Name: Garcia, John Carlos M.

Section: CPE22S3

Date Performed: 04/28/2024

Date Submitted: 04/28/2024

Logistic Regression Analysis

Task: Predict Cervic Cancer

Setup

```
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 MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

%matplotlib inline
```

```
pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
cervical_cancer_risk_factors = fetch_ucirepo(id=383)

# data (as pandas dataframes)
X = cervical_cancer_risk_factors.data.features
y = cervical_cancer_risk_factors.data.targets

# metadata
print(cervical_cancer_risk_factors.metadata)

# variable information
print(cervical_cancer_risk_factors.variables)

```

```
{'uci_id': 383, 'name': 'Cervical Cancer (Risk Factors)', 'repository_url': 'https://archive.ics.uci.edu/dataset/383/cervical+cancer+
                                  name     role        type demographic  \
0                                  Age  Feature     Integer         Age   
1            Number of sexual partners  Feature  Continuous       Other   
2             First sexual intercourse  Feature  Continuous        None   
3                   Num of pregnancies  Feature  Continuous        None   
4                               Smokes  Feature  Continuous        None   
5                       Smokes (years)  Feature  Continuous        None   
6                  Smokes (packs/year)  Feature  Continuous        None   
7               Hormonal Contraceptives  Feature  Continuous        None   
8       Hormonal Contraceptives (years)  Feature  Continuous        None   
9                                  IUD  Feature  Continuous        None   
10                         IUD (years)  Feature  Continuous        None   
11                                STDs  Feature  Continuous        None   
12                       STDs (number)  Feature  Continuous        None   
13                STDs:condylomatosis  Feature  Continuous        None   
14       STDs:cervical condylomatosis  Feature  Continuous        None   
15        STDs:vaginal condylomatosis  Feature  Continuous        None   
16  STDs:vulvo-perineal condylomatosis  Feature  Continuous        None   
```

```
17                         STDs:syphilis  Feature  Continuous        None
18   STDs:pelvic inflammatory disease  Feature  Continuous        None
19                  STDs:genital herpes  Feature  Continuous        None
20          STDs:molluscum contagiosum  Feature  Continuous        None
21                            STDs:AIDS  Feature  Continuous        None
22                             STDs:HIV  Feature  Continuous        None
23                     STDs:Hepatitis B  Feature  Continuous        None
24                             STDs:HPV  Feature  Continuous        None
25          STDs: Number of diagnosis  Feature     Integer        None
26   STDs: Time since first diagnosis  Feature  Continuous        None
27    STDs: Time since last diagnosis  Feature  Continuous        None
28                            Dx:Cancer  Feature     Integer        None
29                               Dx:CIN  Feature     Integer        None
30                               Dx:HPV  Feature     Integer        None
31                                   Dx  Feature     Integer        None
32                           Hinselmann  Feature     Integer        None
33                             Schiller  Feature     Integer        None
34                             Citology  Feature     Integer        None
35                               Biopsy  Feature     Integer        None

   description units missing_values
0         None  None             no
1         None  None            yes
2         None  None            yes
3         None  None            yes
4         None  None            yes
5         None  None            yes
6         None  None            yes
7         None  None            yes
8         None  None            yes
9         None  None            yes
10        None  None            yes
11        None  None            yes
12        None  None            yes
13        None  None            yes
14        None  None            yes
15        None  None            yes
16        None  None            yes
```

Data Frame

```python
logistic_df = pd.concat([X, y], axis=1) #Combine both dataframes into one for more efficient manipulation of data
```

Exploration

```python
logistic_df
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormo Contracepti
0	18	4.0	15.0	1.0	0.0	0.0	0.0	
1	15	1.0	14.0	1.0	0.0	0.0	0.0	
2	34	1.0	NaN	1.0	0.0	0.0	0.0	
3	52	5.0	16.0	4.0	1.0	37.0	37.0	
4	46	3.0	21.0	4.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
853	34	3.0	18.0	0.0	0.0	0.0	0.0	
854	32	2.0	19.0	1.0	0.0	0.0	0.0	
855	25	2.0	17.0	0.0	0.0	0.0	0.0	
856	33	2.0	24.0	2.0	0.0	0.0	0.0	
857	29	2.0	20.0	1.0	0.0	0.0	0.0	

```python
print("Head:\n",logistic_df.head(), "\n\n")

print("DTypes:\n",logistic_df.dtypes, "\n\n")
print("Description:\n",logistic_df.describe())
```

```
Head:
    Age  Number of sexual partners  First sexual intercourse  \
0   18                        4.0                      15.0
1   15                        1.0                      14.0
2   34                        1.0                       NaN
3   52                        5.0                      16.0
4   46                        3.0                      21.0

   Num of pregnancies  Smokes  Smokes (years)  Smokes (packs/year)  \
0                 1.0     0.0             0.0                  0.0
1                 1.0     0.0             0.0                  0.0
2                 1.0     0.0             0.0                  0.0
3                 4.0     1.0            37.0                 37.0
4                 4.0     0.0             0.0                  0.0

   Hormonal Contraceptives  Hormonal Contraceptives (years)  IUD  ...  \
0                      0.0                              0.0  0.0  ...
1                      0.0                              0.0  0.0  ...
2                      0.0                              0.0  0.0  ...
3                      1.0                              3.0  0.0  ...
4                      1.0                             15.0  0.0  ...

   STDs: Time since first diagnosis  STDs: Time since last diagnosis  \
0                               NaN                              NaN
1                               NaN                              NaN
2                               NaN                              NaN
3                               NaN                              NaN
4                               NaN                              NaN

   Dx:Cancer  Dx:CIN  Dx:HPV  Dx  Hinselmann  Schiller  Citology  Biopsy
0          0       0       0   0           0         0         0       0
1          0       0       0   0           0         0         0       0
2          0       0       0   0           0         0         0       0
3          1       0       1   0           0         0         0       0
4          0       0       0   0           0         0         0       0

[5 rows x 36 columns]


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
```

Missing Values

```
print("Nulls:\n",logistic_df.isnull().sum())
```

```
Nulls:
 Age                                   0
Number of sexual partners            26
First sexual intercourse              7
Num of pregnancies                   56
Smokes                               13
Smokes (years)                       13
Smokes (packs/year)                  13
Hormonal Contraceptives             108
Hormonal Contraceptives (years)     108
IUD                                 117
IUD (years)                         117
STDs                                105
STDs (number)                       105
STDs:condylomatosis                 105
STDs:cervical condylomatosis        105
```

```
STDs:vaginal condylomatosis           105
STDs:vulvo-perineal condylomatosis    105
STDs:syphilis                         105
STDs:pelvic inflammatory disease      105
STDs:genital herpes                   105
STDs:molluscum contagiosum            105
STDs:AIDS                             105
STDs:HIV                              105
STDs:Hepatitis B                      105
STDs:HPV                              105
STDs: Number of diagnosis               0
STDs: Time since first diagnosis      787
STDs: Time since last diagnosis       787
Dx:Cancer                               0
Dx:CIN                                  0
Dx:HPV                                  0
Dx                                      0
Hinselmann                              0
Schiller                                0
Citology                                0
Biopsy                                  0
dtype: int64
```

```
print("Unique values in 'IUD' column:")
print(logistic_df['IUD'].unique())

print("\n\nUnique values in 'STDs:HIV ' column:")
print(logistic_df['STDs:HIV'].unique())
```

```
Unique values in 'IUD' column:
[ 0.  1. nan]


Unique values in 'STDs:HIV ' column:
[ 0.  1. nan]
```

Duplicates

```
duplicates = logistic_df.duplicated()
print("Duplicates:\n\n", duplicates, "\n\n")
print("Duplicate Rows:\n\n",logistic_df[duplicates])
```

```
Duplicates:

 0      False
1      False
2      False
3      False
4      False
       ...
853    False
854    False
855    False
856    False
857    False
Length: 858, dtype: bool


Duplicate Rows:

      Age  Number of sexual partners  First sexual intercourse  \
66    34                        3.0                      19.0
234   25                        NaN                      18.0
255   25                        2.0                      18.0
356   18                        1.0                      17.0
395   18                        1.0                      18.0
406   17                        1.0                      17.0
419   19                        4.0                      14.0
431   18                        1.0                      14.0
435   17                        2.0                      15.0
440   15                        1.0                      14.0
442   16                        1.0                      15.0
453   15                        1.0                      15.0
454   15                        1.0                      14.0
466   15                        1.0                      14.0
486   28                        2.0                      20.0
525   17                        1.0                      16.0
530   21                        4.0                      15.0
536   16                        1.0                      14.0
575   17                        2.0                      15.0
```

```
580   17                       2.0                     15.0
638   21                       1.0                     20.0
715   15                       2.0                     14.0
785   28                       1.0                     19.0

     Num of pregnancies  Smokes  Smokes (years)  Smokes (packs/year)  \
66                  3.0     0.0             0.0                  0.0
234                 2.0     0.0             0.0                  0.0
255                 2.0     0.0             0.0                  0.0
356                 1.0     0.0             0.0                  0.0
395                 1.0     0.0             0.0                  0.0
406                 1.0     0.0             0.0                  0.0
419                 1.0     0.0             0.0                  0.0
431                 2.0     0.0             0.0                  0.0
435                 1.0     0.0             0.0                  0.0
440                 1.0     0.0             0.0                  0.0
442                 1.0     0.0             0.0                  0.0
453                 1.0     0.0             0.0                  0.0
454                 1.0     0.0             0.0                  0.0
466                 1.0     0.0             0.0                  0.0
```

Steps:

Cleaning Missing Values

Since 'STDs: Time since first diagnosis' and 'STDs: Time since last diagnosis' have a huge amount of missing values, accounting to about 92%, we will just drop these 2 columns to remain the integrity of the data

```
logistic_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')
```

```
drop_columns = ['STDs: Time since first diagnosis', 'STDs: Time since last diagnosis']
logistic_df.drop(drop_columns, axis=1, inplace=True)
```

```
logistic_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',
       'Dx:Cancer', 'Dx:CIN', 'Dx:HPV', 'Dx', 'Hinselmann', 'Schiller',
       'Citology', 'Biopsy'],
      dtype='object')
```

```
no_missing_df = logistic_df.dropna()
```

```
no_missing_df
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STDs:HPV	STDs Number o diagnosi
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.00	0.0	...	0.0	
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.00	0.0	...	0.0	
5	42	3.0	23.0	2.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.00	0.0	...	0.0	
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0	
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0	
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.50	0.0	...	0.0	

Cleaning Duplicates

```
new_df = no_missing_df.drop_duplicates()
```

```
new_df
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STDs:HPV	STDs Number o diagnosi
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.00	0.0	...	0.0	
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.00	0.0	...	0.0	
5	42	3.0	23.0	2.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0	
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.00	0.0	...	0.0	
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0	
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0	
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.50	0.0	...	0.0	

Checking:

```
print("Nulls:\n",new_df.isnull().sum())
```

```
Nulls:
 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
Dx:Cancer                             0
Dx:CIN                                0
Dx:HPV                                0
Dx                                    0
Hinselmann                            0
Schiller                              0
Citology                              0
Biopsy                                0
dtype: int64
```

```
print("Duplicates:\n\n", duplicates, "\n\n")
print("Duplicate Rows:\n\n",new_df[duplicates])
```

```
Duplicates:

 0      False
1      False
2      False
3      False
4      False
       ...
853    False
854    False
855    False
856    False
857    False
Length: 858, dtype: bool


Duplicate Rows:

 Empty DataFrame
Columns: [Age, Number of sexual partners, First sexual intercourse, Num of pregnancies, Smokes, Smokes (years), Smokes (packs/year), Hor
Index: []

[0 rows x 34 columns]
<ipython-input-119-0d93af64aaf1>:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  print("Duplicate Rows:\n\n",new_df[duplicates])
```

```
original_data = len(logistic_df)
new_data = len(new_df)
rows_dropped = original_data - new_data
print("Number of rows dropped: ", rows_dropped)
```

```
Number of rows dropped:  208
```

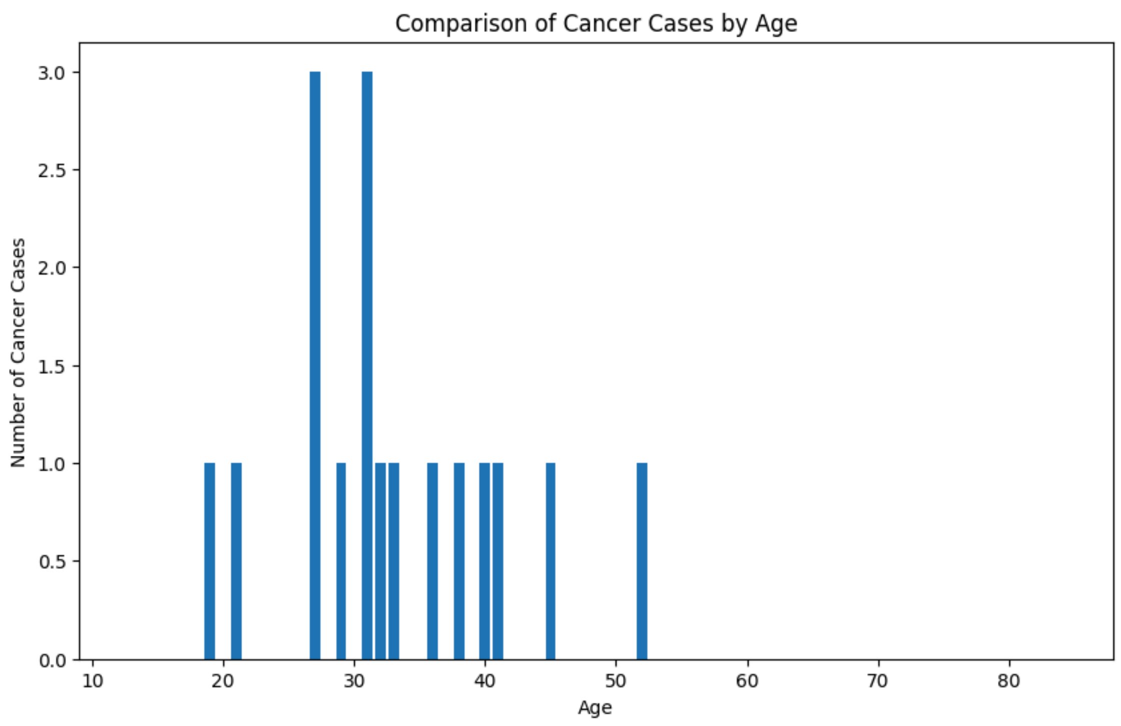
Great! We have removed all the missing and duplicated values! Let's proceed to EDA

Task: Find the relationship between Age and Cervical Cancer

```
print("Unique values in 'Age' column:")
print(new_df['Age'].unique())
```

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

```
filtered_new_df = new_df[['Age', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Age')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Age'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Age')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by Age')
plt.show()
```



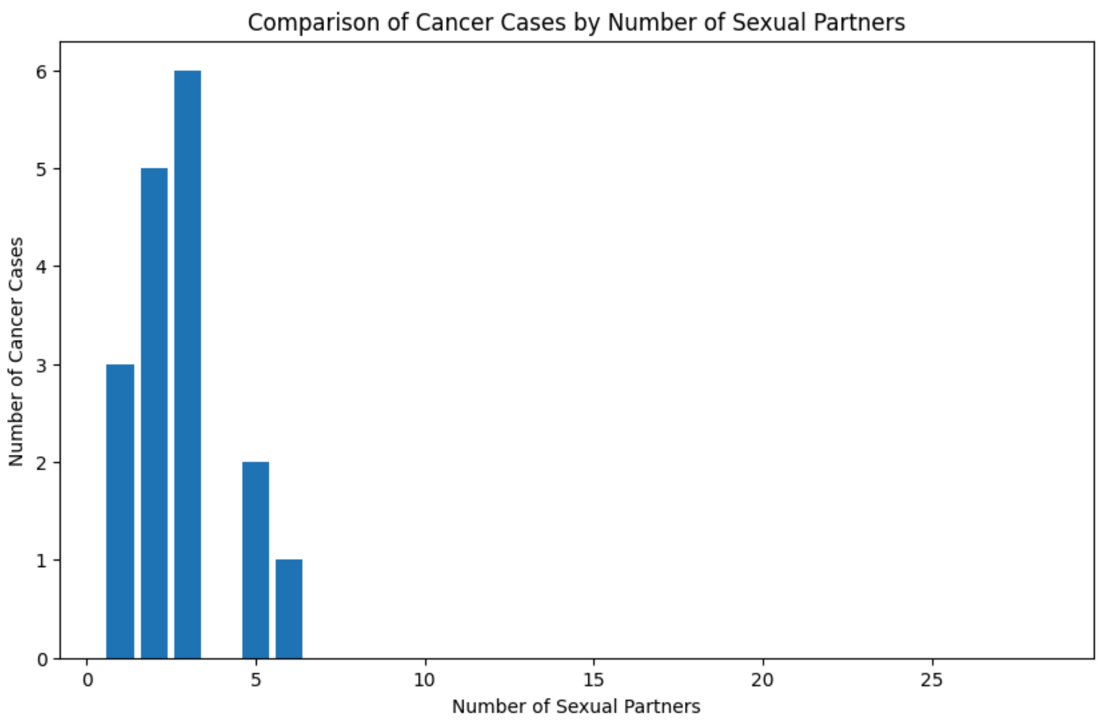
We can see that the number of people with cervic cancer are only present in 20s - 50s

Task: Find the relationship between people that has a high number of sexual partners to cervical cancer

```
print("Unique values in 'Number of sexual partners' column:")
print(new_df['Number of sexual partners'].unique())
```

```
Unique values in 'Number of sexual partners' column:
[ 4.  1.  5.  3.  2.  6.  8.  7. 28.]
```

```
filtered_new_df = new_df[['Number of sexual partners', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Number of sexual partners')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Number of sexual partners'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Number of Sexual Partners')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by Number of Sexual Partners')
plt.show()
```



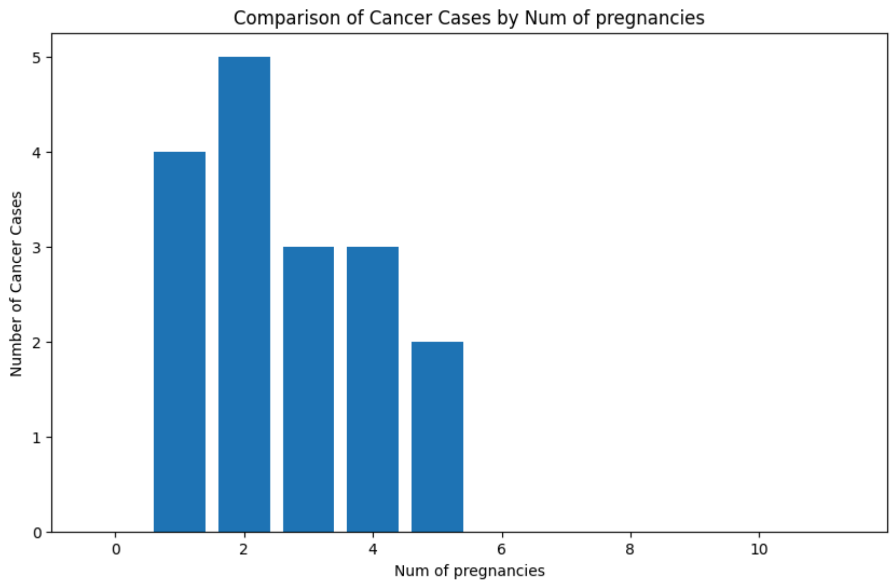
Conclusion, it seems that a high number of sexual partners do not correlate with having a cervic cancer, as there are not even a single cancer case for those with more than 7 up to 28 sexual partners

Task: Find the relationship between pregnancies to cervical cancer

```
print("Unique values in 'Num of pregnancies' column:")
print(new_df['Num of pregnancies'].unique())
```

```
Unique values in 'Num of pregnancies' column:
[ 1.  4.  2.  6.  3.  5.  8.  7.  0. 11. 10.]
```

```
filtered_new_df = new_df[['Num of pregnancies', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Num of pregnancies')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Num of pregnancies'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Num of pregnancies')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by Num of pregnancies')
plt.show()
```



```
correlation_coefficient = new_df['Num of pregnancies'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and Num of pregnancies:", correlation_coefficient)
```

```
Correlation between cervix cancer and Num of pregnancies: 0.03283919105903373
```

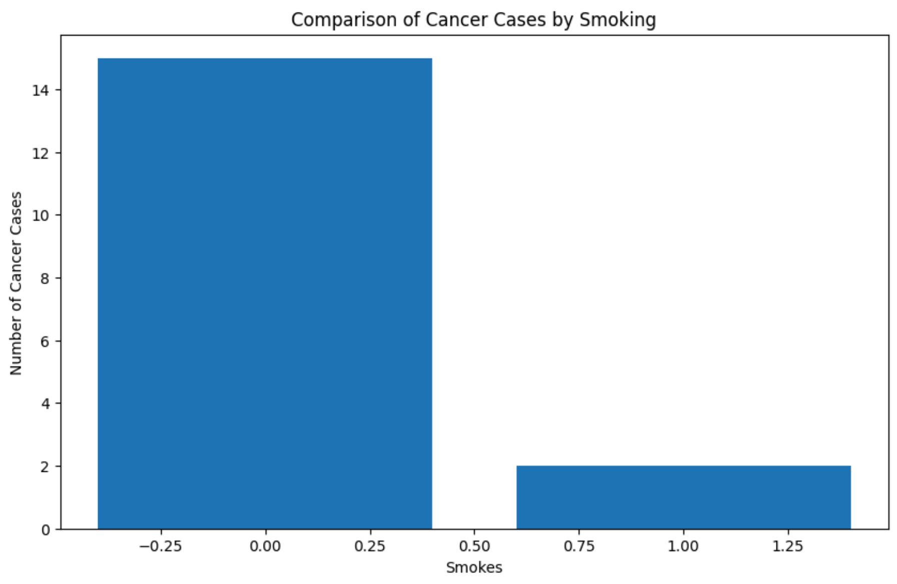
Same with the number of sexual partners, a higher amount of pregnancies does not seem to correlate to higher cases of people with cervic cancer

Task: Find the relationship between smoking to cervical cancer

```
print("Unique values in 'Smokes' column:")
print(new_df['Smokes'].unique())
```

```
Unique values in 'Smokes' column:
[0. 1.]
```

```
filtered_new_df = new_df[['Smokes', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Smokes')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Smokes'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Smokes')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by Smoking')
plt.show()
```



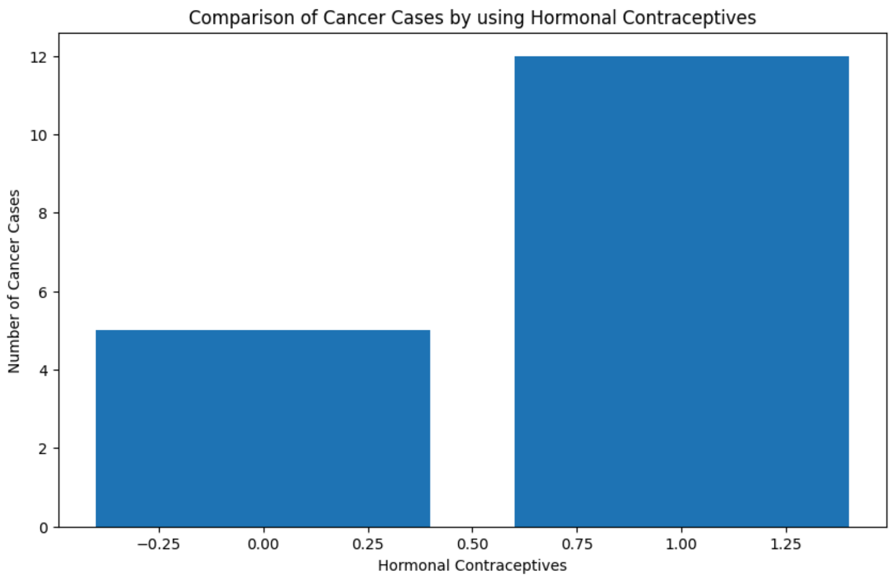
From the plot above, we can see that smoking does not affect whether the person has cervic cancer, as there are more cancer cases in those that do not smoke

Task: Find the relationship between using Hormonal Contraceptives to cervical cancer

```
print("Unique values in 'Hormonal Contraceptives' column:")
print(new_df['Hormonal Contraceptives'].unique())
```

```
Unique values in 'Hormonal Contraceptives' column:
[0. 1.]
```

```
filtered_new_df = new_df[['Hormonal Contraceptives', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Hormonal Contraceptives')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Hormonal Contraceptives'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Hormonal Contraceptives')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by using Hormonal Contraceptives')
plt.show()
```



```
correlation_coefficient = new_df['Hormonal Contraceptives'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and usage of Hormonal Contraceptives:", correlation_coefficient)
```

```
Correlation between cervix cancer and usage of Hormonal Contraceptives: 0.017413219501020982
```

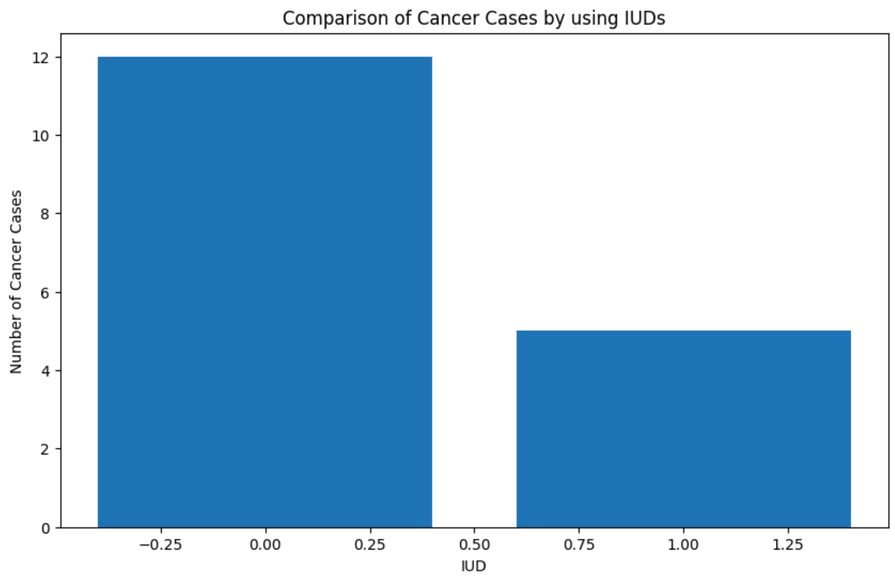
We can see that those that use hormonal contraceptives have higher cancer cases than those that do not.

Task: Find the relationship between using IUD to cervical cancer

```
print("Unique values in 'IUD' column:")
print(new_df['IUD'].unique())
```

```
Unique values in 'IUD' column:
[0. 1.]
```

```
filtered_new_df = new_df[['IUD', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('IUD')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['IUD'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('IUD')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by using IUDs')
plt.show()
```



There is still a significant difference as more than half of those that do not use IUDs have cancer compared to those that uses them, therefore, we can infer that using IUDs do not affect whether an individual has cervix cancer or not

```
correlation_coefficient = new_df['IUD'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and usage of IUD:", correlation_coefficient)
```

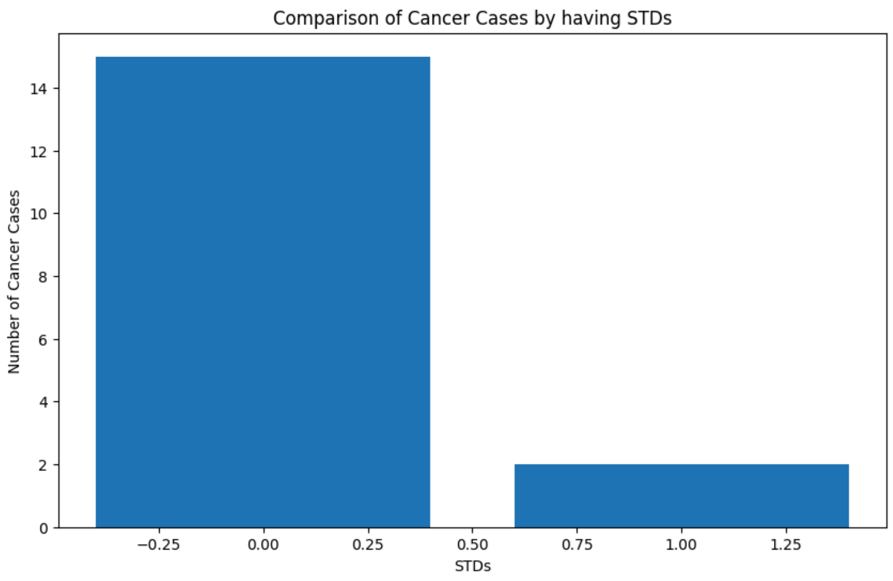
```
Correlation between cervix cancer and usage of IUD: 0.09168034316657442
```

Task: Find the relationship between having STDs to cervical cancer

```
print("Unique values in 'STDs' column:")
print(new_df['STDs'].unique())
```

```
Unique values in 'STDs' column:
[0. 1.]
```

```
filtered_new_df = new_df[['STDs', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('STDs')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['STDs'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('STDs')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by having STDs')
plt.show()
```



```python
correlation_coefficient = new_df['STDs'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and STDs:", correlation_coefficient)
```

```
Correlation between cervix cancer and STDs: 0.009639925649468188
```

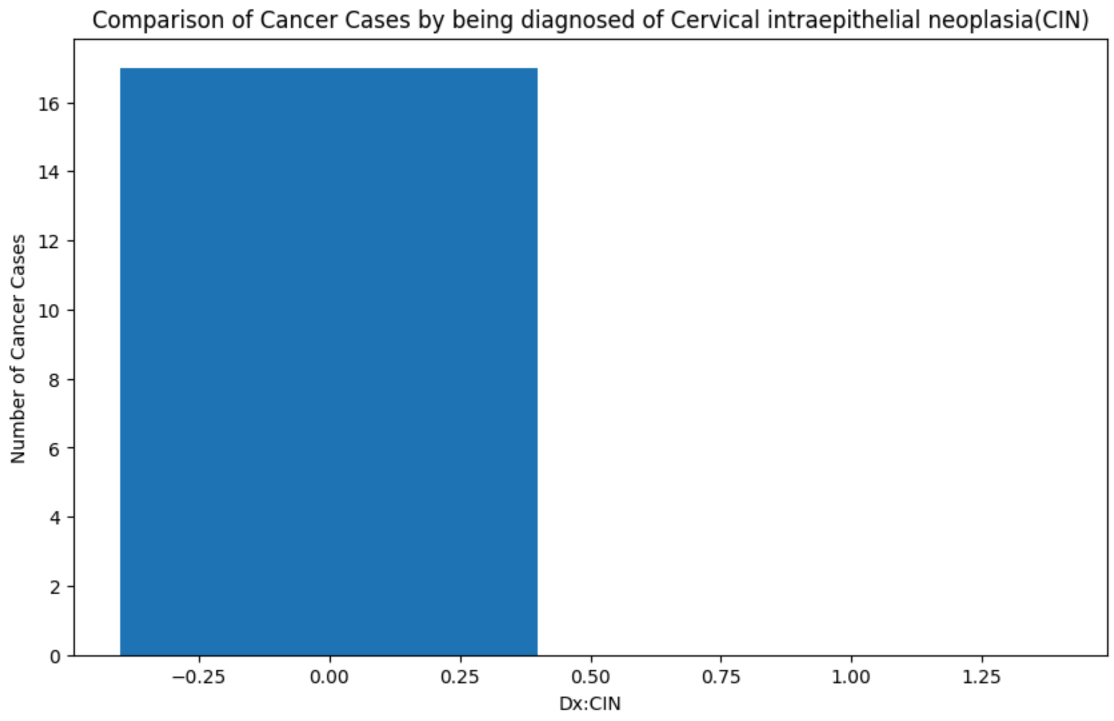
Surprisingly, the number of cancer cases is higher for those that do not have STD than those that has them. Along with the correlation analysis, we can say that having an STD doesn't mean that they will also have cervix cancer

Task: Find the relationship between diagnosed of CIN to cervical cancer

```python
print("Unique values in 'Dx:CIN' column:")
print(new_df['Dx:CIN'].unique())
```

```
Unique values in 'Dx:CIN' column:
[0 1]
```

```python
filtered_new_df = new_df[['Dx:CIN', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Dx:CIN')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Dx:CIN'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Dx:CIN')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by being diagnosed of Cervical intraepithelial neoplasia(CIN)')
plt.show()
```



```
correlation_coefficient = new_df['Dx:CIN'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and Dx:CIN:", correlation_coefficient)
```

```
Correlation between cervix cancer and Dx:CIN: -0.011159149417981603
```

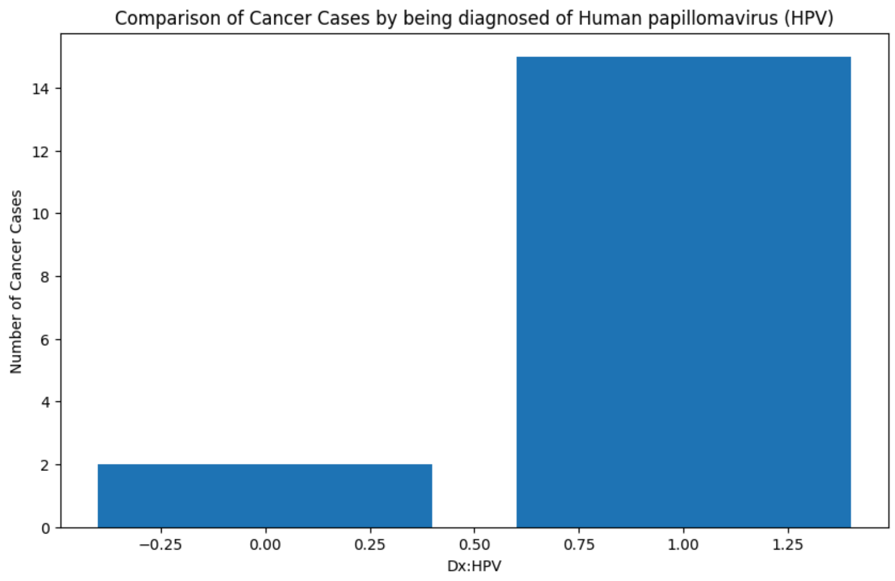
A negative correlation combined with the graph tells us that the people with this diagnosis absolutely has nothing to do with having a cervix cancer.

Task: Find the relationship between diagnosed of HPV to cervical cancer

```
print("Unique values in 'Dx:HPV' column:")
print(new_df['Dx:HPV'].unique())
```

```
Unique values in 'Dx:HPV' column:
[0 1]
```

```
filtered_new_df = new_df[['Dx:HPV', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Dx:HPV')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Dx:HPV'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Dx:HPV')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases by being diagnosed of Human papillomavirus (HPV)')
plt.show()
```



```
correlation_coefficient = new_df['Dx:HPV'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and Dx:HPV:", correlation_coefficient)
```

```
Correlation between cervix cancer and Dx:HPV: 0.9071639187403662
```

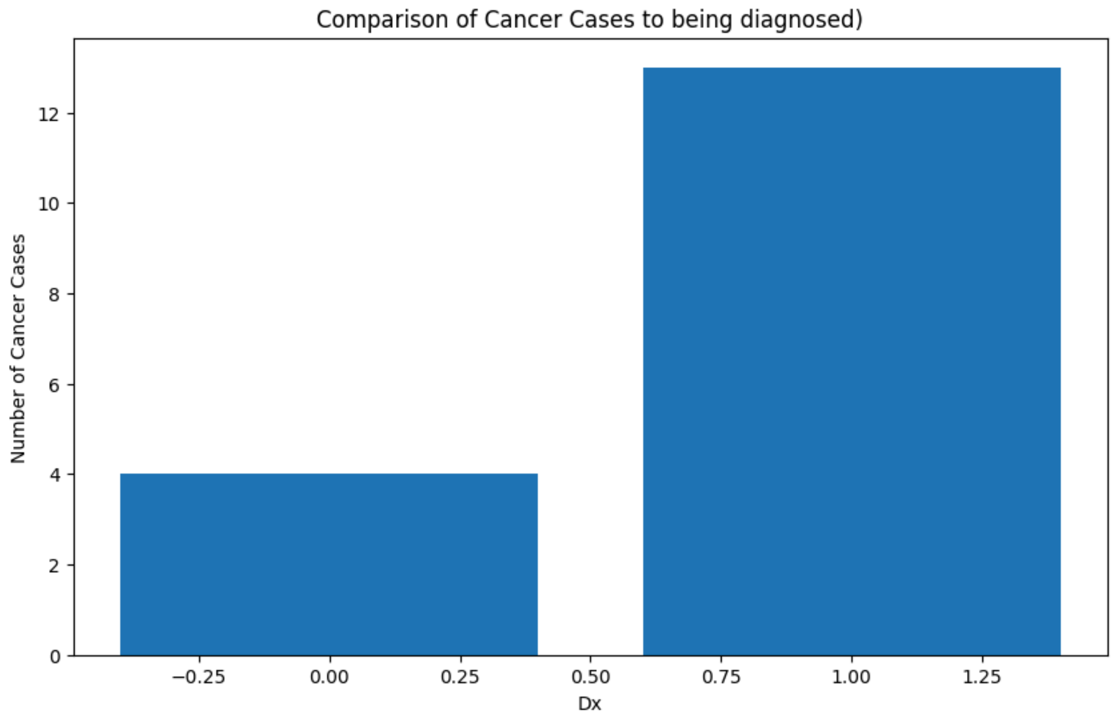
The comparison plus the correlation of HPV to Cervix Cancer tells us that there is a strong relationship between the two.

Task: Find the relationship between doing a dx test to cervical cancer

```
print("Unique values in 'Dx column:")
print(new_df['Dx'].unique())
```

```
Unique values in 'Dx column:
[0 1]
```

```
filtered_new_df = new_df[['Dx', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Dx')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Dx'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Dx')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases to doing the Onoctype dx test)')
plt.show()
```



```
correlation_coefficient = new_df['Dx'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and doing the onoctype dx test:", correlation_coefficient)
```

```
Correlation between cervix cancer and doing the onoctype dx test: 0.7827375530131594
```

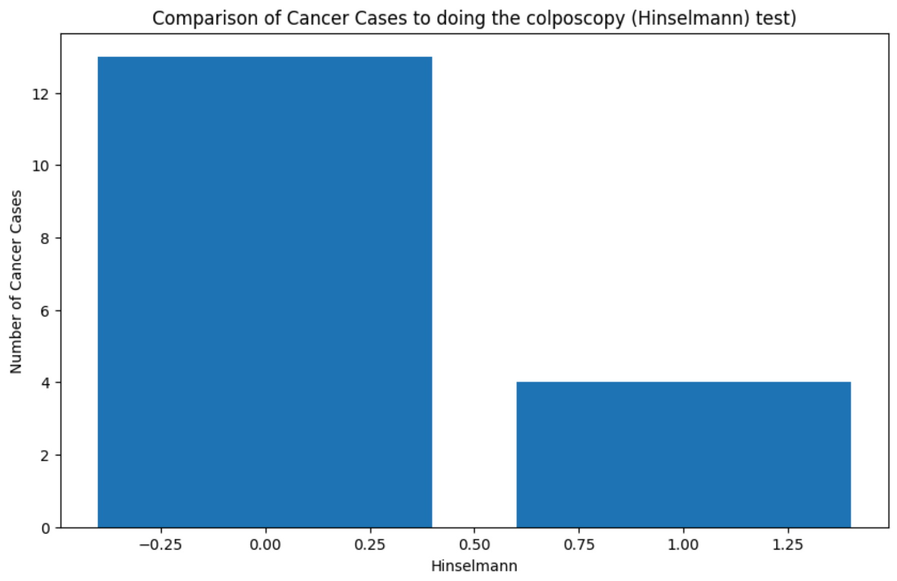
From the comparison and correlation above, we can infer that those that do the test are found to have cancer more than those that do not

Task: Find the relationship between doing colposcopy to cervical cancer

```
print("Unique values in 'Hinselmann column:")
print(new_df['Hinselmann'].unique())
```

```
Unique values in 'Hinselmann column:
[0 1]
```

```
filtered_new_df = new_df[['Hinselmann', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Hinselmann')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Hinselmann'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Hinselmann')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases to doing the colposcopy (Hinselmann) test)')
plt.show()
```



```
correlation_coefficient = new_df['Hinselmann'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and doing the colposcopy test:", correlation_coefficient)
```

```
Correlation between cervix cancer and doing the colposcopy test: 0.1477282313363026
```

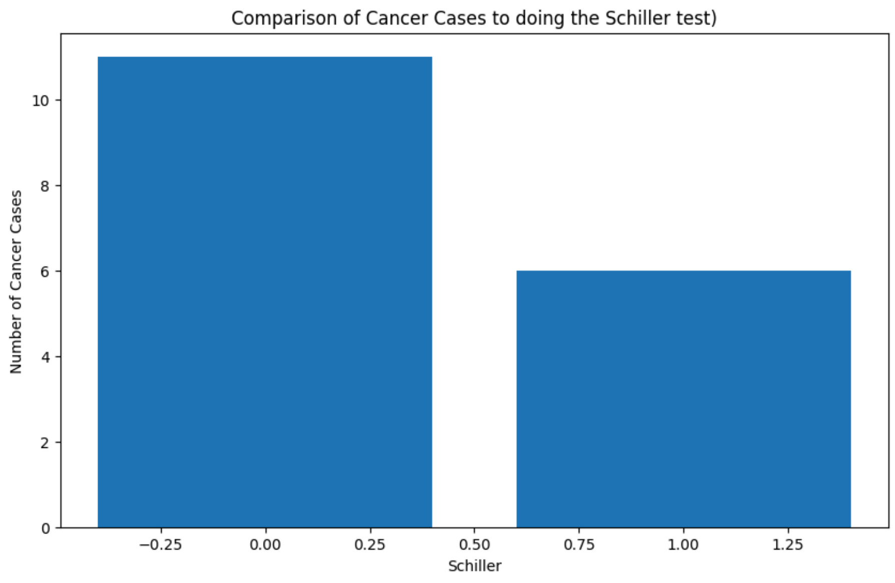
There is only a weak correlation between the colposcopy test and having cancer, on top of the huge difference between those that do not do the test having cancer than those that did it.

Task: Find the relationship between doing the Schiller test to having cervical cancer

```
print("Unique values in 'Schiller column:")
print(new_df['Schiller'].unique())
```

```
Unique values in 'Schiller column:
[0 1]
```

```
filtered_new_df = new_df[['Schiller', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Schiller')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Schiller'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Schiller')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases to doing the Schiller test)')
plt.show()
```



```
correlation_coefficient = new_df['Schiller'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and doing the Schiller test:", correlation_coefficient)
```

```
Correlation between cervix cancer and doing the Schiller test: 0.14368918194911728
```

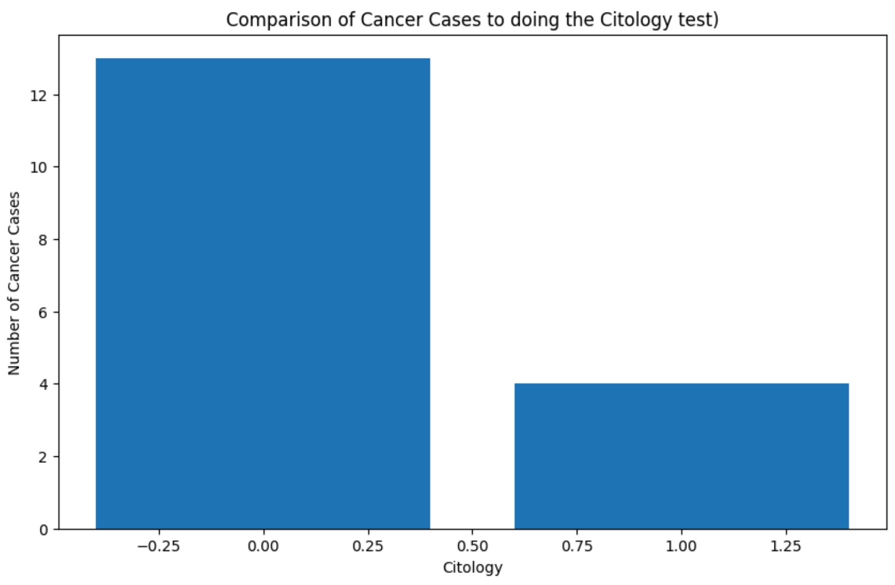
While the plot shows that there isn't that huge of a difference between those that took the schiller test to having cancer or not, the correlation coefficient says otherwise.

Task: Find the relationship between doing the Citology test to having cervical cancer

```
print("Unique values in 'Citology column:")
print(new_df['Citology'].unique())
```

```
Unique values in 'Citology column:
[0 1]
```

```
filtered_new_df = new_df[['Citology', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Citology')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Citology'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Citology')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases to doing the Citology test)')
plt.show()
```



```
correlation_coefficient = new_df['Citology'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and doing the Citology test:", correlation_coefficient)
```

```
Correlation between cervix cancer and doing the Citology test: 0.1235181374669367
```

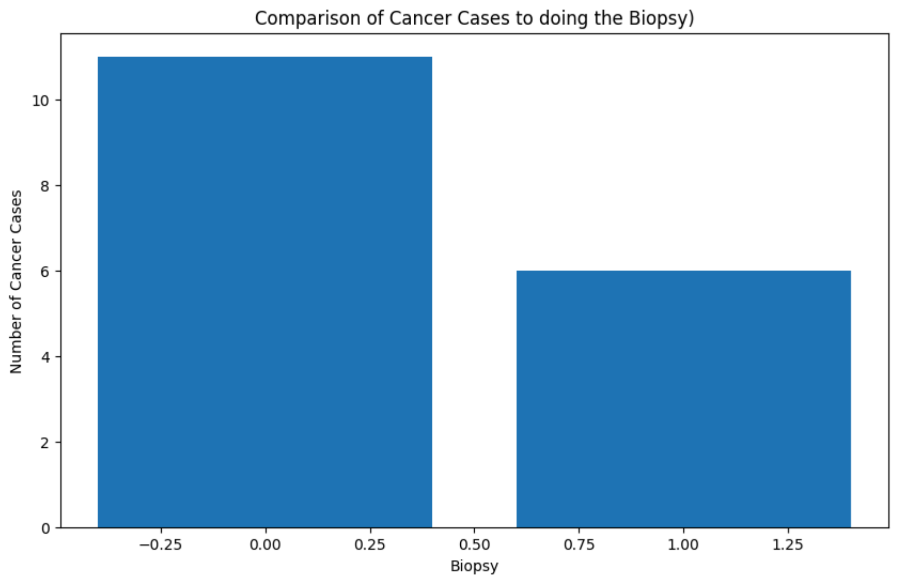
Same with the previous one, there isn't a relationship between the Citology test to having a cervical cancer

Task: Find the relationship between doing the Biopsy to having cervical cancer

```
print("Unique values in 'Biopsy column:")
print(new_df['Biopsy'].unique())
```

```
Unique values in 'Biopsy column:
[0 1]
```

```
filtered_new_df = new_df[['Biopsy', 'Dx:Cancer']].copy()
grouped_new_df = filtered_new_df.groupby('Biopsy')['Dx:Cancer'].sum().reset_index()
grouped_new_df_sorted = grouped_new_df.sort_values(by='Dx:Cancer', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(grouped_new_df_sorted['Biopsy'], grouped_new_df_sorted['Dx:Cancer'])
plt.xlabel('Biopsy')
plt.ylabel('Number of Cancer Cases')
plt.title('Comparison of Cancer Cases to doing the Biopsy)')
plt.show()
```



```
correlation_coefficient = new_df['Biopsy'].corr(new_df['Dx:Cancer'])
print("Correlation between cervix cancer and doing the Biopsy:", correlation_coefficient)
```

```
Correlation between cervix cancer and doing the Biopsy: 0.1860789482965945
```

From the plot and coefficient above, we can see that there isn't a relationship between doing a biopsy to having a cervical cancer.

We can then remove the columns relating to sexual partners, pregnancies, smoking, IUDs, STDs, Dx:CIN, Hinselmann, Schiller, Citology, and biopsy. as they do not affect wether an individual will have cervix cancer or not

Thus, we will only use Age, Dx:HPV, and Dx as the Features and Dx:Cancer as the target.

Logistic Regression

Declare Feature Vector and Target Variable

```
X = new_df[['Age', 'Dx:HPV', 'Dx']]
y = new_df['Dx:Cancer']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0, stratify=y)
X_train.shape, X_test.shape
```

```
((520, 3), (130, 3))
```

```
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (520, 3)
Shape of X_test: (130, 3)
Shape of y_train: (520,)
Shape of y_test: (130,)
```

Feature Engineering

```
categorical = [col for col in X_train.columns if X_train[col].dtypes == '0']
numerical = [col for col in X_train.columns if X_train[col].dtypes != '0']
print("X_train Dtypes:\n", X_train.dtypes)
print("\nCategorical columns:\n", categorical)
print("\nNumerical columns:\n", numerical)
```

```
X_train Dtypes:
 Age       int64
Dx:HPV    int64
Dx        int64
dtype: object

Categorical columns:
 []

Numerical columns:
 ['Age', 'Dx:HPV', 'Dx']
```

Engineering Missing Values in Numerical Variables

```
print("Missing Values in the numerical variables of X_train:\n", X_train[numerical].isnull().sum())
print("\nMissing Values in the numerical variables of X_test:\n", X_test[numerical].isnull().sum())
```

```
Missing Values in the numerical variables of X_train:
 Age       0
Dx:HPV    0
Dx        0
dtype: int64

Missing Values in the numerical variables of X_test:
 Age       0
Dx:HPV    0
Dx        0
dtype: int64
```

```
#Print Percentage of missing values in the numerical variables in training set
for col in numerical:
  if X_train[col].isnull().mean()>0:
    print(col, round(X_train[col].isnull().mean(), 4))
```

Feature Scaling

```
print("X_train description:\n", X_train.describe())
cols = X_train.columns
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train description:
               Age      Dx:HPV          Dx
count  520.000000  520.000000  520.000000
mean    27.386538    0.025000    0.026923
std      8.711752    0.156275    0.162015
min     13.000000    0.000000    0.000000
25%     21.000000    0.000000    0.000000
50%     26.000000    0.000000    0.000000
75%     33.000000    0.000000    0.000000
max     84.000000    1.000000    1.000000
```

```
X_train = pd.DataFrame(X_train, columns = [cols])
X_test = pd.DataFrame(X_test, columns = [cols])
print("X_train description:\n", X_train.describe())
```

```
X_train description:
               Age      Dx:HPV          Dx
count  520.000000  520.000000  520.000000
mean     0.202627    0.025000    0.026923
std      0.122701    0.156275    0.162015
min      0.000000    0.000000    0.000000
```

```
25%      0.112676    0.000000    0.000000
50%      0.183099    0.000000    0.000000
75%      0.281690    0.000000    0.000000
max      1.000000    1.000000    1.000000
```

Model Training

```
logreg = LogisticRegression(solver = 'liblinear', random_state = 0)
logreg.fit(X_train, y_train)
```

```
▾                 LogisticRegression
LogisticRegression(random_state=0, solver='liblinear')
```

Predict Results

```
y_pred_test = logreg.predict(X_test)
y_pred_test
```

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

```
#Probability of getting output as 0
logreg.predict_proba(X_test)[:,0]
```

```
array([0.98370138, 0.98324086, 0.98294117, 0.98353538, 0.98328325,
       0.98378377, 0.98302733, 0.98366004, 0.98302733, 0.9829843 ,
       0.98349362, 0.98361859, 0.98285458, 0.9838248 , 0.98345175,
       0.98324086, 0.98311306, 0.98315577, 0.98398794, 0.98294117,
       0.98390657, 0.98336771, 0.70527716, 0.98302733, 0.15812346,
       0.9829843 , 0.9838248 , 0.98315577, 0.98340978, 0.9838248 ,
       0.98311306, 0.98294117, 0.98307025, 0.98332553, 0.98289793,
       0.98319837, 0.98340978, 0.98430936, 0.98311306, 0.98319837,
       0.98289793, 0.98332553, 0.98328325, 0.98366004, 0.98366004,
       0.98357703, 0.98315577, 0.98336771, 0.98332553, 0.98311306,
       0.98328325, 0.9829843 , 0.98345175, 0.98311306, 0.98345175,
       0.98361859, 0.98319837, 0.98353538, 0.98361859, 0.98357703,
       0.98307025, 0.98319837, 0.98332553, 0.9829843 , 0.98285458,
       0.98307025, 0.98289793, 0.98319837, 0.98345175, 0.98311306,
       0.98328325, 0.98386574, 0.98340978, 0.98276755, 0.98315577,
       0.98324086, 0.98311306, 0.98340978, 0.98357703, 0.98349362,
       0.98340978, 0.98294117, 0.98311306, 0.98370138, 0.98285458,
       0.98285458, 0.98294117, 0.98345175, 0.98353538, 0.98294117,
       0.98340978, 0.98386574, 0.98311306, 0.98311306, 0.98302733,
       0.98315577, 0.98285458, 0.98336771, 0.9829843 , 0.98328325,
       0.98324086, 0.9829843 , 0.98285458, 0.98357703, 0.98315577,
       0.15915478, 0.98324086, 0.98357703, 0.98302733, 0.98349362,
       0.98361859, 0.98353538, 0.98366004, 0.98402847, 0.98361859,
       0.98353538, 0.98328325, 0.9838248 , 0.98289793, 0.98340978,
       0.98319837, 0.98311306, 0.98311306, 0.98289793, 0.98534799,
       0.98289793, 0.98366004, 0.98324086, 0.98281112, 0.98302733])
```

```
#Probability of getting output as 1
logreg.predict_proba(X_test)[:,1]
```

```
array([0.01629862, 0.01675914, 0.01705883, 0.01646462, 0.01671675,
       0.01621623, 0.01697267, 0.01633996, 0.01697267, 0.0170157 ,
       0.01650638, 0.01638141, 0.01714542, 0.0161752 , 0.01654825,
       0.01675914, 0.01688694, 0.01684423, 0.01601206, 0.01705883,
       0.01609343, 0.01663229, 0.29472284, 0.01697267, 0.84187654,
       0.0170157 , 0.0161752 , 0.01684423, 0.01659022, 0.0161752 ,
       0.01688694, 0.01705883, 0.01692975, 0.01667447, 0.01710207,
       0.01680163, 0.01659022, 0.01569064, 0.01688694, 0.01680163,
       0.01710207, 0.01667447, 0.01671675, 0.01633996, 0.01633996,
       0.01642297, 0.01684423, 0.01663229, 0.01667447, 0.01688694,
       0.01671675, 0.0170157 , 0.01654825, 0.01688694, 0.01654825,
       0.01638141, 0.01680163, 0.01646462, 0.01638141, 0.01642297,
       0.01692975, 0.01680163, 0.01667447, 0.0170157 , 0.01714542,
       0.01692975, 0.01710207, 0.01680163, 0.01654825, 0.01688694,
       0.01671675, 0.01613426, 0.01659022, 0.01723245, 0.01684423,
       0.01675914, 0.01688694, 0.01659022, 0.01642297, 0.01650638,
       0.01659022, 0.01705883, 0.01688694, 0.01629862, 0.01714542,
       0.01714542, 0.01705883, 0.01654825, 0.01646462, 0.01705883,
       0.01659022, 0.01613426, 0.01688694, 0.01688694, 0.01697267,
       0.01684423, 0.01714542, 0.01663229, 0.0170157 , 0.01671675,
       0.01675914, 0.0170157 , 0.01714542, 0.01642297, 0.01684423,
       0.84084522, 0.01675914, 0.01642297, 0.01697267, 0.01650638,
       0.01638141, 0.01646462, 0.01633996, 0.01597153, 0.01638141,
       0.01646462, 0.01671675, 0.0161752 , 0.01710207, 0.01659022,
       0.01680163, 0.01688694, 0.01688694, 0.01710207, 0.01465201,
       0.01710207, 0.01633996, 0.01675914, 0.01718888, 0.01697267])
```

Check Accuracy Score

```
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred_test)))
```

```
Model accuracy score: 0.9923
```

A high model accuracy score!

Compare the train-set and test-set accuracy

```
new_df['Dx:Cancer'].value_counts()
```

```
Dx:Cancer
0    633
1     17
Name: count, dtype: int64
```

```
y_pred_train = logreg.predict(X_train)
print("Prediction Training:\n", y_pred_train)
print('\nTraining-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

```
Prediction Training:
 [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 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 1 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
 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 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
 1 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 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]

Training-set accuracy score: 0.9904
```

Check for overfitting and underfitting

```
print('Training set score: {:.4f}'. format(logreg.score(X_train, y_train)))
print('Test set score: {:.4f}'. format(logreg.score(X_test, y_test)))
```

```
Training set score: 0.9904
Test set score: 0.9923
```

As the training set score and test set score are quite comparable to each other, there is no question of overfitting

Using C = 100

```
logreg100 = LogisticRegression(C = 100, solver = 'liblinear', random_state = 0)
logreg100.fit(X_train, y_train)
```

```
▾                      LogisticRegression
LogisticRegression(C=100, random_state=0, solver='liblinear')
```

```
print('Training set score: {:.4f}'.format(logreg100.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(logreg100.score(X_test, y_test)))
```

```
Training set score: 0.9962
Test set score: 1.0000
```

We can see that C = 100 results in higher accuracy, even making the test score flat 1 as well, meaning that this model performed the best and no question of overfitting or underfitting

Using C = 0.01

```
logreg001 = LogisticRegression(C = 0.01, solver = 'liblinear', random_state = 0)
logreg001.fit(X_train, y_train)
```

```
▾                      LogisticRegression
LogisticRegression(C=0.01, random_state=0, solver='liblinear')
```

```
print('Training set score: {:.4f}'.format(logreg001.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(logreg001.score(X_test, y_test)))
```

```
Training set score: 0.9731
Test set score: 0.9769
```

As we can see, the scores remain to be almost the same as each other while remaining to be high even if c = 0.01. There is now no question for underfitting

Comparing the model accuracy with null accruacy

```
y_test.value_counts()
```

```
Dx:Cancer
0    127
1      3
Name: count, dtype: int64
```

```
null_accuracy = (127/(127+3))
print('\nNull Accuracy Score: {0:0.4f}'.format(null_accuracy))
```

```

Null Accuracy Score: 0.9769
```

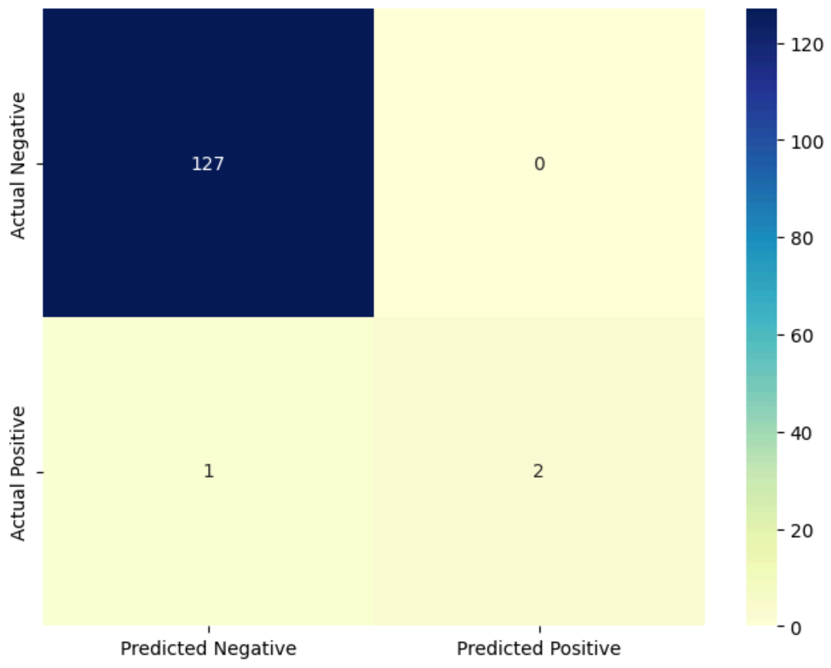
Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred_test)
print('Confusion matrix\n\n', cm)
print("\nTrue Positives(TP) = ", cm[0,0])
print("\nTrue Negatives(TN) = ", cm[1,1])
print("\nFalse Positivves(FP) = ", cm[0,1])
print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[127   0]
 [  1   2]]

True Positives(TP) =  127

True Negatives(TN) =  2

False Positivves(FP) =  0

False Negatives(FN) =  1
```

```
cm_matrix = pd.DataFrame(data=cm, columns=['Predicted Negative', 'Predicted Positive'],
                         index=['Actual Negative', 'Actual Positive'])
plt.figure(figsize=(8, 6))
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

```
<Axes: >
```



```
from sklearn.metrics import roc_auc_score
precision = (127) / (127 + 0)
recall = (127) / (127 + 1)
f1 = 2 * ((precision * recall) / (precision + recall))
roc_auc = roc_auc_score(y_test, y_pred_test)

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-score: {f1:.4f}')
print(f'ROC AUC: {roc_auc:.4f}')
```

```
Precision: 1.0000
Recall: 0.9922
F1-score: 0.9961
ROC AUC: 0.8333
```

Conclusion: The model I created had a high accuracy, especially in getting positive cases right even though they are severly overwhelmed by the number of negative cases, hardly ever making false positives. My scores in precision, recall, f1, and roc auc are also pretty high, meaning that my model is performming very well in terms of precision and its ability to distinguish the two cases. Therefore, I can say that I have developed a model that can accurately predict cervical cancer with the presence of the features at hand.