Predicting Cervical Cancer

STEP 1: IMPORTING LIBRARIES AND DATASETS

In [74]:

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

In [75]:

```
df = pd.read_csv('cervical_cancer.csv')

# display all columns
pd.set_option('display.max_columns', None)
df
```

Out[75]:

	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)	ST
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	34	1.0	?	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	0.0	0.0	0.0	0.0
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	0.0	0.0	0.0	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	0.0	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	0.0	0.0	0.0
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0

858 rows × 36 columns

this is a binary class classification

step 2: Data Analysis

In [76]:

```
df.describe()
```

Out[76]:

	Age	STDs: Number of diagnosis	Dx:Cancer	Dx:CIN	Dx:HPV	Dx	Hinselmann	Schiller	Cit
count	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.00
mean	26.820513	0.087413	0.020979	0.010490	0.020979	0.027972	0.040793	0.086247	0.0512
std	8.497948	0.302545	0.143398	0.101939	0.143398	0.164989	0.197925	0.280892	0.2207
min	13.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
50%	25.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
75%	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
max	84.000000	3.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

In [77]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 858 entries, 0 to 857
Data columns (total 36 columns):
 #   Column                              Non-Null Count  Dtype 
---  ------                              --------------  ----- 
 0   Age                                 858 non-null    int64 
 1   Number of sexual partners           858 non-null    object
 2   First sexual intercourse            858 non-null    object
 3   Num of pregnancies                  858 non-null    object
 4   Smokes                              858 non-null    object
 5   Smokes (years)                      858 non-null    object
 6   Smokes (packs/year)                 858 non-null    object
 7   Hormonal Contraceptives             858 non-null    object
 8   Hormonal Contraceptives (years)     858 non-null    object
 9   IUD                                 858 non-null    object
 10  IUD (years)                         858 non-null    object
 11  STDs                                858 non-null    object
 12  STDs (number)                       858 non-null    object
 13  STDs:condylomatosis                 858 non-null    object
 14  STDs:cervical condylomatosis        858 non-null    object
 15  STDs:vaginal condylomatosis         858 non-null    object
 16  STDs:vulvo-perineal condylomatosis  858 non-null    object
 17  STDs:syphilis                       858 non-null    object
 18  STDs:pelvic inflammatory disease    858 non-null    object
 19  STDs:genital herpes                 858 non-null    object
 20  STDs:molluscum contagiosum          858 non-null    object
 21  STDs:AIDS                           858 non-null    object
 22  STDs:HIV                            858 non-null    object
 23  STDs:Hepatitis B                    858 non-null    object
 24  STDs:HPV                            858 non-null    object
 25  STDs: Number of diagnosis           858 non-null    int64 
 26  STDs: Time since first diagnosis    858 non-null    object
 27  STDs: Time since last diagnosis     858 non-null    object
 28  Dx:Cancer                           858 non-null    int64 
 29  Dx:CIN                              858 non-null    int64 
 30  Dx:HPV                              858 non-null    int64 
 31  Dx                                  858 non-null    int64 
 32  Hinselmann                          858 non-null    int64 
 33  Schiller                            858 non-null    int64 
 34  Citology                            858 non-null    int64 
 35  Biopsy                              858 non-null    int64 
dtypes: int64(10), object(26)
memory usage: 241.4+ KB
```

In [78]:

```
# checking the null values
df.isnull().sum()
```

Out[78]:

```
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
```

no null values

In [79]:

```
# replacing '?' with Null
df = df.replace(to_replace='?', value=np.nan)
df
```

Out[79]:

	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)	ST
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	34	1.0	NaN	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	0.0	0.0	0.0	0.0
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	0.0	0.0	0.0	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	0.0	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	0.0	0.0	0.0
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0

858 rows × 36 columns

In [80]:

```
# checking the number of NAN in the dataset
df.isnull().sum()
```

Out[80]:

```
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
```

In [81]:

```
# plotting a heat map to visualize the number of nan in the data
plt.figure(figsize=(20, 20), facecolor="white")
sns.heatmap(df.isnull(), yticklabels=False)
```

Out[81]:

```
<AxesSubplot:>
```

There are a lot of null values in the columns "STDs: Time since first diagnosis" and "STDs: Time since last diagnosis" This columns will need to be dropped.

In [82]:

```
df = df.drop(columns=['STDs: Time since first diagnosis', 'STDs: Time since
last diagnosis'], axis=1)
df
```

Out[82]:

	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	STD (numb
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	34	1.0	NaN	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	0.0	0.0	0.0
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	0.0	0.0	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	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	0.0	0.0
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	0.0	0.0	0.0

858 rows × 34 columns

my dataframe contains objects as data type which is so generic, we need to convert the columns to numeric data type, which handles float, int and even strings which are in form of numbers. this will aid to perform statistical analysis.

In [83]:

```
df = df.apply(pd.to_numeric)
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 858 entries, 0 to 857
Data columns (total 34 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Age                                 858 non-null    int64
 1   Number of sexual partners           832 non-null    float64
 2   First sexual intercourse            851 non-null    float64
 3   Num of pregnancies                  802 non-null    float64
 4   Smokes                              845 non-null    float64
 5   Smokes (years)                      845 non-null    float64
 6   Smokes (packs/year)                 845 non-null    float64
 7   Hormonal Contraceptives             750 non-null    float64
 8   Hormonal Contraceptives (years)     750 non-null    float64
 9   IUD                                 741 non-null    float64
 10  IUD (years)                         741 non-null    float64
 11  STDs                                753 non-null    float64
 12  STDs (number)                       753 non-null    float64
 13  STDs:condylomatosis                 753 non-null    float64
 14  STDs:cervical condylomatosis        753 non-null    float64
 15  STDs:vaginal condylomatosis         753 non-null    float64
 16  STDs:vulvo-perineal condylomatosis  753 non-null    float64
 17  STDs:syphilis                       753 non-null    float64
 18  STDs:pelvic inflammatory disease    753 non-null    float64
 19  STDs:genital herpes                 753 non-null    float64
 20  STDs:molluscum contagiosum          753 non-null    float64
 21  STDs:AIDS                           753 non-null    float64
 22  STDs:HIV                            753 non-null    float64
 23  STDs:Hepatitis B                    753 non-null    float64
 24  STDs:HPV                            753 non-null    float64
 25  STDs: Number of diagnosis           858 non-null    int64
 26  Dx:Cancer                           858 non-null    int64
 27  Dx:CIN                              858 non-null    int64
 28  Dx:HPV                              858 non-null    int64
 29  Dx                                  858 non-null    int64
 30  Hinselmann                          858 non-null    int64
 31  Schiller                            858 non-null    int64
 32  Citology                            858 non-null    int64
 33  Biopsy                              858 non-null    int64
dtypes: float64(24), int64(10)
memory usage: 228.0 KB
```

In [84]:

```
#i need to replace my null values with mode
df.mode()
```

Out[84]:

	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)
0	23	2.0	15.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

In [95]:

```
for column in df:
    df[column] = df[column].fillna(df[column].mode()[0])
df
```

Out[95]:

	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	STD (numb
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	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	0.0	0.0
2	34	1.0	15.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	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	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	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	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	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	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	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	0.0	0.0

858 rows × 34 columns

In [96]:

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

Out[96]:

```
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
```

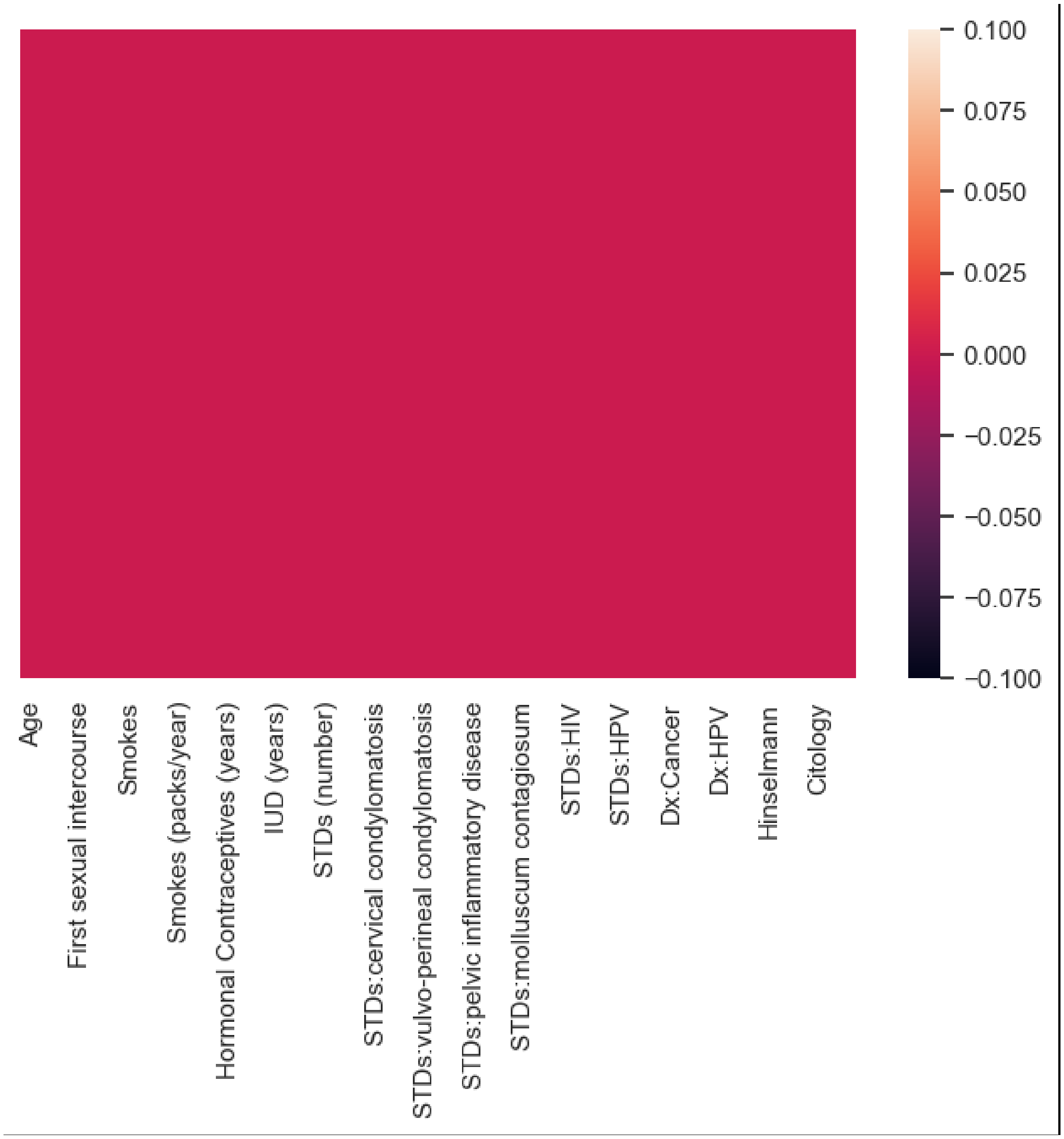
```
STDs:genital herpes                   0
STDs:molluscum contagiosum            0
STDs:AIDS                             0
STDs:HIV                              0
STDs:Hepatitis B                      0
STDs:HPV                              0
STDs: Number of diagnosis             0
Dx:Cancer                             0
Dx:CIN                                0
Dx:HPV                                0
Dx                                    0
Hinselmann                            0
Schiller                              0
Citology                              0
Biopsy                                0
dtype: int64
```

In [102]:

```
plt.figure(figsize=(8, 5), facecolor='white')
sns.heatmap(df.isnull(), yticklabels=False)
```

Out[102]:

```
<AxesSubplot:>
```



no null values now

Data visualization

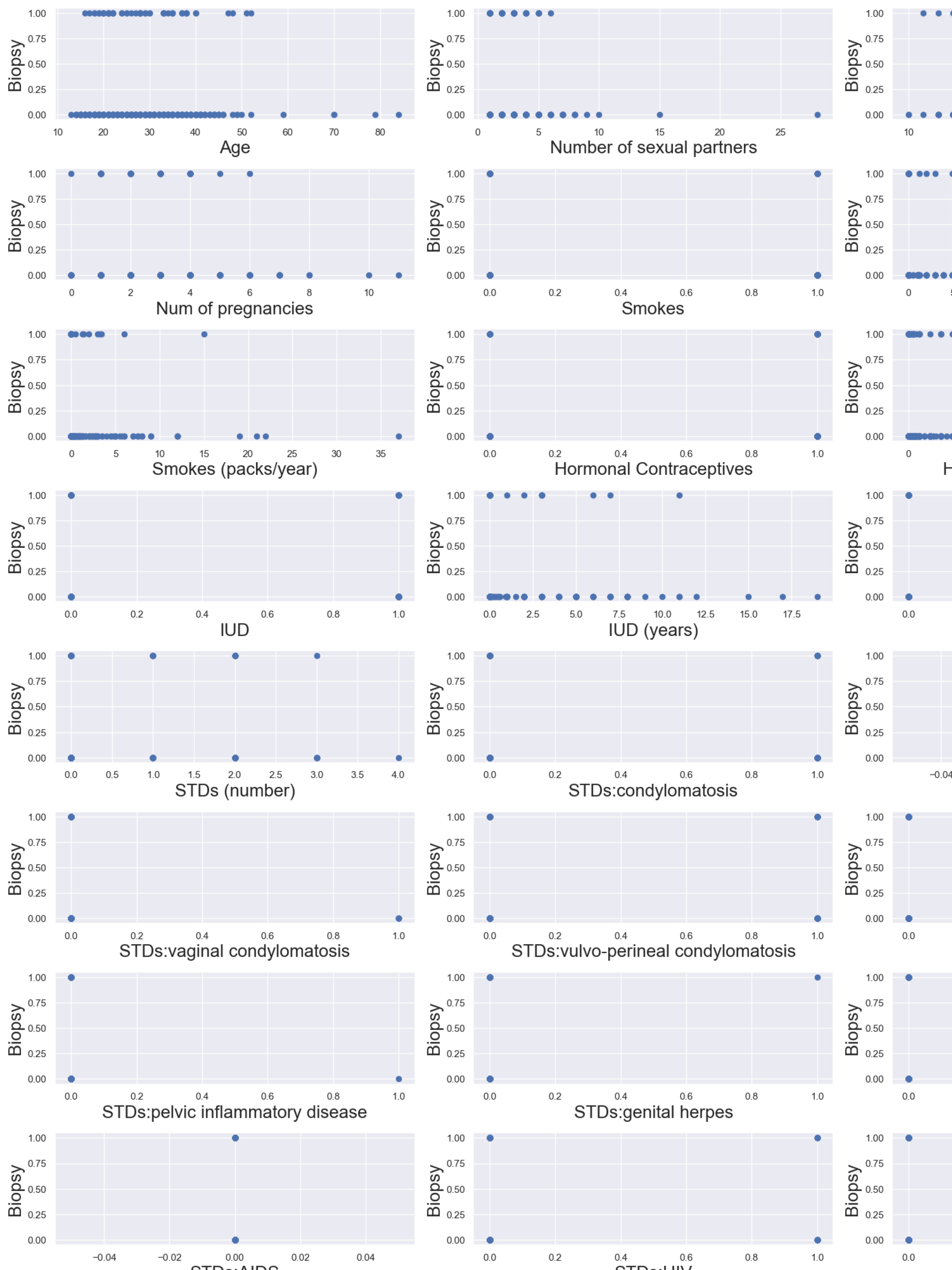
In [109]:

```
# splitting my training data from the target
X = df.drop('Biopsy', axis=1)
y = df['Biopsy']
```

In [110]:

```
plt.figure(figsize=(20,30), facecolor='white')
plotnumber = 1
```

```
for column in X:
    if plotnumber<=len(df.columns) :
        ax = plt.subplot(12,3,plotnumber)
        plt.scatter(X[column],y)
        plt.xlabel(column,fontsize=20)
        plt.ylabel('Biopsy',fontsize=20)
    plotnumber+=1
plt.tight_layout()
```



Biopsy
Age
Number of sexual partners
Num of pregnancies
Smokes
Smokes (packs/year)
Hormonal Contraceptives
IUD
IUD (years)
STDs (number)
STDs:condylomatosis
STDs:vaginal condylomatosis
STDs:vulvo-perineal condylomatosis
STDs:pelvic inflammatory disease
STDs:genital herpes

In [118]:

```
# observing the correlation of the feature columns. columns with very high
correkation should be removed
corr_matrix = df.corr()
corr_matrix
```

Out[118]:

	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)
Age	1.000000	0.085971	0.365248	0.514977	0.057204	0.218261	0.131861	0.029201	0.298892	0.279429	0.215427
Number of sexual partners	0.085971	1.000000	-0.147280	0.082388	0.236858	0.175729	0.174968	0.004027	0.021525	0.032460	0.006252
First sexual intercourse	0.365248	-0.147280	1.000000	-0.046099	-0.123017	-0.058620	-0.057013	-0.009563	0.031976	-0.008826	-0.015697
Num of pregnancies	0.514977	0.082388	-0.046099	1.000000	0.077363	0.172084	0.092214	0.116944	0.221456	0.198134	0.148692
Smokes	0.057204	0.236858	-0.123017	0.077363	1.000000	0.723572	0.493843	0.004036	0.040917	-0.055115	-0.035798
Smokes (years)	0.218261	0.175729	-0.058620	0.172084	0.723572	1.000000	0.724320	-0.013888	0.052436	0.027492	0.038061
Smokes (packs/year)	0.131861	0.174968	-0.057013	0.092214	0.493843	0.724320	1.000000	0.001713	0.043803	0.008226	0.016292
Hormonal Contraceptives	0.029201	0.004027	-0.009563	0.116944	0.004036	-0.013888	0.001713	1.000000	0.370696	0.000188	-0.056548
Hormonal Contraceptives (years)	0.298892	0.021525	0.031976	0.221456	0.040917	0.052436	0.043803	0.370696	1.000000	0.115456	0.017955
IUD	0.279429	0.032460	-0.008826	0.198134	-0.055115	0.027492	0.008226	0.000188	0.115456	1.000000	0.749288
IUD (years)	0.215427	0.006252	-0.015697	0.148692	-0.035798	0.038061	0.016292	-0.056548	0.017955	0.749288	1.000000

	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	0.025241	0.055370	-0.000494	0.055698	0.111289	0.089300	0.029252	-0.045460	0.006918	0.059427	0.022691
STDs (number)	-0.001330	0.041459	0.017948	0.012938	0.100117	0.088605	0.030247	-0.053642	0.002236	0.060591	0.021286
STDs:condylomatosis	-0.013751	0.036925	0.035761	-0.031189	0.055674	0.043504	0.007599	-0.025116	0.016465	0.084794	0.034293
STDs:cervical condylomatosis	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
STDs:vaginal condylomatosis	0.009505	-0.042120	0.073945	0.002754	0.069651	0.114655	0.041939	-0.064390	-0.032782	0.035484	-0.007345
STDs:vulvo-perineal condylomatosis	-0.011499	0.038992	0.039933	-0.030813	0.058468	0.045561	0.008761	-0.029001	0.018090	0.069399	0.033032
STDs:syphilis	0.017457	0.028646	-0.094885	0.150551	0.079358	0.013850	-0.003754	-0.006252	0.003385	-0.020393	-0.026924
STDs:pelvic inflammatory disease	0.024854	0.030929	0.000256	-0.052239	-0.013974	-0.010111	-0.006901	0.023085	-0.011613	-0.011179	-0.008376
STDs:genital herpes	-0.027433	-0.031413	0.024691	-0.028411	-0.013974	-0.010111	-0.006901	0.023085	-0.016362	-0.011179	-0.008376
STDs:molluscum contagiosum	0.000722	0.030929	-0.011961	0.043074	-0.013974	-0.010111	-0.006901	-0.050547	-0.018737	-0.011179	-0.008376
STDs:AIDS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
STDs:HIV	0.005009	0.018752	-0.007627	0.014401	0.056151	0.088930	0.053995	-0.076371	-0.035063	0.007118	0.017928
STDs:Hepatitis B	-0.027433	-0.010633	0.012473	-0.028411	0.083503	0.099313	0.101342	-0.050547	-0.018737	-0.011179	-0.008376

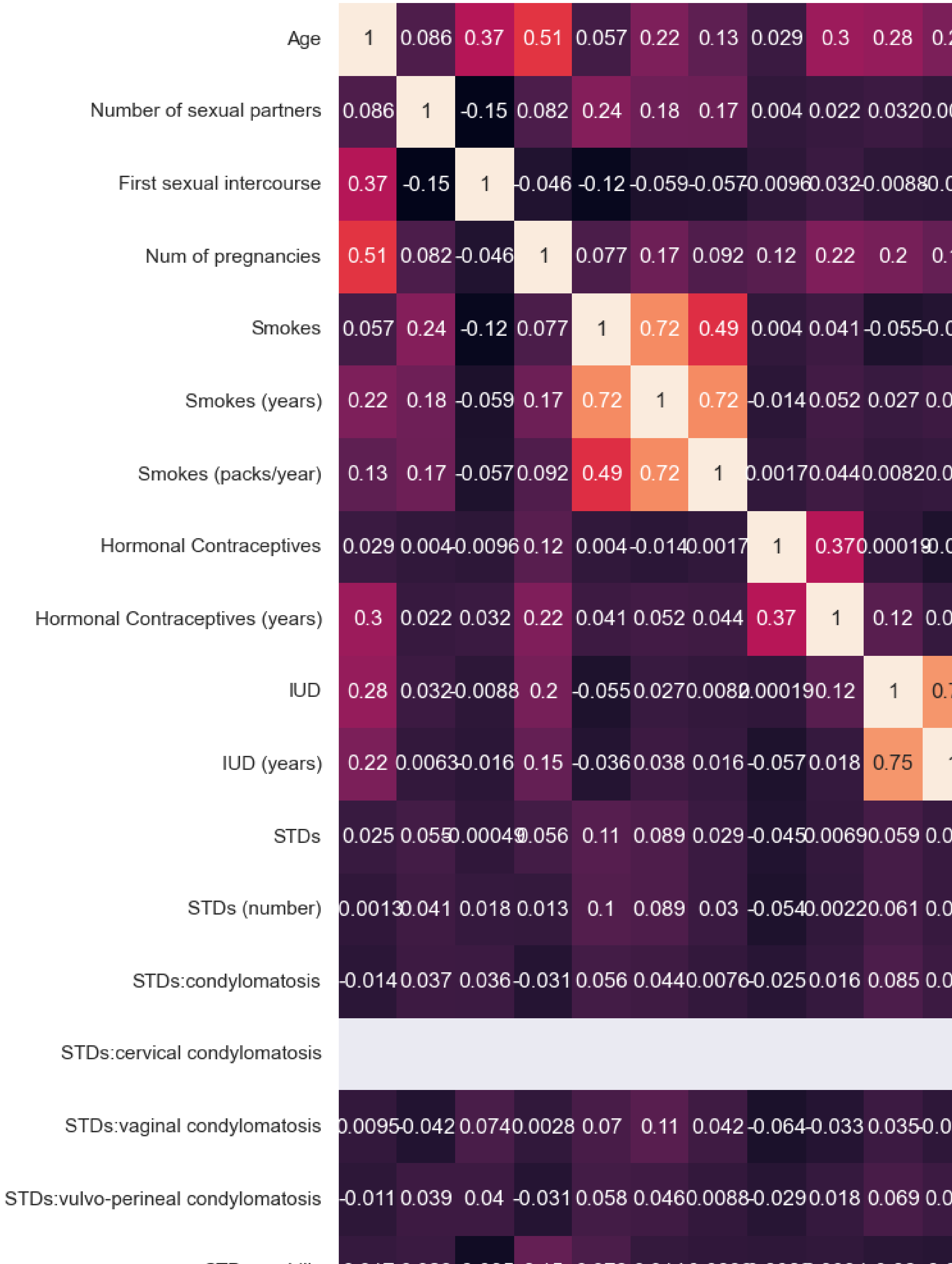
	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	IUD (years)
STDs:HPV	0.040861	0.014360	0.034938	-0.023343	0.049193	0.051201	-0.008015	0.032666	0.054142	-0.015819	-0.011853
STDs: Number of diagnosis	-0.001606	0.053056	-0.011617	0.039195	0.090725	0.078303	0.029912	-0.062199	-0.025662	0.035791	0.012191
Dx:Cancer	0.110340	0.023699	0.067996	0.042765	-0.013470	0.052859	0.107229	0.011278	0.064993	0.117166	0.103148
Dx:CIN	0.061443	0.016669	-0.031960	-0.037752	-0.042119	-0.030476	-0.020800	-0.004397	0.003972	0.043708	0.008887
Dx:HPV	0.101722	0.028646	0.044727	0.054111	0.009737	0.055398	0.109118	0.028808	0.066509	0.062142	0.035869
Dx	0.092635	0.024597	0.036664	-0.003034	-0.069396	-0.050213	-0.034270	-0.007245	-0.008054	0.135778	0.100340
Hinselmann	-0.003967	-0.039098	-0.015311	0.033987	0.033333	0.070352	0.026086	0.012360	0.054264	0.052108	0.014132
Schiller	0.103283	-0.007230	0.005275	0.077526	0.052028	0.093479	0.017200	-0.034002	0.101250	0.096089	0.087032
Citology	-0.016862	0.024067	-0.009594	-0.020131	-0.004639	-0.007275	0.004250	-0.025116	0.084429	0.013292	0.007103
Biopsy	0.055956	-0.000408	0.008771	0.027959	0.028724	0.061204	0.024487	-0.018015	0.097937	0.059231	0.038176

In [121]:

```
# drawing a heatmap to show the correlation
plt.figure(figsize=(25, 25))
sns.heatmap(corr_matrix, annot=True)
```

Out[121]:

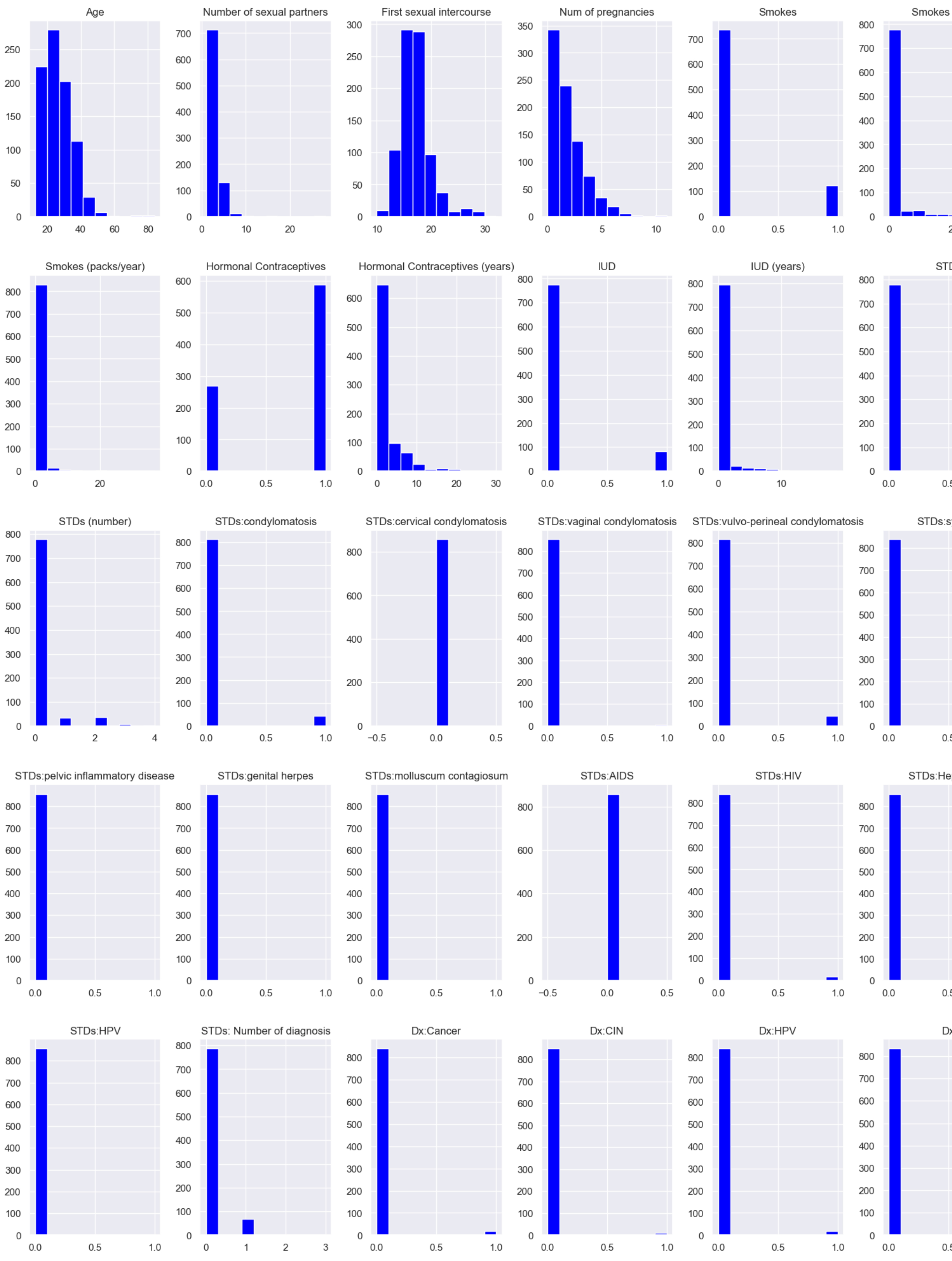
```
<AxesSubplot:>
```


Age 1 0.086 0.37 0.51 0.057 0.22 0.13 0.029 0.3 0.28
Number of sexual partners 0.086 1 -0.15 0.082 0.24 0.18 0.17 0.004 0.022 0.032
First sexual intercourse 0.37 -0.15 1 -0.046 -0.12 -0.059 -0.057 -0.0096 0.032 -0.0088
Num of pregnancies 0.51 0.082 -0.046 1 0.077 0.17 0.092 0.12 0.22 0.2
Smokes 0.057 0.24 -0.12 0.077 1 0.72 0.49 0.004 0.041 -0.055
Smokes (years) 0.22 0.18 -0.059 0.17 0.72 1 0.72 -0.014 0.052 0.027
Smokes (packs/year) 0.13 0.17 -0.057 0.092 0.49 0.72 1 0.0017 0.044 0.0082
Hormonal Contraceptives 0.029 0.004 -0.0096 0.12 0.004 -0.014 0.0017 1 0.37 0.00019
Hormonal Contraceptives (years) 0.3 0.022 0.032 0.22 0.041 0.052 0.044 0.37 1 0.12
IUD 0.28 0.032 -0.0088 0.2 -0.055 0.027 0.0082 0.00019 0.12 1
IUD (years) 0.22 0.0063 -0.016 0.15 -0.036 0.038 0.016 -0.057 0.018 0.75
STDs 0.025 0.055 0.00049 0.056 0.11 0.089 0.029 -0.045 0.0069 0.059
STDs (number) 0.0013 0.041 0.018 0.013 0.1 0.089 0.03 -0.054 0.0022 0.061
STDs:condylomatosis -0.014 0.037 0.036 -0.031 0.056 0.044 0.0076 -0.025 0.016 0.085
STDs:cervical condylomatosis
STDs:vaginal condylomatosis 0.0095 -0.042 0.074 0.0028 0.07 0.11 0.042 -0.064 -0.033 0.035
STDs:vulvo-perineal condylomatosis -0.011 0.039 0.04 -0.031 0.058 0.046 0.0088 -0.029 0.018 0.069

In [131]:

```
# using a histogram to visualize the whole dataset

df.hist(bins = 10, figsize = (20,30), color='blue', sharex=False)
plt.show()
```


Age
Number of sexual partners
First sexual intercourse
Num of pregnancies
Smokes
Smokes
Smokes (packs/year)
Hormonal Contraceptives
Hormonal Contraceptives (years)
IUD
IUD (years)
STD
STDs (number)
STDs:condylomatosis
STDs:cervical condylomatosis
STDs:vaginal condylomatosis
STDs:vulvo-perineal condylomatosis
STDs:s
STDs:pelvic inflammatory disease
STDs:genital herpes
STDs:molluscum contagiosum
STDs:AIDS
STDs:HIV
STDs:He
STDs:HPV
STDs: Number of diagnosis
Dx:Cancer
Dx:CIN
Dx:HPV
Dx

4) PREPARING DATA BEFORE TRAINING

In [132]:

```
X
```

Out[132]:

	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	ST (nu
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	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	0.0	0.0
2	34	1.0	15.0	1.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	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	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	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	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	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	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	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	0.0	0.0

858 rows × 33 columns

In [133]:

```
y
```

Out[133]:

```
0      0
1      0
2      0
3      0
4      0
      ..
853    0
854    0
855    0
856    0
857    0
Name: Biopsy, Length: 858, dtype: int64
```

In [135]:

```
# scaling the data
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [136]:

```
x_scaled = scaler.fit_transform(X)
x_scaled
```

Out[136]:

```
array([[-1.03856336,  0.90542946, -0.70779867, ..., -0.20622158,
        -0.3072259 , -0.23249528],
       [-1.39179566, -0.91961003, -1.06544958, ..., -0.20622158,
        -0.3072259 , -0.23249528],
       [ 0.84534227, -0.91961003, -0.70779867, ..., -0.20622158,
        -0.3072259 , -0.23249528],
       ...,
       [-0.21435465, -0.31126353,  0.00750317, ..., -0.20622158,
        -0.3072259 ,  4.30116263],
       [ 0.72759817, -0.31126353,  2.51105958, ..., -0.20622158,
        -0.3072259 , -0.23249528],
       [ 0.25662176, -0.31126353,  1.08045591, ..., -0.20622158,
        -0.3072259 , -0.23249528]])
```

In [138]:

```
# splitting the data into training and testing dataset
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y,
test_size=0.30, random_state=355)
```

5) TRAINING AND EVALUATING XGBOOST CLASSIFIER

In [141]:

```
import xgboost as xgb
from xgboost import XGBClassifier
```

In [143]:

```
xgboost_model = XGBClassifier(objective='binary:logistic')
```

In [144]:

```
xgboost_model.fit(x_train, y_train)
```

Out[144]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

In [146]:

```
# model evaluation on the training data
xgboost_model.score(x_train, y_train)
```

Out[146]:

```
0.9983333333333333
```

In [158]:

```
# model evaluation on the testing data
xgboost_model.score(x_test, y_test)
```

Out[158]:

```
0.9496124031007752
```

the valiation between the training data and the test data, seems like my model has overfitted on the training data

hyperparameter tuning

In [163]:

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'learning_rate':[0.1],
    'max_depth' : range(20),
    'n_estimators' : range(100)
}
```

In [164]:

```
grid_search = GridSearchCV(XGBClassifier(objective='binary:logistic'),
param_grid, verbose=3)
grid_search.fit(x_train, y_train)
Fitting 5 folds for each of 2000 candidates, totalling 10000 fits
[CV 1/5] END learning_rate=0.1, max_depth=0, n_estimators=0;, score=0.942
total time=   0.0s
[CV 2/5] END learning_rate=0.1, max_depth=0, n_estimators=0;, score=0.942
total time=   0.0s
[CV 3/5] END learning_rate=0.1, max_depth=0, n_estimators=0;, score=0.933
total time=   0.0s
[CV 4/5] END learning_rate=0.1, max_depth=0, n_estimators=0;, score=0.933
total time=   0.0s
[CV 5/5] END learning_rate=0.1, max_depth=0, n_estimators=0;, score=0.933
total time=   0.0s
[CV 1/5] END learning_rate=0.1, max_depth=0, n_estimators=1;, score=nan total
time=   0.0s
[CV 2/5] END learning_rate=0.1, max_depth=0, n_estimators=1;, score=nan total
time=   0.0s
[CV 3/5] END learning_rate=0.1, max_depth=0, n_estimators=1;, score=nan total
time=   0.0s
[CV 4/5] END learning_rate=0.1, max_depth=0, n_estimators=1;, score=nan total
time=   0.0s
```

Out[164]:

```
GridSearchCV(estimator=XGBClassifier(base_score=None, booster=None,
                                     callbacks=None, colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None,
                                     early_stopping_rounds=None,
                                     enable_categorical=False,
eval_metric=None,
                                     feature_types=None, gamma=None,
                                     gpu_id=None, grow_policy=None,
                                     importance_type=None,
                                     interaction_constraints=None,
                                     learning_rate=None, max_bin=None,
                                     max_cat_threshold=None,
                                     max_cat_to_onehot=None,
                                     max_delta_step=None, max_depth=None,
                                     max_leaves=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs=None,
                                     num_parallel_tree=None, predictor=None,
                                     random_state=None, ...),
             param_grid={'learning_rate': [0.1], 'max_depth': range(0, 20),
                         'n_estimators': range(0, 100)},
             verbose=3)
```

In [167]:

```
grid_search.best_params_
```

Out[167]:

```
{'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 63}
```

In [168]:

```
new_xgboost_model = XGBClassifier(learning_rate=0.1, max_depth=1,
n_estimators=63)
new_xgboost_model.fit(x_train, y_train)
```

Out[168]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=1, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=63, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

In [170]:

```
# evaluation of the new model on the training data
new_xgboost_model.score(x_train, y_train)
```

Out[170]:

```
0.965
```

In [171]:

```
# evaluation of the new model on the test data
new_xgboost_model.score(x_test, y_test)
```

Out[171]:

```
0.9651162790697675
```

my test and the train score is almost same, now i can say that my model is performing better and has not overfitted on the training data

In [186]:

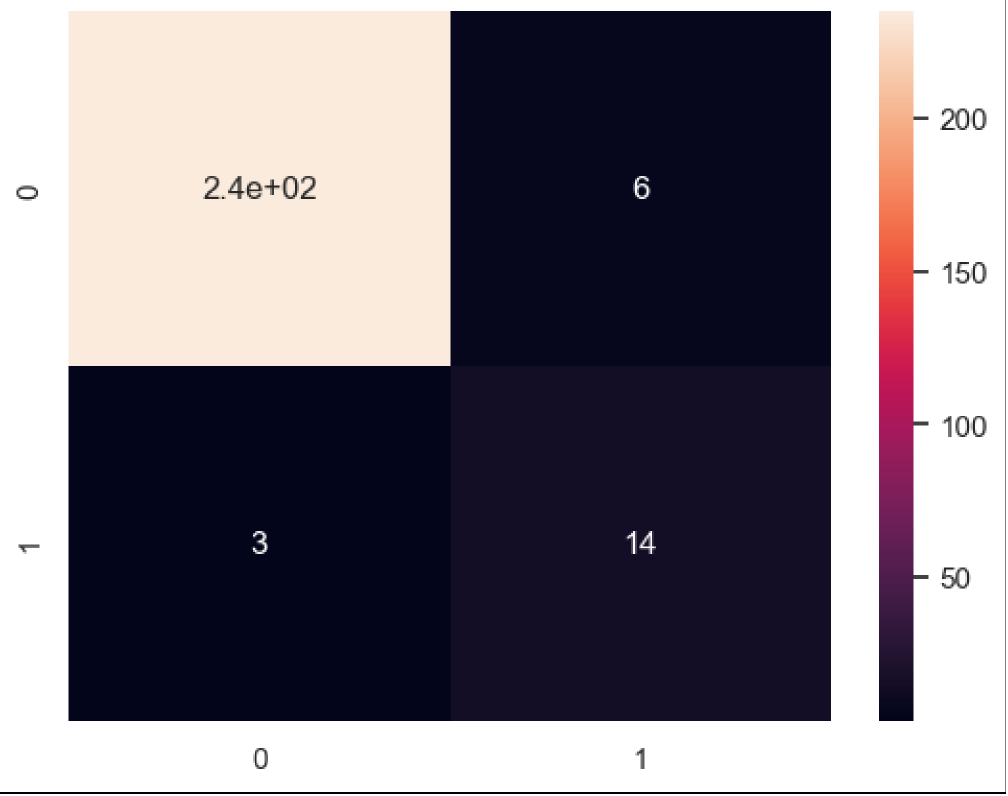
```
# evaluating the model using a confusion matrix
from sklearn.metrics import confusion_matrix, classification_report
y_pred = new_xgboost_model.predict(x_test)
cof_mat = confusion_matrix(y_test, y_pred)
cof_mat
```

Out[186]:

```
array([[235,   6],
       [  3,  14]], dtype=int64)
```

In [187]:

```
# drawing a confusion matrix using heat map
sns.heatmap(cof_mat, annot=True)
plt.show()
```



In [188]:

```
print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       241
           1       0.70      0.82      0.76        17

    accuracy                           0.97       258
   macro avg       0.84      0.90      0.87       258
weighted avg       0.97      0.97      0.97       258
```

6) saving new_xgb_model

In [189]:

```
import pickle
with open('new_xgboost_model', 'wb') as file:
    pickle.dump(new_xgboost_model, file)
    file.close
```

In [190]:

```
# saving the standardscaler model
with open('StandardScaler.pickle', 'wb') as file:
    pickle.dump(scaler, file)
    file.close()
```

In [ ]: