:molluscum contagiosum            1
STDs:AIDS                             1
STDs:HIV                              1
STDs:Hepatitis B                      1
STDs:HPV                              1
STDs: Number of diagnosis             0
STDs: Time since first diagnosis      1
STDs: Time since last diagnosis       1
Dx:Cancer                             0
Dx:CIN                                0
Dx:HPV                                0
Dx                                    0
Hinselmann                            0
Schiller                              0
Citology                              0
Biopsy                                0
dtype: int64

<ipython-input-91-bc148da93585>:8: FutureWarning: DataFrame.applymap
has been deprecated. Use DataFrame.map instead.
  decimal_places = df.applymap(count_decimal_places).max()
```

```
for column in df.columns:
    print(f"Column: {column}")
    print("Unique values:")
    print(df[column].unique())
    print()
```

```
Column: Age
Unique values:
[18 15 34 52 46 42 51 26 45 44 27 43 40 41 39 37 38 36 35 33 31 32 30
23
 28 29 20 25 21 24 22 48 19 17 16 14 59 79 84 47 13 70 50 49]

Column: Number of sexual partners
Unique values:
[ 4.  1.  5.  3.  2.  6. nan  7. 15.  8. 10. 28.  9.]

Column: First sexual intercourse
Unique values:
[15. 14. nan 16. 21. 23. 17. 26. 20. 25. 18. 27. 19. 24. 32. 13. 29.
11.
```

```
 12. 22. 28. 10.]

Column: Num of pregnancies
Unique values:
[ 1.  4.  2.  6.  3.  5. nan  8.  7.  0. 11. 10.]

Column: Smokes
Unique values:
[ 0.  1. nan]

Column: Smokes (years)
Unique values:
[ 0.   37.   34.    1.27  3.   12.     nan 18.    7.   19.   21.   15.
 13.   16.    8.    4.   10.   22.   14.    0.5  11.    9.    2.    5.
  6.    1.   32.   24.   28.   20.    0.16]

Column: Smokes (packs/year)
Unique values:
[0.00e+00 3.70e+01 3.40e+00 2.80e+00 4.00e-02 5.10e-01 2.40e+00
6.00e+00
      nan 9.00e+00 1.60e+00 1.90e+01 2.10e+01 3.20e-01 2.60e+00 8.00e-
01
 1.50e+01 2.00e+00 5.70e+00 1.00e+00 3.30e+00 3.50e+00 1.20e+01 3.00e-
02
 2.75e+00 2.00e-01 1.40e+00 5.00e+00 2.10e+00 7.00e-01 1.20e+00
7.50e+00
 1.25e+00 3.00e+00 7.50e-01 1.00e-01 8.00e+00 2.25e+00 7.00e+00 4.50e-
01
 1.50e-01 5.00e-02 2.50e-01 4.80e+00 4.50e+00 4.00e-01 3.70e-01
2.20e+00
 1.60e-01 9.00e-01 2.20e+01 1.35e+00 5.00e-01 2.50e+00 4.00e+00
1.30e+00
 1.65e+00 2.70e+00 7.60e+00 5.50e+00 3.00e-01]

Column: Hormonal Contraceptives
Unique values:
[ 0.  1. nan]

Column: Hormonal Contraceptives (years)
Unique values:
[ 0.    3.   15.    2.    8.   10.    5.    0.25  7.   22.   19.
0.5
  1.    0.58  9.   13.   11.    4.   12.   16.    0.33   nan  0.16 14.
  0.08  2.28  0.66  6.    1.5   0.42  0.67  0.75  2.5   4.5   6.5
0.17
 20.    3.5   0.41 30.   17.  ]

Column: IUD
Unique values:
[ 0.  1. nan]
```

```
Column: IUD (years)
Unique values:
[ 0.    7.     nan  5.    8.    6.    1.    0.58  2.   19.    0.5  17.
  0.08  0.25 10.   11.    3.   15.   12.    9.    1.5   0.91  4.
0.33
  0.41  0.16  0.17]

Column: STDs
Unique values:
[ 0.  1. nan]

Column: STDs (number)
Unique values:
[ 0.  2.  1. nan  3.  4.]

Column: STDs:condylomatosis
Unique values:
[ 0.  1. nan]

Column: STDs:cervical condylomatosis
Unique values:
[ 0. nan]

Column: STDs:vaginal condylomatosis
Unique values:
[ 0. nan  1.]

Column: STDs:vulvo-perineal condylomatosis
Unique values:
[ 0.  1. nan]

Column: STDs:syphilis
Unique values:
[ 0.  1. nan]

Column: STDs:pelvic inflammatory disease
Unique values:
[ 0. nan  1.]

Column: STDs:genital herpes
Unique values:
[ 0. nan  1.]

Column: STDs:molluscum contagiosum
Unique values:
[ 0. nan  1.]

Column: STDs:AIDS
Unique values:
```

```
[ 0. nan]

Column: STDs:HIV
Unique values:
[ 0.  1. nan]

Column: STDs:Hepatitis B
Unique values:
[ 0. nan  1.]

Column: STDs:HPV
Unique values:
[ 0. nan  1.]

Column: STDs: Number of diagnosis
Unique values:
[0 1 3 2]

Column: STDs: Time since first diagnosis
Unique values:
[nan 21.  2. 15. 19.  3. 12.  1. 11.  9.  7.  8. 16.  6.  5. 10.  4.
22.
 18.]

Column: STDs: Time since last diagnosis
Unique values:
[nan 21.  2. 15. 19.  3. 12.  1. 11.  9.  7.  8. 16.  6.  5. 10.  4.
22.
 18.]

Column: Dx:Cancer
Unique values:
[0 1]

Column: Dx:CIN
Unique values:
[0 1]

Column: Dx:HPV
Unique values:
[0 1]

Column: Dx
Unique values:
[0 1]

Column: Hinselmann
Unique values:
[0 1]
```

```
Column: Schiller
Unique values:
[0 1]

Column: Citology
Unique values:
[0 1]

Column: Biopsy
Unique values:
[0 1]
```

```python
import pandas as pd
columns_to_convert = [
    'Smokes', 'Hormonal Contraceptives', 'IUD', 'STDs',
    'STDs:condylomatosis', 'STDs:vulvo-perineal condylomatosis',
    'STDs:vaginal condylomatosis', 'STDs:HIV', 'STDs:Hepatitis B',
'STDs:HPV'
]

df[columns_to_convert] = df[columns_to_convert].applymap(lambda x: 1
if x >= 0.5 else 0).astype(int)
for col in columns_to_convert:
    print(f"Column: {col}\nUnique values:\n{df[col].unique()}")
```

```
Column: Smokes
Unique values:
[0 1]
Column: Hormonal Contraceptives
Unique values:
[0 1]
Column: IUD
Unique values:
[0 1]
Column: STDs
Unique values:
[0 1]
Column: STDs:condylomatosis
Unique values:
[0 1]
Column: STDs:vulvo-perineal condylomatosis
Unique values:
[0 1]
Column: STDs:vaginal condylomatosis
Unique values:
[0 1]
Column: STDs:HIV
Unique values:
[0 1]
```

```
Column: STDs:Hepatitis B
Unique values:
[0 1]
Column: STDs:HPV
Unique values:
[0 1]
```

```
<ipython-input-93-db0d7a8c7cae>:8: FutureWarning: DataFrame.applymap
has been deprecated. Use DataFrame.map instead.
  df[columns_to_convert] = df[columns_to_convert].applymap(lambda x: 1
if x >= 0.5 else 0).astype(int)
```

```python
import pandas as pd
for col in df.columns:
    print(f"Column: {col}\nUnique values:\n{df[col].unique()}\n")
```

```
Column: Age
Unique values:
[18 15 34 52 46 42 51 26 45 44 27 43 40 41 39 37 38 36 35 33 31 32 30
23
 28 29 20 25 21 24 22 48 19 17 16 14 59 79 84 47 13 70 50 49]

Column: Number of sexual partners
Unique values:
[ 4.  1.  5.  3.  2.  6. nan  7. 15.  8. 10. 28.  9.]

Column: First sexual intercourse
Unique values:
[15. 14. nan 16. 21. 23. 17. 26. 20. 25. 18. 27. 19. 24. 32. 13. 29.
11.
 12. 22. 28. 10.]

Column: Num of pregnancies
Unique values:
[ 1.  4.  2.  6.  3.  5. nan  8.  7.  0. 11. 10.]

Column: Smokes
Unique values:
[0 1]

Column: Smokes (years)
Unique values:
[ 0.   37.   34.    1.27  3.   12.     nan 18.    7.   19.   21.   15.
 13.   16.    8.    4.   10.   22.   14.    0.5  11.    9.    2.    5.
  6.    1.   32.   24.   28.   20.    0.16]

Column: Smokes (packs/year)
Unique values:
[0.00e+00 3.70e+01 3.40e+00 2.80e+00 4.00e-02 5.10e-01 2.40e+00
6.00e+00
      nan 9.00e+00 1.60e+00 1.90e+01 2.10e+01 3.20e-01 2.60e+00 8.00e-
01
 1.50e+01 2.00e+00 5.70e+00 1.00e+00 3.30e+00 3.50e+00 1.20e+01 3.00e-
02
 2.75e+00 2.00e-01 1.40e+00 5.00e+00 2.10e+00 7.00e-01 1.20e+00
7.50e+00
 1.25e+00 3.00e+00 7.50e-01 1.00e-01 8.00e+00 2.25e+00 7.00e+00 4.50e-
01
 1.50e-01 5.00e-02 2.50e-01 4.80e+00 4.50e+00 4.00e-01 3.70e-01
2.20e+00
 1.60e-01 9.00e-01 2.20e+01 1.35e+00 5.00e-01 2.50e+00 4.00e+00
1.30e+00
 1.65e+00 2.70e+00 7.60e+00 5.50e+00 3.00e-01]

Column: Hormonal Contraceptives
Unique values:
[0 1]

Column: Hormonal Contraceptives (years)
Unique values:
[ 0.    3.   15.    2.    8.   10.    5.    0.25  7.   22.   19.
0.5
  1.    0.58  9.   13.   11.    4.   12.   16.    0.33   nan  0.16 14.
  0.08  2.28  0.66  6.    1.5   0.42  0.67  0.75  2.5   4.5   6.5
0.17
 20.    3.5   0.41 30.   17.  ]

Column: IUD
Unique values:
[0 1]

Column: IUD (years)
Unique values:
[ 0.    7.     nan  5.    8.    6.    1.    0.58  2.   19.    0.5  17.
  0.08  0.25 10.   11.    3.   15.   12.    9.    1.5   0.91  4.
0.33
  0.41  0.16  0.17]

Column: STDs
Unique values:
[0 1]

Column: STDs (number)
Unique values:
[ 0.  2.  1. nan  3.  4.]

Column: STDs:condylomatosis
Unique values:
[0 1]

Column: STDs:cervical condylomatosis
```

```
Unique values:
[ 0. nan]

Column: STDs:vaginal condylomatosis
Unique values:
[0 1]

Column: STDs:vulvo-perineal condylomatosis
Unique values:
[0 1]

Column: STDs:syphilis
Unique values:
[ 0.  1. nan]

Column: STDs:pelvic inflammatory disease
Unique values:
[ 0. nan  1.]

Column: STDs:genital herpes
Unique values:
[ 0. nan  1.]

Column: STDs:molluscum contagiosum
Unique values:
[ 0. nan  1.]

Column: STDs:AIDS
Unique values:
[ 0. nan]

Column: STDs:HIV
Unique values:
[0 1]

Column: STDs:Hepatitis B
Unique values:
[0 1]

Column: STDs:HPV
Unique values:
[0 1]

Column: STDs: Number of diagnosis
Unique values:
[0 1 3 2]

Column: STDs: Time since first diagnosis
Unique values:
[nan 21.  2. 15. 19.  3. 12.  1. 11.  9.  7.  8. 16.  6.  5. 10.  4.
22.
 18.]

Column: STDs: Time since last diagnosis
Unique values:
[nan 21.  2. 15. 19.  3. 12.  1. 11.  9.  7.  8. 16.  6.  5. 10.  4.
22.
 18.]

Column: Dx:Cancer
Unique values:
[0 1]

Column: Dx:CIN
Unique values:
[0 1]

Column: Dx:HPV
Unique values:
[0 1]

Column: Dx
Unique values:
[0 1]

Column: Hinselmann
Unique values:
[0 1]

Column: Schiller
Unique values:
[0 1]

Column: Citology
Unique values:
[0 1]

Column: Biopsy
Unique values:
[0 1]
```

```
data= df.fillna(df.mean())
```

```
data.isnull().sum()
```

```
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
```

```
age_max_partners = data.groupby('Age')['Number of sexual
partners'].max()

age_max_partners
```

```
Age
13     1.0
14     5.0
15     4.0
16    28.0
17     5.0
18     7.0
19     7.0
20     5.0
21     5.0
22     4.0
23     8.0
```

```
24     5.0
25    15.0
26    10.0
27     6.0
28     8.0
29     8.0
30     5.0
31     9.0
32     7.0
33     5.0
34     5.0
35     6.0
36     6.0
37     6.0
38     4.0
39     5.0
40     3.0
41     4.0
42     3.0
43     4.0
44     3.0
45     5.0
46     3.0
47     2.0
48     4.0
49     3.0
50     2.0
51     3.0
52     5.0
59     2.0
70     4.0
79     2.0
84     3.0
Name: Number of sexual partners, dtype: float64
```

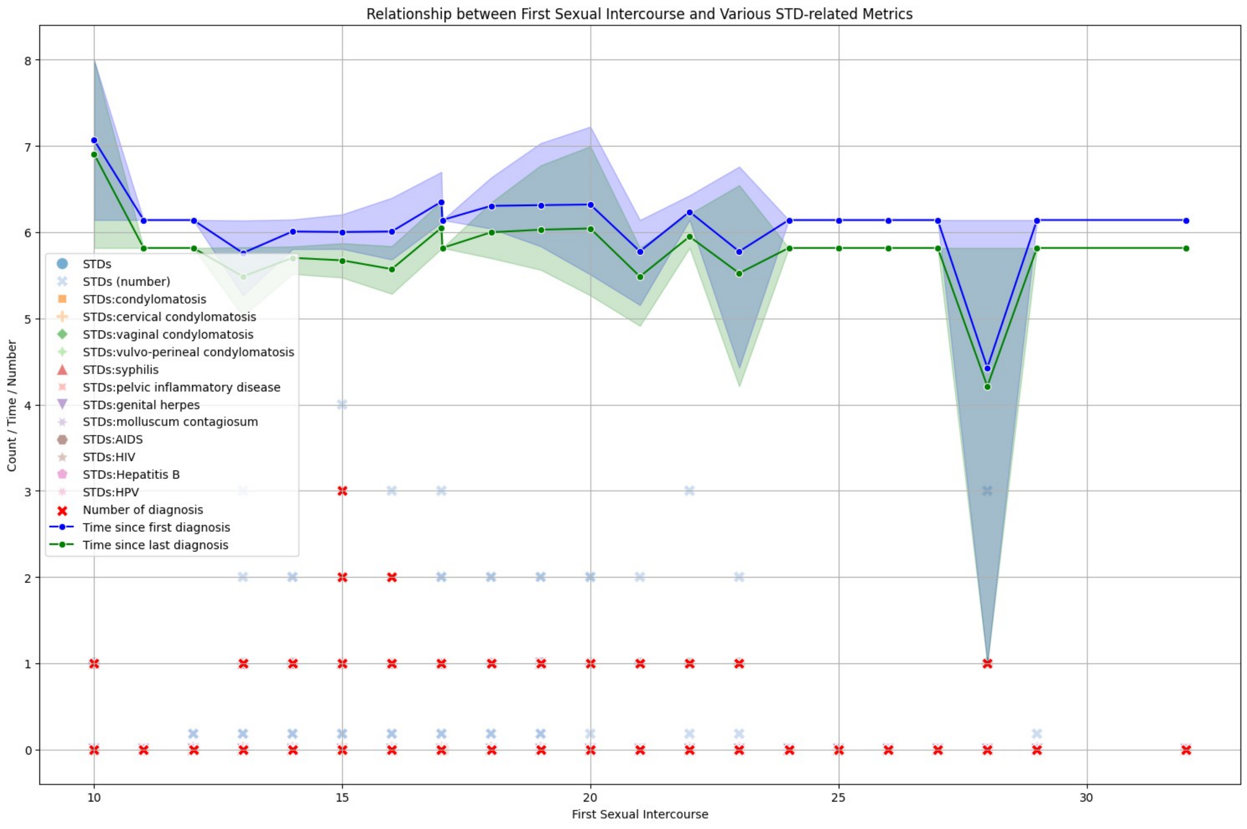
```
grouped_sexual_by_intercourse = data.groupby('First sexual
intercourse')['Num of pregnancies'].apply(list).reset_index()

grouped_sexual_by_intercourse
```

{"summary":"{\n \"name\": \"grouped_sexual_by_intercourse\",\n
\"rows\": 22,\n \"fields\": [\n   {\n   \"column\": \"First
sexual intercourse\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 6.2595969211372635,\n   \"min\":
10.0,\n   \"max\": 32.0,\n   \"num_unique_values\": 22,\n
\"samples\": [\n   10.0,\n   22.0,\n
17.020531400966185\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Num
of pregnancies\",\n   \"properties\": {\n   \"dtype\":
\"object\",\n   \"semantic_type\": \"\",\n

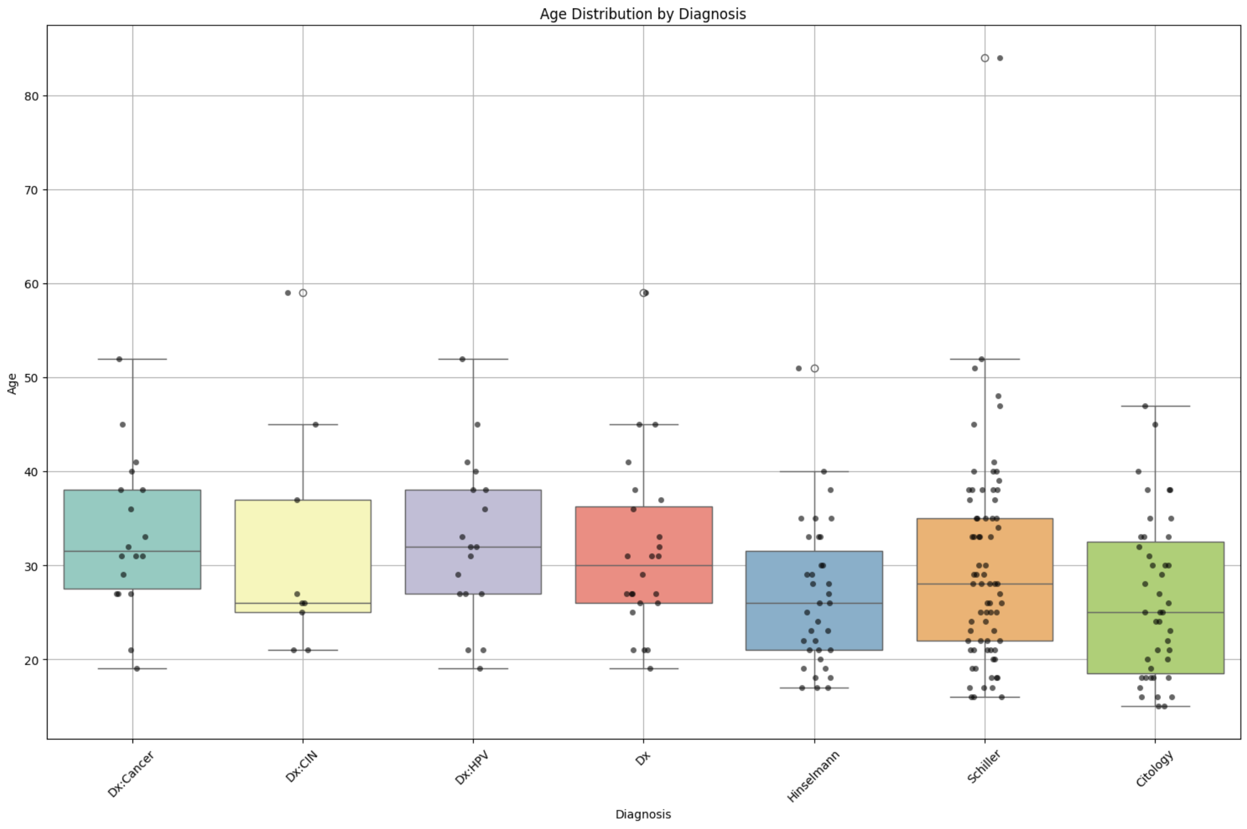
```
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"grouped_sexual_by_intercourse"
}
```

DATA VISUALIZATION

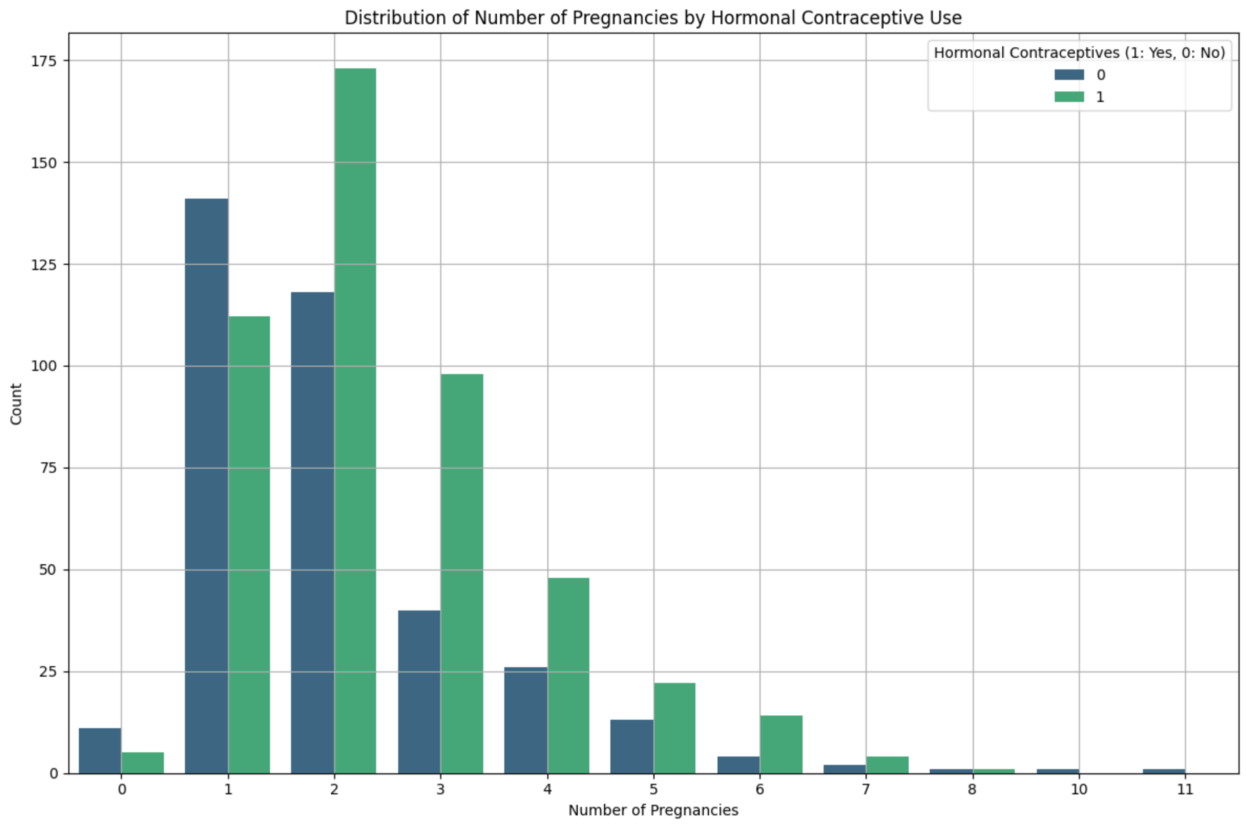
```
plt.figure(figsize=(15, 10))

std_columns = [
    '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'
]

data_melted = data.melt(id_vars=['First sexual intercourse', 'STDs:
Number of diagnosis', 'STDs: Time since first diagnosis', 'STDs: Time
since last diagnosis'],
                        value_vars=std_columns,
                        var_name='STD Type',
                        value_name='Count')

sns.scatterplot(data=data_melted, x='First sexual intercourse',
y='Count', hue='STD Type', style='STD Type', palette='tab20', s=100,
alpha=0.6)
sns.scatterplot(data=data, x='First sexual intercourse', y='STDs:
Number of diagnosis', label='Number of diagnosis', color='red',
marker='X', s=100)
sns.lineplot(data=data, x='First sexual intercourse', y='STDs: Time
since first diagnosis', label='Time since first diagnosis',
color='blue', marker='o')
sns.lineplot(data=data, x='First sexual intercourse', y='STDs: Time
since last diagnosis', label='Time since last diagnosis',
color='green', marker='o')

plt.title('Relationship between First Sexual Intercourse and Various
STD-related Metrics')
plt.xlabel('First Sexual Intercourse')
plt.ylabel('Count / Time / Number')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



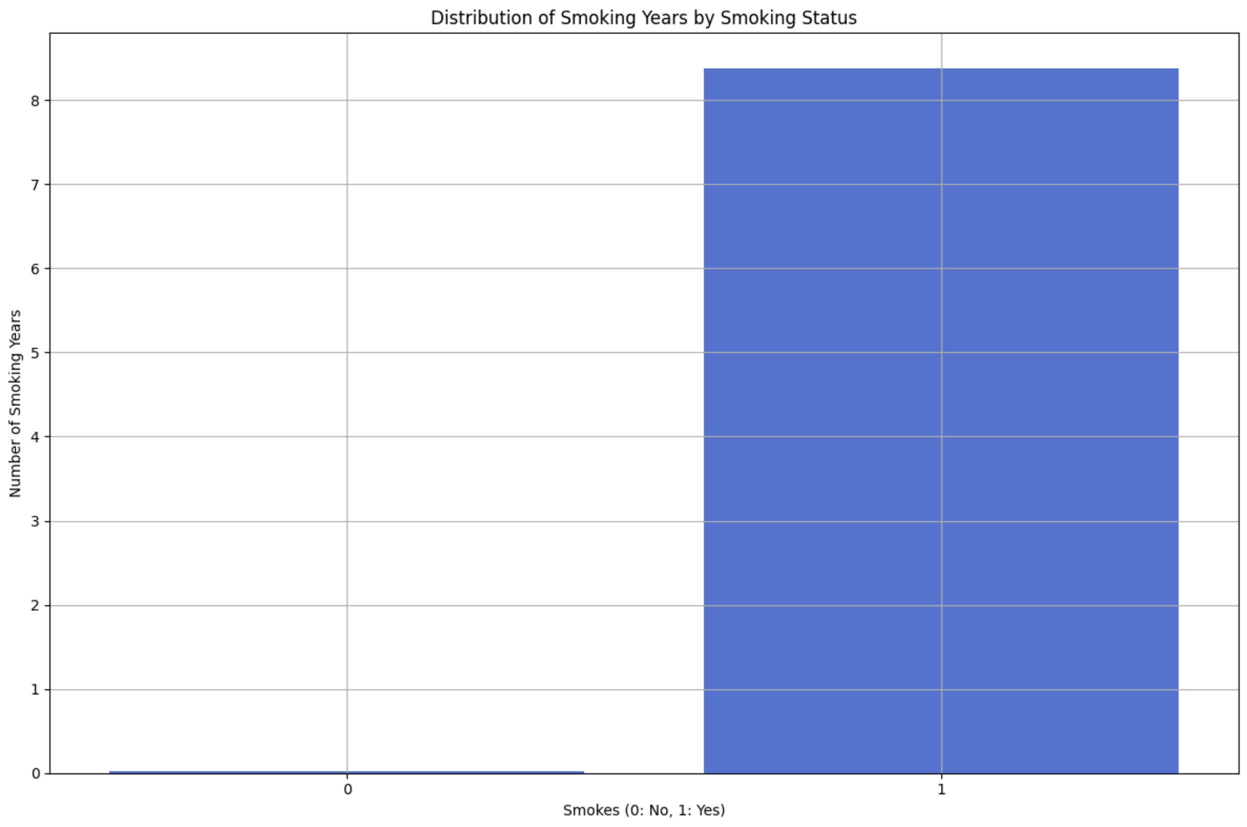
```
diagnosis_columns = [
    'Dx:Cancer', 'Dx:CIN', 'Dx:HPV', 'Dx', 'Hinselmann', 'Schiller',
'Citology'
]

data_melted = data.melt(id_vars=['Age'],
                        value_vars=diagnosis_columns,
                        var_name='Diagnosis',
                        value_name='Presence')

data_melted = data_melted[data_melted['Presence'] == 1]
plt.figure(figsize=(15, 10))
sns.boxplot(data=data_melted, x='Diagnosis', y='Age', hue='Diagnosis',
palette='Set3', dodge=False, legend=False)

sns.stripplot(data=data_melted, x='Diagnosis', y='Age', color='black',
alpha=0.6)

plt.title('Age Distribution by Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Age')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



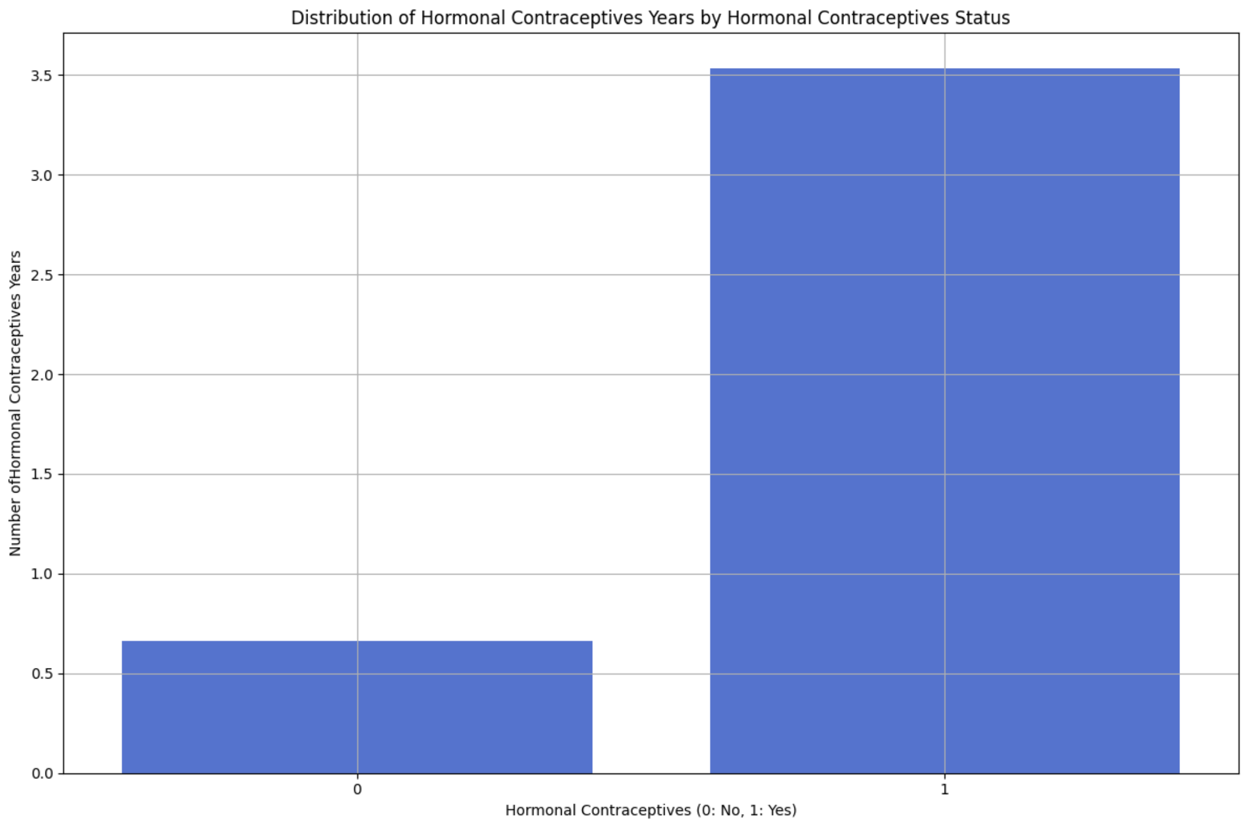
```
data['Num of pregnancies'] = data['Num of pregnancies'].astype(int)
plt.figure(figsize=(12, 8))

barplot = sns.countplot(
    data=data,
    x='Num of pregnancies',
    hue='Hormonal Contraceptives',
    palette='viridis'
)

plt.title('Distribution of Number of Pregnancies by Hormonal
Contraceptive Use')
plt.xlabel('Number of Pregnancies')
plt.ylabel('Count')
plt.legend(title='Hormonal Contraceptives (1: Yes, 0: No)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



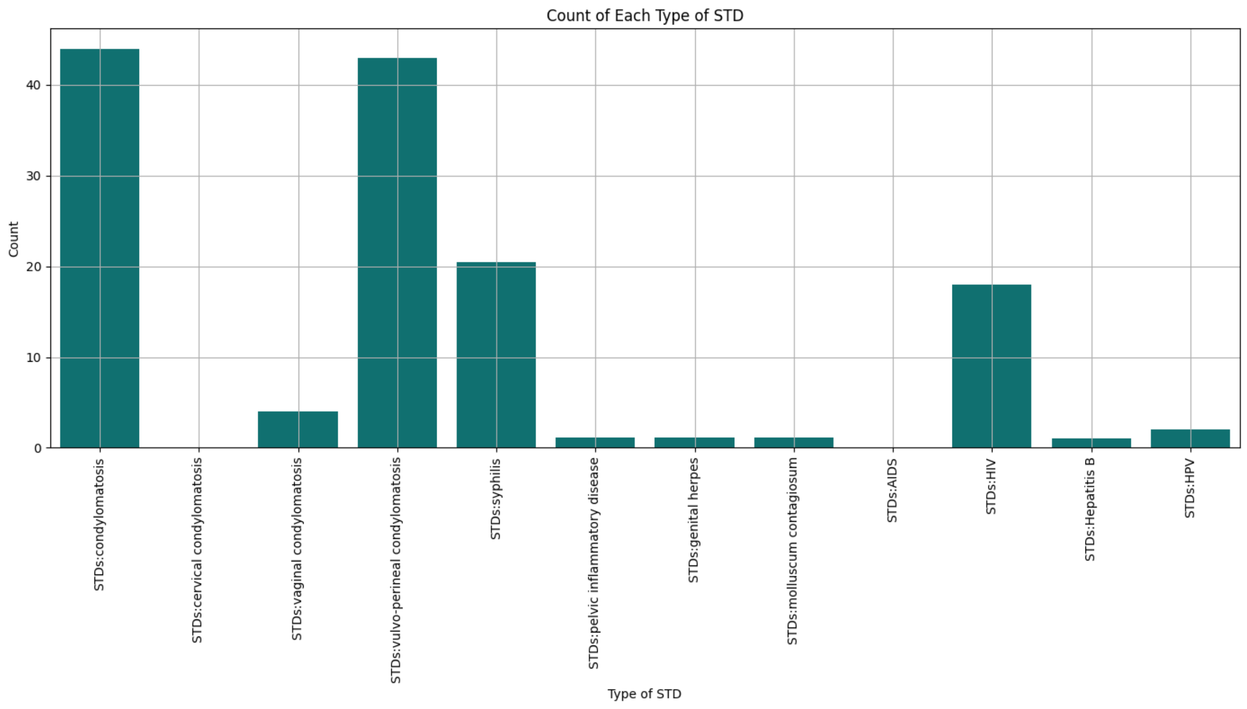
```
plt.figure(figsize=(12, 8))
barplot = sns.barplot(
    data=data,
    x='Smokes',
    y='Smokes (years)',
    color='royalblue',
    errorbar=None
)

plt.title('Distribution of Smoking Years by Smoking Status')
plt.xlabel('Smokes (0: No, 1: Yes)')
plt.ylabel('Number of Smoking Years')
plt.grid(True)
plt.tight_layout()
plt.show()
```



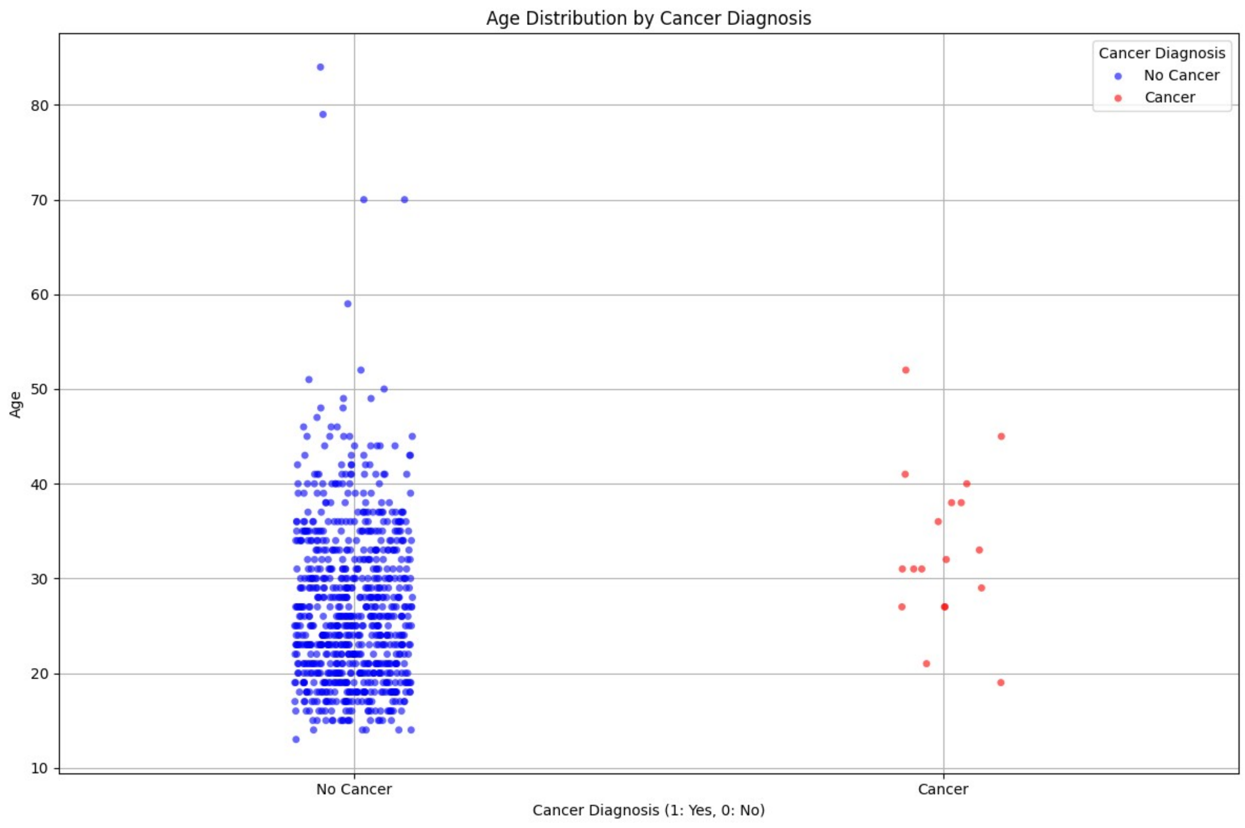
```
plt.figure(figsize=(12, 8))
barplot = sns.barplot(
    data=data,
    x='Hormonal Contraceptives',
    y='Hormonal Contraceptives (years)',
    color='royalblue',  # Use a single color since hue is not
specified
    errorbar=None  # Disable error bars
)

plt.title('Distribution of Hormonal Contraceptives Years by Hormonal
Contraceptives Status')
plt.xlabel('Hormonal Contraceptives (0: No, 1: Yes)')
plt.ylabel('Number ofHormonal Contraceptives Years')
plt.grid(True)
plt.tight_layout()
plt.show()
```



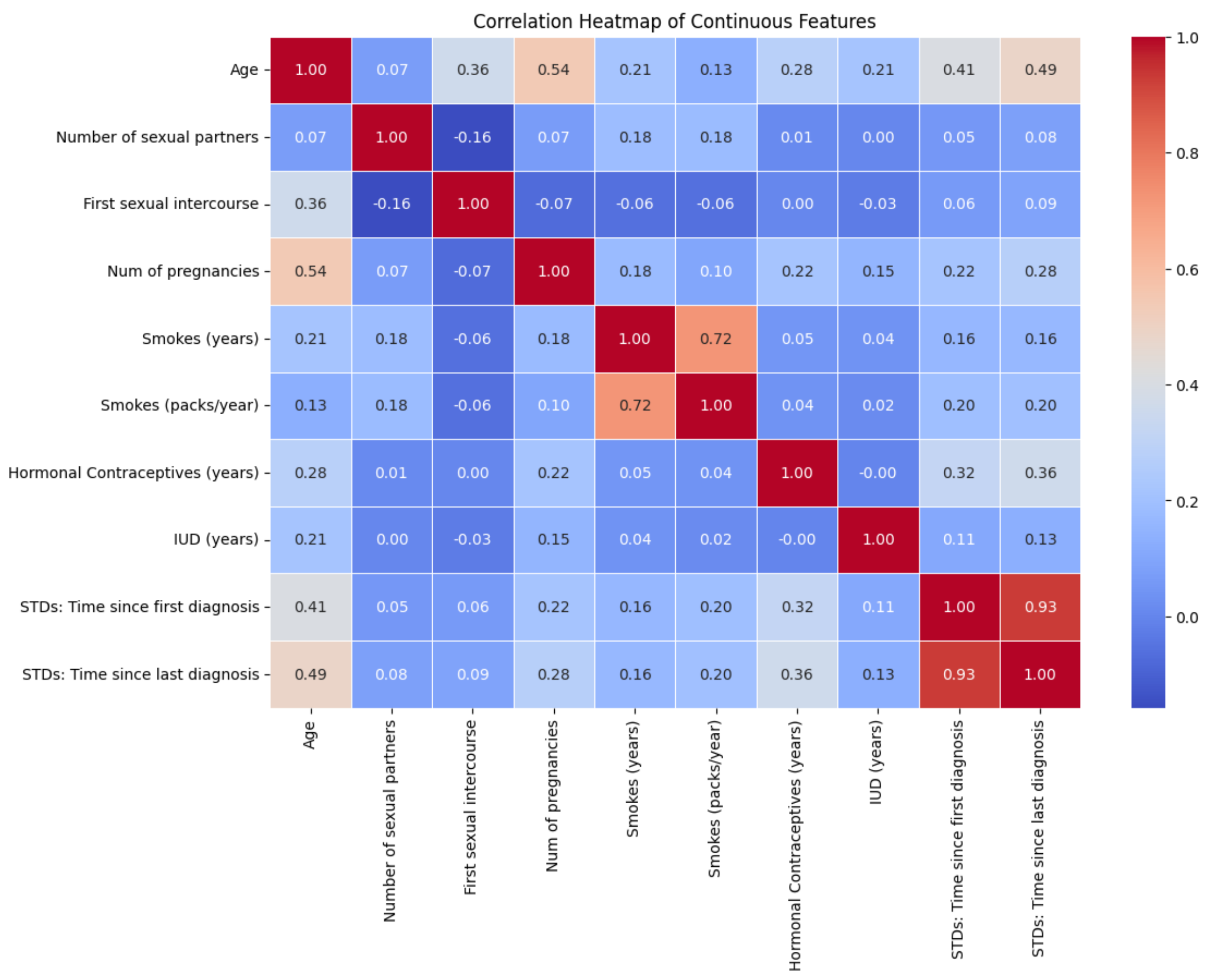
```
std_columns = [
    'STDs:condylomatosis', 'STDs:cervical condylomatosis',
    'STDs:vaginal condylomatosis', 'STDs:vulvo-perineal
condylomatosis',
    'STDs:syphilis', 'STDs:pelvic inflammatory disease',
    'STDs:genital herpes', 'STDs:molluscum contagiosum',
    'STDs:AIDS', 'STDs:HIV', 'STDs:Hepatitis B', 'STDs:HPV'
]

plot_data = data[std_columns].sum().reset_index()
plot_data.columns = ['Type', 'Count']

plt.figure(figsize=(14, 8))

barplot = sns.barplot(
    data=plot_data,
    x='Type',
    y='Count',
    color='teal',
    errorbar=None
)

plt.title('Count of Each Type of STD')
plt.xlabel('Type of STD')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
data['Dx:Cancer'] = data['Dx:Cancer'].apply(lambda x: 1 if x >= 0.5
else 0).astype(int)
plt.figure(figsize=(12, 8))

sns.stripplot(
    data=data,
    x='Dx:Cancer',
    y='Age',
    hue='Dx:Cancer',
    jitter=True,
    alpha=0.6,
    palette={0: 'blue', 1: 'red'}
)

plt.title('Age Distribution by Cancer Diagnosis')
plt.xlabel('Cancer Diagnosis (1: Yes, 0: No)')
plt.ylabel('Age')
plt.xticks(ticks=[0, 1], labels=['No Cancer', 'Cancer'])
plt.legend(title='Cancer Diagnosis', labels=['No Cancer', 'Cancer'])
plt.grid(True)
plt.tight_layout()
plt.show()
```

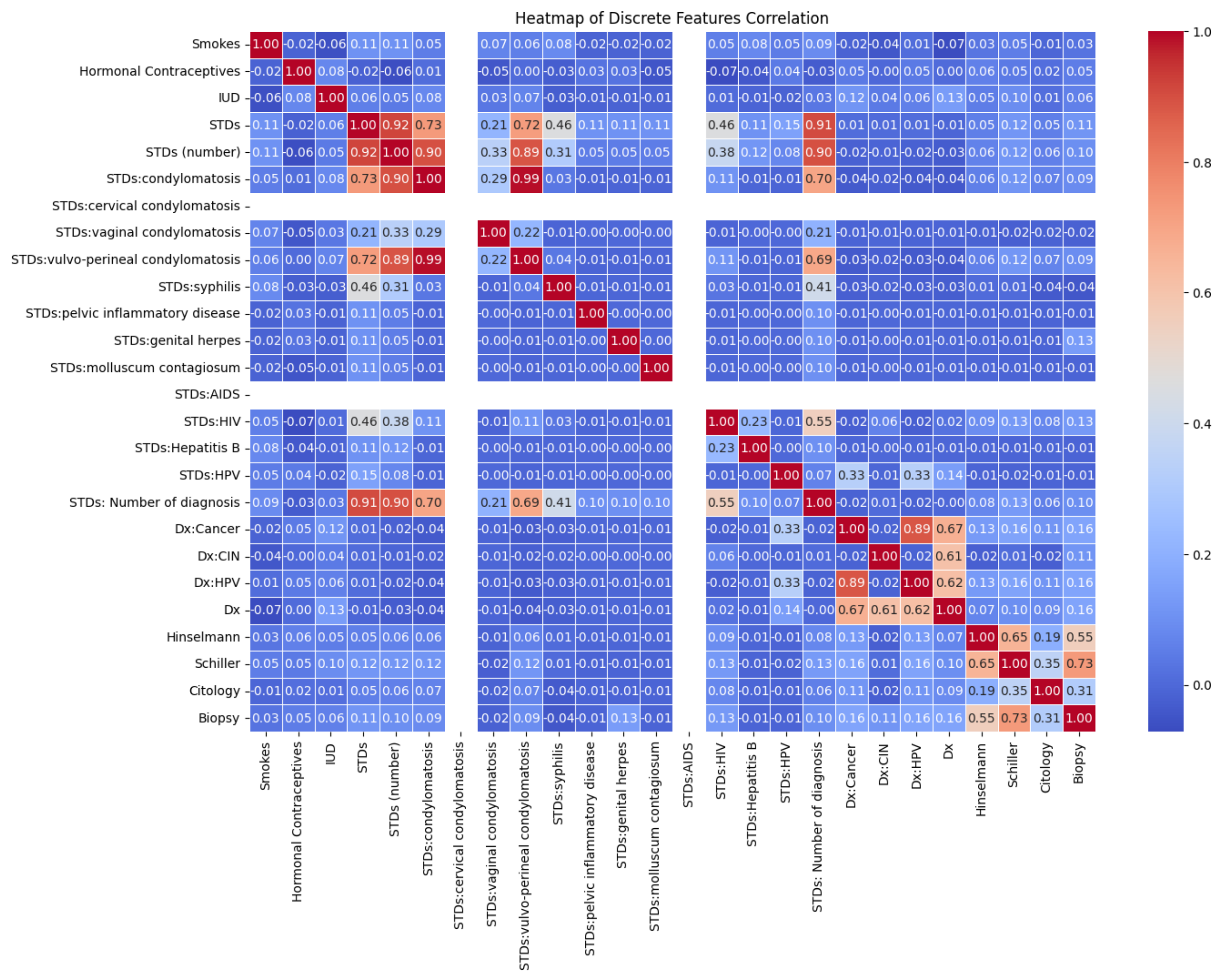


```
continuous_features = [
    'Age',
    'Number of sexual partners',
    'First sexual intercourse',
    'Num of pregnancies',
    'Smokes (years)',
    'Smokes (packs/year)',
    'Hormonal Contraceptives (years)',
    'IUD (years)',
    'STDs: Time since first diagnosis',
    'STDs: Time since last diagnosis'
]

corr_matrix = df[continuous_features].corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title('Correlation Heatmap of Continuous Features')
plt.show()
```



	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives (years)	IUD (years)	STDs: Time since first diagnosis	STDs: Time since last diagnosis
Age	1.00	0.07	0.36	0.54	0.21	0.13	0.28	0.21	0.41	0.49
Number of sexual partners	0.07	1.00	-0.16	0.07	0.18	0.18	0.01	0.00	0.05	0.08
First sexual intercourse	0.36	-0.16	1.00	-0.07	-0.06	-0.06	0.00	-0.03	0.06	0.09
Num of pregnancies	0.54	0.07	-0.07	1.00	0.18	0.10	0.22	0.15	0.22	0.28
Smokes (years)	0.21	0.18	-0.06	0.18	1.00	0.72	0.05	0.04	0.16	0.16
Smokes (packs/year)	0.13	0.18	-0.06	0.10	0.72	1.00	0.04	0.02	0.20	0.20
Hormonal Contraceptives (years)	0.28	0.01	0.00	0.22	0.05	0.04	1.00	-0.00	0.32	0.36
IUD (years)	0.21	0.00	-0.03	0.15	0.04	0.02	-0.00	1.00	0.11	0.13
STDs: Time since first diagnosis	0.41	0.05	0.06	0.22	0.16	0.20	0.32	0.11	1.00	0.93
STDs: Time since last diagnosis	0.49	0.08	0.09	0.28	0.16	0.20	0.36	0.13	0.93	1.00

```
discrete = ['Smokes', 'Hormonal Contraceptives', 'IUD', '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', 'Dx:Cancer', 'Dx:CIN',
'Dx:HPV', 'Dx',
            'Hinselmann', 'Schiller', 'Citology', 'Biopsy']

corr_matrix_discrete = df[discrete].corr()
plt.figure(figsize=(15, 10))
sns.heatmap(corr_matrix_discrete, annot=True, cmap='coolwarm',
fmt='.2f', linewidths=0.5)
plt.title('Heatmap of Discrete Features Correlation')
plt.show()
```



MODEL BUILDING

```
x= data.drop('Dx:Cancer', axis=1)
y= data['Dx:Cancer']

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(x,
y ,test_size=0.2)

X_train

{"type":"dataframe","variable_name":"X_train"}

X_test

{"type":"dataframe","variable_name":"X_test"}

y_train
```

```
218    0
504    0
421    0
38     0
522    0
      ..
338    0
526    0
535    0
420    0
564    0
Name: Dx:Cancer, Length: 668, dtype: int64
```

```
y_test
```

```
452    0
512    0
415    0
591    0
331    0
      ..
497    0
667    0
356    0
285    0
701    0
Name: Dx:Cancer, Length: 167, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
norm = MinMaxScaler().fit(X_train)
X_train_norm = norm.transform(X_train)
```

```
X_train_norm
```

```
array([[0.1754386 , 0.11111111, 0.42105263, ..., 0.        , 0.
,
         0.        ],
       [0.10526316, 0.        , 0.36842105, ..., 0.        , 0.
,
         0.        ],
       [0.05263158, 0.        , 0.26315789, ..., 0.        , 0.
,
         0.        ],
       ...,
       [0.35087719, 0.07407407, 0.26315789, ..., 0.        , 0.
,
         0.        ],
       [0.12280702, 0.        , 0.31578947, ..., 0.        , 0.
,
        0.        ],
       [0.29824561, 0.03703704, 0.42105263, ..., 0.        , 0.
,
        0.        ]])
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
scaler = StandardScaler().fit(X_train)
X_test_norm = scaler.transform(X_test)
```

```
X_test_norm
```

```
array([[-1.2204397 , -0.90432351, -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [ 0.36732035, -0.90432351, -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [-1.09830431, -0.3174778 , -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       ...,
       [-0.36549198, -0.90432351,  1.04683246, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [-0.48762737, -0.3174778 , -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [-0.97616892, -0.90432351, -0.02960923, ..., -0.30835364,
        -0.22430886, -0.25902249]])
```

```
stan = StandardScaler().fit(X_train)
X_train_stan = stan.transform(X_train)
X_test_stan = stan.transform(X_test)
```

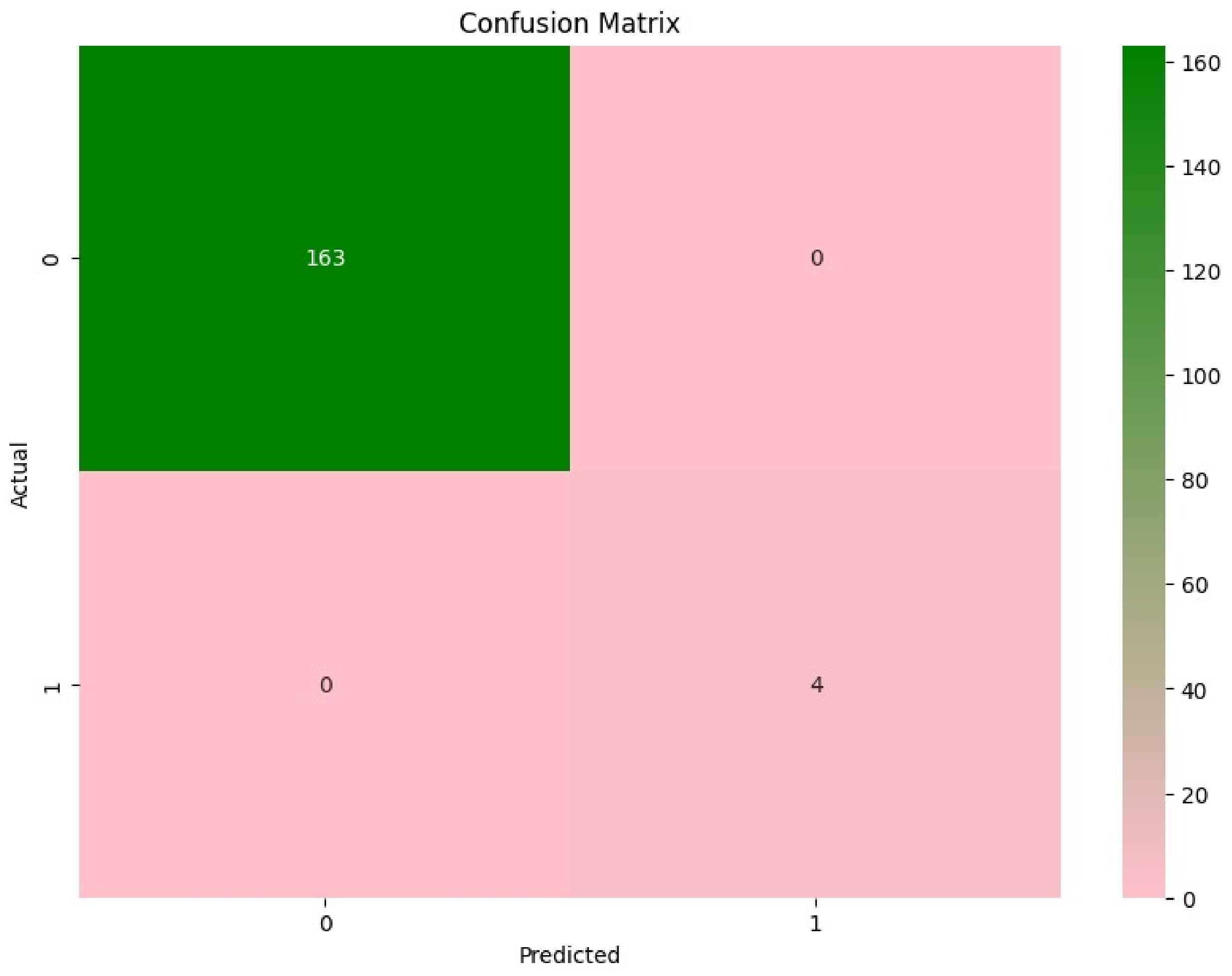
```
X_test_stan
```

```
array([[-1.2204397 , -0.90432351, -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [ 0.36732035, -0.90432351, -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [-1.09830431, -0.3174778 , -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       ...,
       [-0.36549198, -0.90432351,  1.04683246, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [-0.48762737, -0.3174778 , -0.74723702, ..., -0.30835364,
        -0.22430886, -0.25902249],
       [-0.97616892, -0.90432351, -0.02960923, ..., -0.30835364,
        -0.22430886, -0.25902249]])
```

RANDOM FOREST MODEL

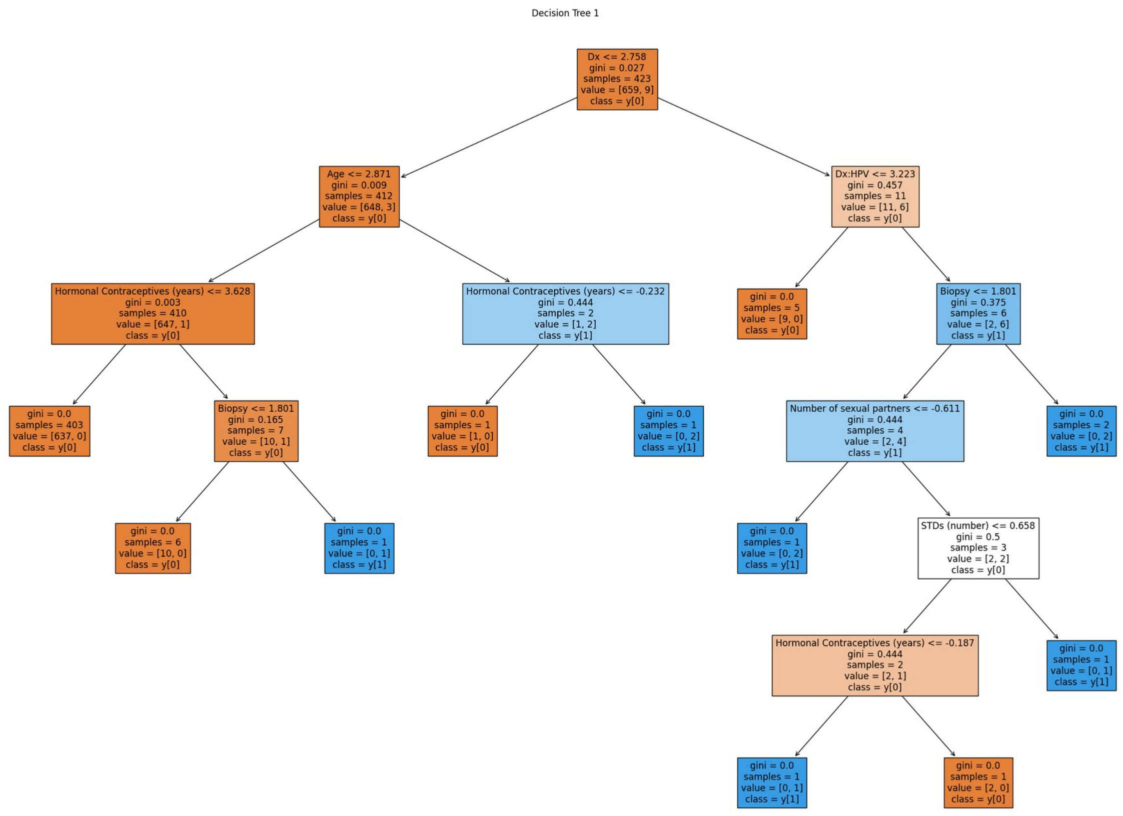
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
precision_score, recall_score, ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV,
train_test_split

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_stan, y_train)

RandomForestClassifier(random_state=42)

y_pred = rf.predict(X_test_stan)

y_pred

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

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

Accuracy: 1.0

from sklearn.metrics import classification_report
print("Classification Report:")
print(classification_report(y_test, y_pred))

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       163
           1       1.00      1.00      1.00         4

    accuracy                           1.00       167
   macro avg       1.00      1.00      1.00       167
```

```
weighted avg       1.00      1.00      1.00       167


from matplotlib.colors import LinearSegmentedColormap
from sklearn.metrics import confusion_matrix
cmap = LinearSegmentedColormap.from_list("pink_green", ["#FFC0CB",
"#008000"])

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap=cmap)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

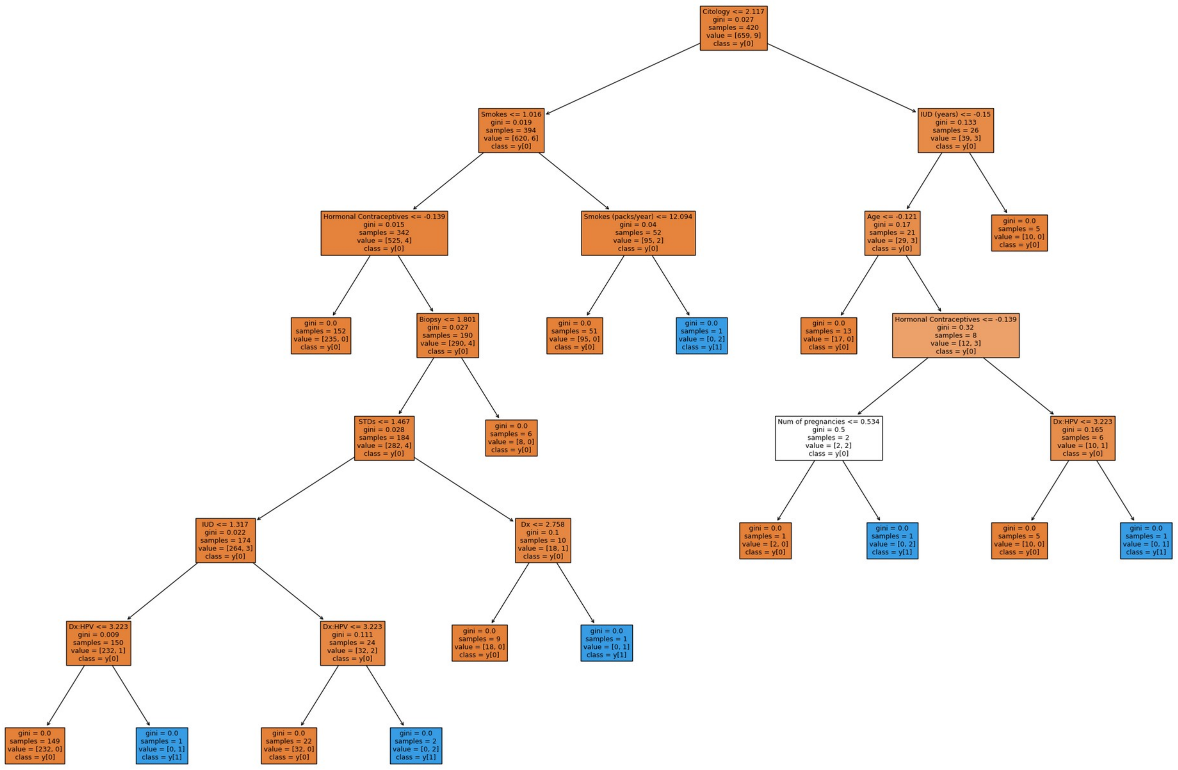


```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

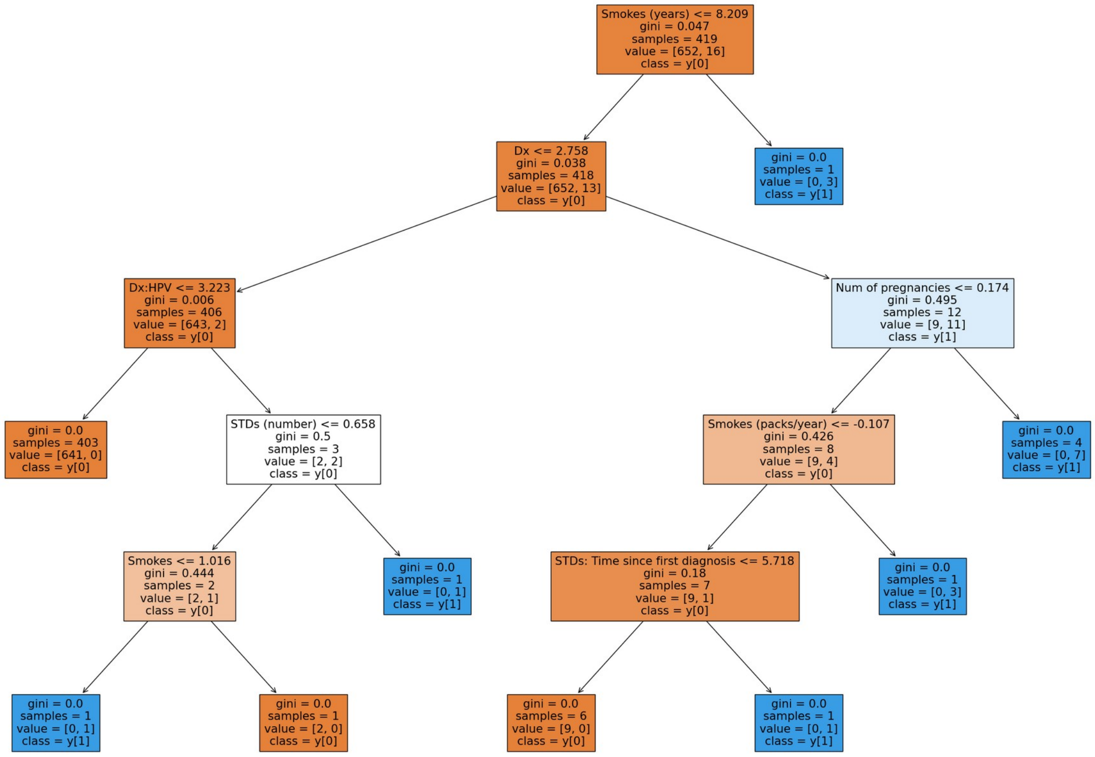
```python
def plot_single_tree(tree, tree_id):
    plt.figure(figsize=(30, 20))
    plot_tree(tree, filled=True, feature_names=X_train.columns,
class_names=True)
    plt.title(f"Decision Tree {tree_id}")
    plt.show()

for i in range(3):
    plot_single_tree(rf.estimators_[i], i + 1)
```


Decision Tree 1
Dx <= 2.758 gini = 0.027 samples = 423 value = [659, 9] class = y[0]
Age <= 2.871 gini = 0.009 samples = 412 value = [648, 3] class = y[0]
Dx:HPV <= 3.223 gini = 0.457 samples = 11 value = [11, 6] class = y[0]
Hormonal Contraceptives (years) <= 3.628 gini = 0.003 samples = 410 value = [647, 1] class = y[0]
Hormonal Contraceptives (years) <= -0.232 gini = 0.444 samples = 2 value = [1, 2] class = y[1]
gini = 0.0 samples = 5 value = [9, 0] class = y[0]
Biopsy <= 1.801 gini = 0.375 samples = 6 value = [2, 6] class = y[1]
gini = 0.0 samples = 403 value = [637, 0] class = y[0]
Biopsy <= 1.801 gini = 0.165 samples = 7 value = [10, 1] class = y[0]
gini = 0.0 samples = 1 value = [1, 0] class = y[0]
gini = 0.0 samples = 1 value = [0, 2] class = y[1]
Number of sexual partners <= -0.611 gini = 0.444 samples = 4 value = [2, 4] class = y[1]
gini = 0.0 samples = 2 value = [0, 2] class = y[1]
gini = 0.0 samples = 6 value = [10, 0] class = y[0]
gini = 0.0 samples = 1 value = [0, 1] class = y[1]
gini = 0.0 samples = 1 value = [0, 2] class = y[1]
STDs (number) <= 0.658 gini = 0.5 samples = 3 value = [2, 2] class = y[0]
Hormonal Contraceptives (years) <= -0.187 gini = 0.444 samples = 2 value = [2, 1] class = y[0]
gini = 0.0 samples = 1 value = [0, 1] class = y[1]
gini = 0.0 samples = 1 value = [0, 1] class = y[1]
gini = 0.0 samples = 1 value = [2, 0] class = y[0]

Decision Tree 2



Decision Tree 3



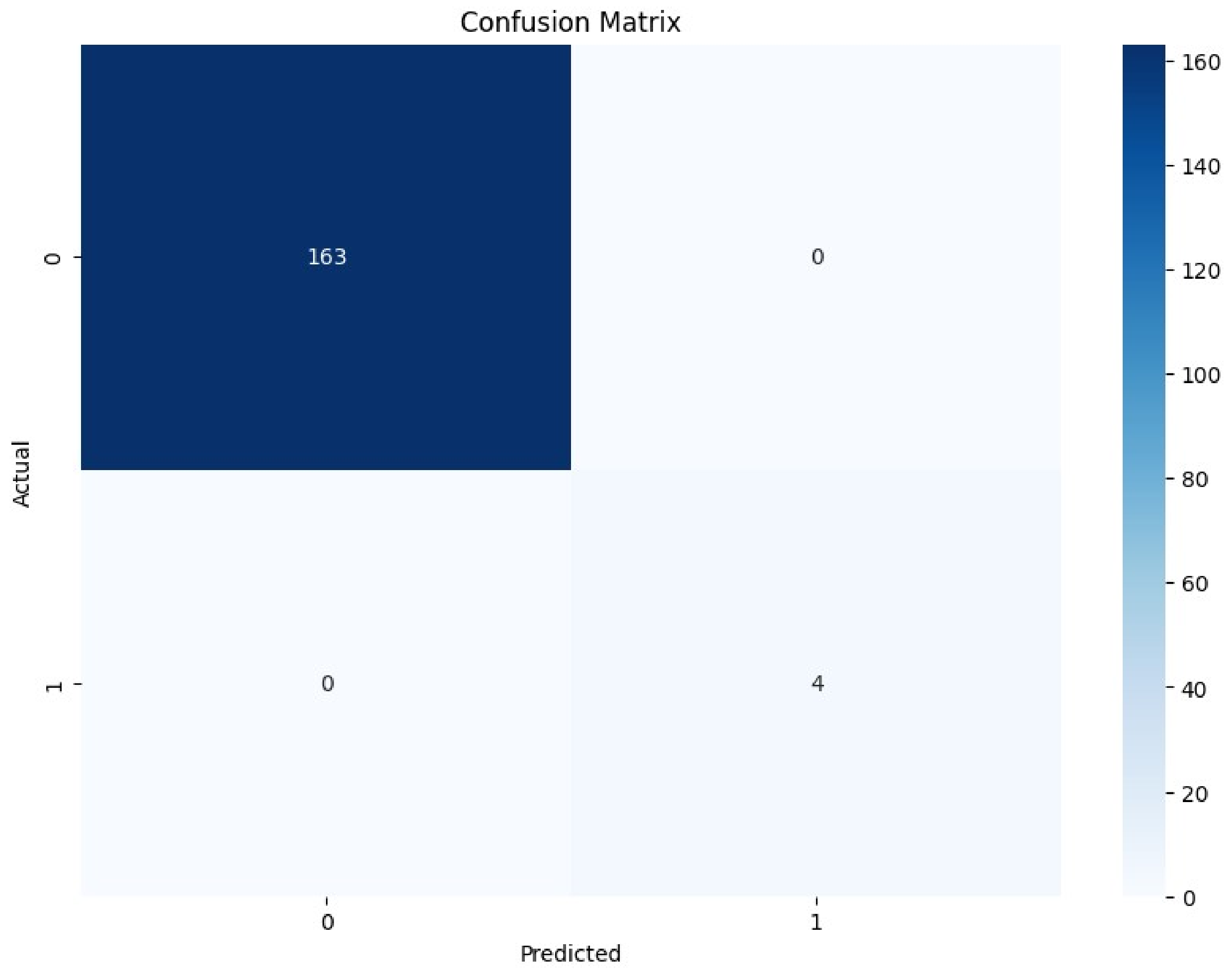
```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')

conf_matrix = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf_matrix.ravel()
specificity = tn / (tn + fp)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
print(f'Specificity: {specificity}')

Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Specificity: 1.0
```

SUPPORT VECTOR MACHINE

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

SVM_model = SVC(kernel='linear', random_state=42)
SVM_model.fit(X_train_stan, y_train)

SVC(kernel='linear', random_state=42)

SVM_y_pred = SVM_model.predict(X_test_stan)

svm_acuuracy= accuracy_score(y_test, SVM_y_pred)
print(f"Accuracy: {svm_acuuracy}")

Accuracy: 0.9820359281437125

print(f'Classification Report:\n{classification_report(y_test,
SVM_y_pred)}')

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       163
           1       0.60      0.75      0.67         4

    accuracy                           0.98       167
```

```
   macro avg       0.80      0.87      0.83       167
weighted avg       0.98      0.98      0.98       167
```

```
confusion_matrix(y_test, SVM_y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



```
accuracy = accuracy_score(y_test,SVM_y_pred)
precision = precision_score(y_test, SVM_y_pred, average='macro')
recall = recall_score(y_test, SVM_y_pred, average='macro')
f1 = f1_score(y_test,SVM_y_pred, average='macro')

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
Accuracy: 0.9820359281437125
Precision: 0.7969135802469136
Recall: 0.8688650306748467
F1 Score: 0.8287179487179486
```

LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
LR_model = LinearRegression()
LR_model.fit(X_train_stan, y_train)
```

```
LinearRegression()
```

```
LR_Y_pred = LR_model.predict(X_test_stan)
```

```
print("Mean Squared Error:", mean_squared_error(y_test, LR_Y_pred))
print("R-squared:", r2_score(y_test, LR_Y_pred))
```

```
Mean Squared Error: 0.012662802197635
R-squared: 0.45835446243889166
```

ADABOOST

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier
```

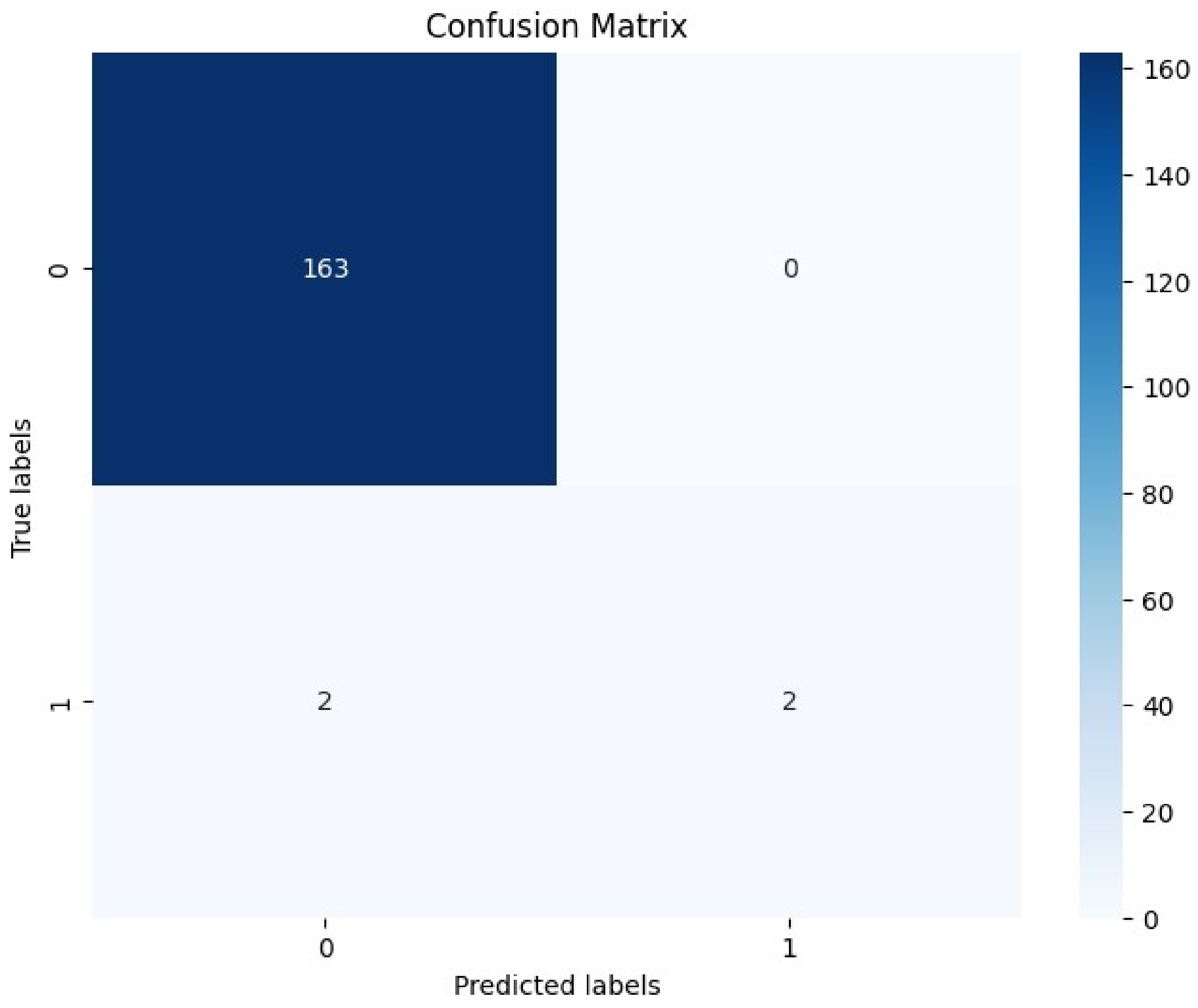
```
from sklearn.tree import DecisionTreeClassifier
base_estimator = DecisionTreeClassifier(max_depth=1)
AB_classifier= AdaBoostClassifier(base_estimator=base_estimator,
n_estimators=50, random_state=42)
AB_classifier.fit(X_train_stan, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:156:
FutureWarning: `base_estimator` was renamed to `estimator` in version
1.2 and will be removed in 1.4.
  warnings.warn(
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1),
                   random_state=42)
```

```
AB_y_pred = AB_classifier.predict(X_test_stan)
```

```
AB_Confusion= confusion_matrix(y_test, AB_y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(AB_Confusion, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```



```
print(f'Classification Report:\n{classification_report(y_test,
AB_y_pred)}')

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       163
           1       1.00      0.50      0.67         4
```

```
    accuracy                           0.99       167
   macro avg       0.99      0.75      0.83       167
weighted avg       0.99      0.99      0.99       167


AB_acuuracy= accuracy_score(y_test, AB_y_pred)
print(f"Accuracy: {AB_acuuracy}")

Accuracy: 0.9880239520958084
```

CATBOOST MODEL

```
pip install catboost

Requirement already satisfied: catboost in
/usr/local/lib/python3.10/dist-packages (1.2.5)
Requirement already satisfied: graphviz in
/usr/local/lib/python3.10/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in
/usr/local/lib/python3.10/dist-packages (from catboost) (1.26.4)
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.10/dist-packages (from catboost) (2.1.4)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages (from catboost) (1.13.1)
Requirement already satisfied: plotly in
/usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost)
(2024.1)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost)
(2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
(1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
```

```
(4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
(24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
(9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost)
(3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly->catboost)
(9.0.0)
```

```
from catboost import CatBoostClassifier

CatBoostClassifier= CatBoostClassifier (iterations=1000,
learning_rate=0.1, depth=6, random_seed=42, verbose=100)

catboost_model = CatBoostClassifier.fit(X_train_stan, y_train)
```

```
0:	learn: 0.4333289	total: 2.06ms	remaining: 2.06s
100:	learn: 0.0009961	total: 184ms	remaining: 1.64s
200:	learn: 0.0004895	total: 338ms	remaining: 1.34s
300:	learn: 0.0003346	total: 493ms	remaining: 1.14s
400:	learn: 0.0002513	total: 713ms	remaining: 1.06s
500:	learn: 0.0001968	total: 944ms	remaining: 940ms
600:	learn: 0.0001571	total: 1.12s	remaining: 742ms
700:	learn: 0.0001327	total: 1.38s	remaining: 589ms
800:	learn: 0.0001155	total: 1.81s	remaining: 449ms
900:	learn: 0.0001024	total: 2.12s	remaining: 233ms
999:	learn: 0.0000924	total: 2.38s	remaining: 0us
```

```
CB_y_pred = catboost_model.predict(X_test_stan)

catboost_accuracy = accuracy_score(y_test, CB_y_pred)
print(f"Accuracy: {catboost_accuracy}")
```

```
Accuracy: 1.0
```

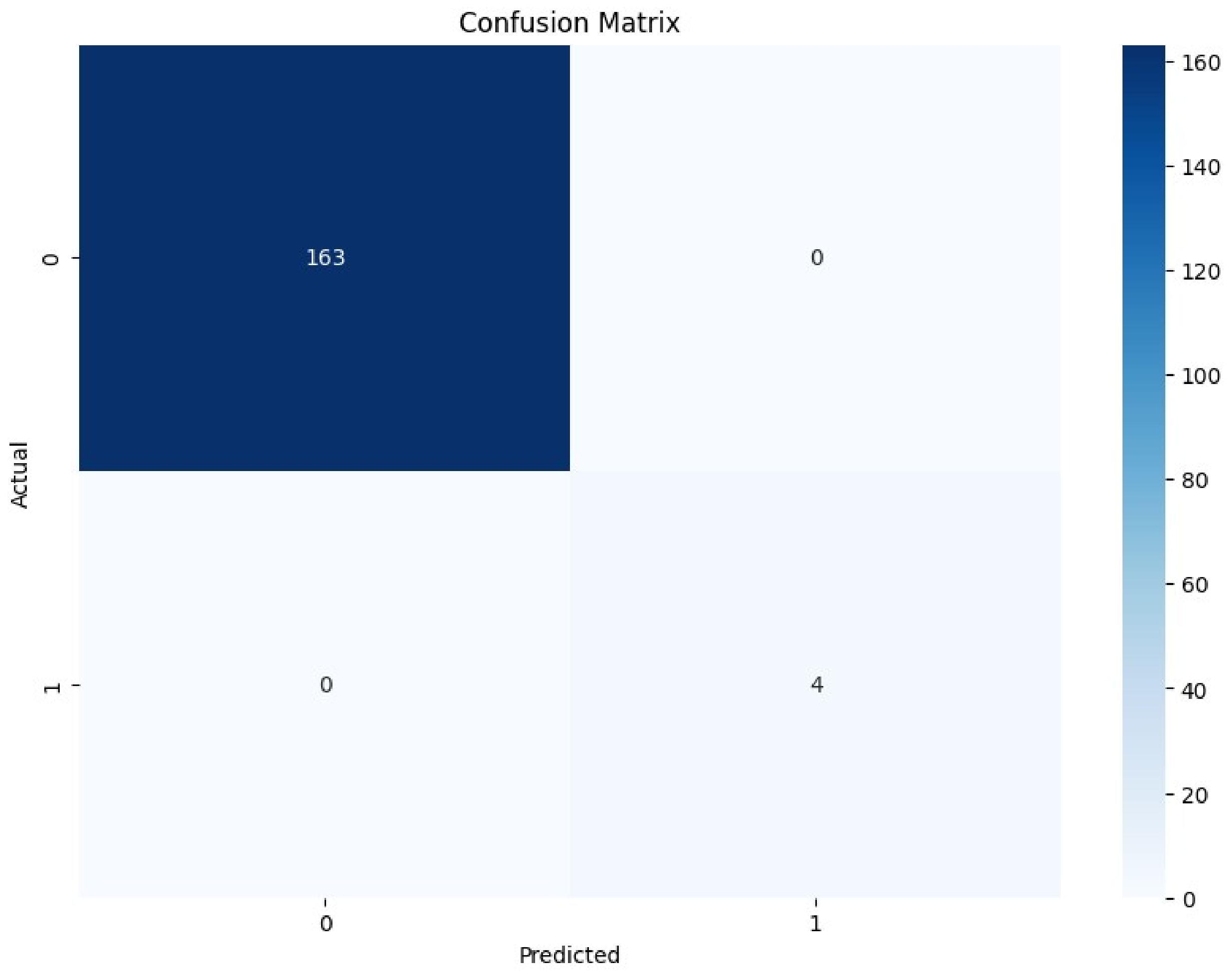
```
catboost_confusion = confusion_matrix(y_test,CB_y_pred)
print(catboost_confusion)
```

```
[[163   0]
 [  0   4]]
```

```
print(f'Classification Report:\n{classification_report(y_test,
CB_y_pred)}')
```

```
conf_matrix = confusion_matrix(y_test,CB_y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       163
           1       1.00      1.00      1.00         4

    accuracy                           1.00       167
   macro avg       1.00      1.00      1.00       167
weighted avg       1.00      1.00      1.00       167
```



```python
accuracy = accuracy_score(y_test, CB_y_pred)
precision = precision_score(y_test, CB_y_pred, average='macro')
recall = recall_score(y_test, CB_y_pred, average='macro')
f1 = f1_score(y_test, CB_y_pred, average='macro')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```