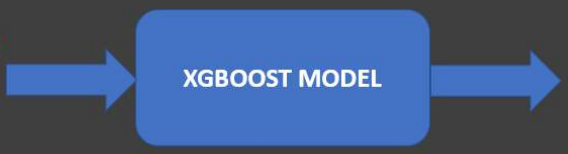
TASK #1: UNDERSTAND THE PROBLEM STATEMENT/GOAL

CREVICAL CANCER RISK PREDICTION

- In this hands-on project, we will build and train an XGBoost model to predict cervical cancer in 858 patients.
- The dataset was collected at 'Hospital Universitario de Caracas' in Caracas, Venezuela and contains demographic information, habits, and historic medical records of 858 patients.
- Cervical cancer kills about 4,000 women in the U.S. and about 300,000 women worldwide. Due to increased medical screening, cervical cancer death rate has been reduced by 74% from 1955 to 1992.
- Studies have shown that High sexual activity Human papilloma virus (HPV) is one of the key factors that increases the risk of having cervical cancer.
- The presence of hormones in oral contraceptives, having many children, and smoking increase the risk for developing cervical cancer, particularly in women infected with HPV. Also, people with weak immune systems (HIV/AIDS) have high risk of HPV.

INPUT FEATURES

- AGE
- NUM OF PREGNENCIES
- SMOKES
- IUD
- STDs



TARGET VARIABLE (BIOPSY)

YES

NO

https://www.kaggle.com/loveall/cervical-cancer-risk-classification
https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29

1

INSTRUCTOR

- Adjunct professor & online instructor
- Passionate about artificial intelligence, machine learning, and electric vehicles
- Taught 200,000+ students globally
- MBA (2018), Ph.D. (2014), M.A.Sc (2011)

Ryan Ahmed, Ph.D.

TASK #2: IMPORT DATASET AND LIBRARIES

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import zipfile

!pip install plotly
import plotly.express as px
from jupyterthemes import jtplot
jtplot.style(theme = 'monokai', context = 'notebook', ticks = True, grid = False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see the x and y axes clearly
# If you don't run this code line, you will notice that the xlabel and ylabel on any plot is black
on black and it will be hard to see them.
```

```
Requirement already satisfied: plotly in c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages (4.14.1)
Requirement already satisfied: retrying>=1.3.3 in c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages (from pl
otly) (1.3.3)
Requirement already satisfied: six in c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages (from plotly) (1.14.
0)


tensorboard 2.1.1 has requirement setuptools>=41.0.0, but you'll have setuptools 39.0.1 which is incompatible.
google-auth 1.12.0 has requirement setuptools>=40.3.0, but you'll have setuptools 39.0.1 which is incompatible.
You are using pip version 10.0.1, however version 24.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

In [37]:
```
# import the csv files using pandas
cancer_df = pd.read_csv('cervical_cancer.csv')

# (int) Age
# (int) Number of sexual partners
#  (int) First sexual intercourse (age)
# (int) Num of pregnancies
# (bool) Smokes
# (bool) Smokes (years)
# (bool) Smokes (packs/year)
# (bool) Hormonal Contraceptives
# (int) Hormonal Contraceptives (years)
# (bool) IUD ("IUD" stands for "intrauterine device" and used for birth control
# (int) IUD (years)
# (bool) STDs (Sexually transmitted disease)
# (int) STDs (number)
# (bool) STDs:condylomatosis
# (bool) STDs:cervical condylomatosis
# (bool) STDs:vaginal condylomatosis
# (bool) STDs:vulvo-perineal condylomatosis
# (bool) STDs:syphilis
# (bool) STDs:pelvic inflammatory disease
# (bool) STDs:genital herpes
# (bool) STDs:molluscum contagiosum
# (bool) STDs:AIDS
# (bool) STDs:HIV
# (bool) STDs:Hepatitis B
# (bool) STDs:HPV
# (int) STDs: Number of diagnosis
# (int) STDs: Time since first diagnosis
# (int) STDs: Time since last diagnosis
# (bool) Dx:Cancer
# (bool) Dx:CIN
# (bool) Dx:HPV
# (bool) Dx
# (bool) Hinselmann: target variable - A colposcopy is a procedure in which doctors examine the ce
rvix.
# (bool) Schiller: target variable - Schiller's Iodine test is used for cervical cancer diagnosis
# (bool) Cytology: target variable - Cytology is the exam of a single cell type used for cancer sc
reening.
# (bool) Biopsy: target variable - Biopsy is performed by removing a piece of tissue and examine i
t under microscope,
# Biopsy is the main way doctors diagnose most types of cancer.
```

In [38]:
```
# Let's explore the dataframe
cancer_df.head()
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STD Tim sinc fir diagnos
0	18	4.0	15.0	1.0	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	...	?
2	34	1.0	?	1.0	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	...	?
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	...	?

5 rows × 36 columns

MINI CHALLENGE #1:

- Print the last 20 rows in the dataframe

In [39]:
```
cancer_df_20 = cancer_df.tail(20)
cancer_df_20_new = cancer_df.iloc[-20:]
cancer_df_20_new
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	ST T s diagn
838	35	3.0	18.0	3.0	0.0	0.0	0.0	1.0	5.0	0.0	...	?
839	31	3.0	19.0	1.0	0.0	0.0	0.0	1.0	0.08	1.0	...	?
840	24	2.0	16.0	3.0	0.0	0.0	0.0	1.0	5.0	0.0	...	?
841	23	2.0	15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?
842	36	3.0	16.0	3.0	1.0	6.0	0.3	1.0	2.0	0.0	...	?
843	30	3.0	14.0	3.0	0.0	0.0	0.0	1.0	2.0	0.0	...	?
844	26	8.0	15.0	1.0	1.0	9.0	1.35	1.0	5.0	1.0	...	?
845	19	2.0	15.0	2.0	0.0	0.0	0.0	1.0	0.75	0.0	...	?
846	35	2.0	17.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?
847	30	3.0	22.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?
848	31	3.0	18.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	...	?
849	32	3.0	18.0	1.0	1.0	11.0	0.16	1.0	6.0	0.0	...	?
850	19	1.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?
851	23	2.0	15.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?
852	43	3.0	17.0	3.0	0.0	0.0	0.0	1.0	5.0	0.0	...	?
853	34	3.0	18.0	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	...	?
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	...	?

20 rows × 36 columns

TASK #3: PERFORM EXPLORATORY DATA ANALYSIS

In [40]:
```
# Get data frame info
cancer_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 [41]:
```
# Get the statistics of the data frame
cancer_df.describe()
```

	Age	STDs: Number of diagnosis	Dx:Cancer	Dx:CIN	Dx:HPV	Dx	Hinselmann	Schiller	Citology	Biopsy
count	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000	858.000000
mean	26.820513	0.087413	0.020979	0.010490	0.020979	0.027972	0.040793	0.086247	0.051282	0.064103
std	8.497948	0.302545	0.143398	0.101939	0.143398	0.164989	0.197925	0.280892	0.220701	0.245078
min	13.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	25.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	84.000000	3.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [42]:
```
# Notice many question marks indicating missing values
cancer_df
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	ST s diagn
0	18	4.0	15.0	1.0	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	...	?
2	34	1.0	?	1.0	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	...	?
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	...	?
...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0	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	...	?
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	...	?

858 rows × 36 columns

In [43]:
```
# Let's replace '?' with NaN
cancer_df = cancer_df.replace('?', np.nan)
cancer_df
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	ST s diagn
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	NaN
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	NaN
2	34	1.0	NaN	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	NaN
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	...	NaN
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	NaN
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	0.0	...	NaN
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	NaN
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	NaN
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	...	NaN

858 rows × 36 columns

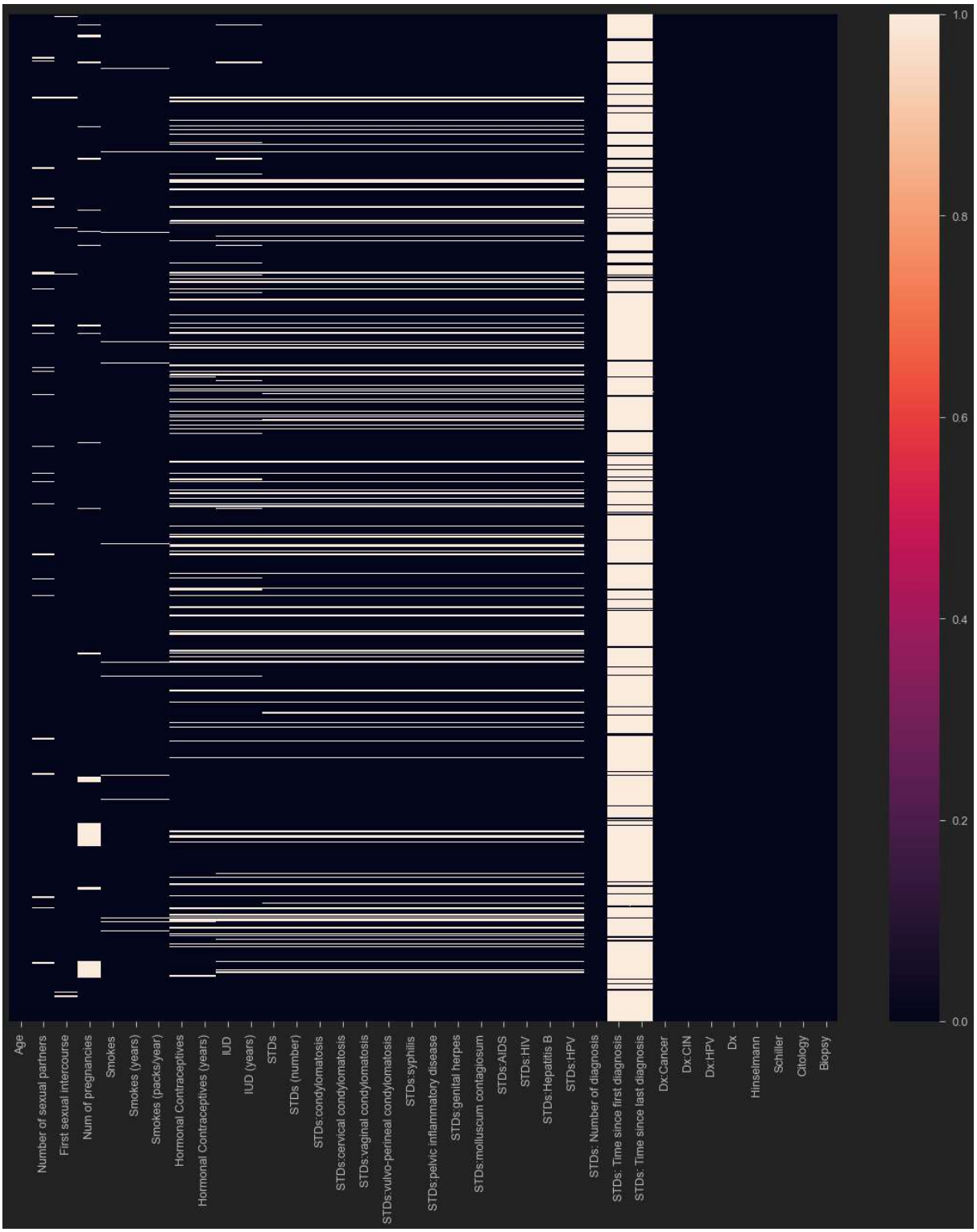
In [44]:
```
# Plot heatmap
cancer_df.isnull()
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	S diag
0	False	False	False	False	False	False	False	False	False	False	...	True
1	False	False	False	False	False	False	False	False	False	False	...	True
2	False	False	True	False	False	False	False	False	False	False	...	True
3	False	False	False	False	False	False	False	False	False	False	...	True
4	False	False	False	False	False	False	False	False	False	False	...	True
...	...	...	...	...	...	...	...	...	...	...	...	...
853	False	False	False	False	False	False	False	False	False	False	...	True
854	False	False	False	False	False	False	False	False	False	False	...	True
855	False	False	False	False	False	False	False	False	False	False	...	True
856	False	False	False	False	False	False	False	False	False	False	...	True
857	False	False	False	False	False	False	False	False	False	False	...	True

858 rows × 36 columns

In [45]:
```
plt.figure(figsize = (20, 20))
sns.heatmap(cancer_df.isnull(), yticklabels = False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x202189fec88>
```



In [46]:
```
# Get data frame info
cancer_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           832 non-null    object
 2   First sexual intercourse            851 non-null    object
 3   Num of pregnancies                  802 non-null    object
 4   Smokes                              845 non-null    object
 5   Smokes (years)                      845 non-null    object
 6   Smokes (packs/year)                 845 non-null    object
 7   Hormonal Contraceptives             750 non-null    object
 8   Hormonal Contraceptives (years)     750 non-null    object
 9   IUD                                 741 non-null    object
 10  IUD (years)                         741 non-null    object
 11  STDs                                753 non-null    object
 12  STDs (number)                       753 non-null    object
 13  STDs:condylomatosis                 753 non-null    object
 14  STDs:cervical condylomatosis        753 non-null    object
 15  STDs:vaginal condylomatosis         753 non-null    object
 16  STDs:vulvo-perineal condylomatosis  753 non-null    object
 17  STDs:syphilis                       753 non-null    object
 18  STDs:pelvic inflammatory disease    753 non-null    object
 19  STDs:genital herpes                 753 non-null    object
 20  STDs:molluscum contagiosum          753 non-null    object
 21  STDs:AIDS                           753 non-null    object
 22  STDs:HIV                            753 non-null    object
 23  STDs:Hepatitis B                    753 non-null    object
 24  STDs:HPV                            753 non-null    object
 25  STDs: Number of diagnosis           858 non-null    int64 
 26  STDs: Time since first diagnosis    71 non-null     object
 27  STDs: Time since last diagnosis     71 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 [47]:
```
# Since STDs: Time since first diagnosis  and STDs: Time since last diagnosis have more than 80% m
issing values
# we can drop them
cancer_df = cancer_df.drop(columns = ['STDs: Time since first diagnosis','STDs: Time since last di
agnosis'])
cancer_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:
0	18	4.0	15.0	1.0	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
2	34	1.0	NaN	1.0	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
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	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
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	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
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	0.5	0.0	...	0.0

858 rows × 34 columns

In [48]:
```
# Since most of the column types are object, we are not able to get the statistics of the datafram
e.
# Convert them to numeric type

cancer_df = cancer_df.apply(pd.to_numeric)
cancer_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 [49]:
```
# Get the statistics of the dataframe
cancer_df.describe()
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	
count	858.000000	832.000000	851.000000	802.000000	845.000000	845.000000	845.000000	750.000000	750.000000	74
mean	26.820513	2.527644	16.995300	2.275561	0.145562	1.219721	0.453144	0.641333	2.256419	0.1
std	8.497948	1.667760	2.803355	1.447414	0.352876	4.089017	2.226610	0.479929	3.764254	0.3
min	13.000000	1.000000	10.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	20.000000	2.000000	15.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	25.000000	2.000000	17.000000	2.000000	0.000000	0.000000	0.000000	1.000000	0.500000	0.0
75%	32.000000	3.000000	18.000000	3.000000	0.000000	0.000000	0.000000	1.000000	3.000000	0.0
max	84.000000	28.000000	32.000000	11.000000	1.000000	37.000000	37.000000	1.000000	30.000000	1.0

8 rows × 34 columns

In [50]:
```
cancer_df.mean()
```

```
Age                                 26.820513
Number of sexual partners            2.527644
First sexual intercourse            16.995300
Num of pregnancies                   2.275561
Smokes                               0.145562
Smokes (years)                       1.219721
Smokes (packs/year)                  0.453144
Hormonal Contraceptives              0.641333
Hormonal Contraceptives (years)      2.256419
IUD                                  0.112011
IUD (years)                          0.514804
STDs                                 0.104914
STDs (number)                        0.176627
STDs:condylomatosis                  0.058433
STDs:cervical condylomatosis         0.000000
STDs:vaginal condylomatosis          0.005312
STDs:vulvo-perineal condylomatosis    0.057105
STDs:syphilis                        0.023904
STDs:pelvic inflammatory disease     0.001328
STDs:genital herpes                  0.001328
STDs:molluscum contagiosum           0.001328
STDs:AIDS                            0.000000
STDs:HIV                             0.023904
STDs:Hepatitis B                     0.001328
STDs:HPV                             0.002656
STDs: Number of diagnosis            0.087413
Dx:Cancer                            0.020979
Dx:CIN                               0.010490
Dx:HPV                               0.020979
Dx                                   0.027972
Hinselmann                           0.040793
Schiller                             0.086247
Citology                             0.051282
Biopsy                               0.064103
dtype: float64
```

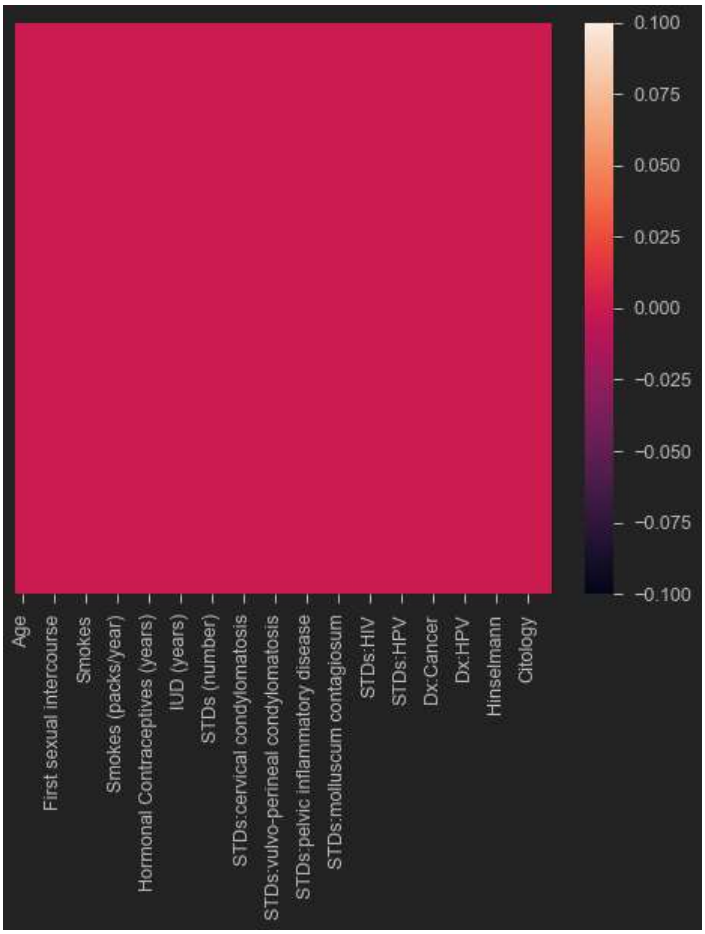
In [51]:
```
# Replace null values with mean
cancer_df = cancer_df.fillna(cancer_df.mean())
cancer_df
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STDs:
0	18	4.0	15.0000	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
1	15	1.0	14.0000	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
2	34	1.0	16.9953	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
3	52	5.0	16.0000	4.0	1.0	37.0	37.0	1.0	3.00	0.0	...	0.0
4	46	3.0	21.0000	4.0	0.0	0.0	0.0	1.0	15.00	0.0	...	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0000	0.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
854	32	2.0	19.0000	1.0	0.0	0.0	0.0	1.0	8.00	0.0	...	0.0
855	25	2.0	17.0000	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0
856	33	2.0	24.0000	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0
857	29	2.0	20.0000	1.0	0.0	0.0	0.0	1.0	0.50	0.0	...	0.0

858 rows × 34 columns

In [52]:
```
# Nan heatmap
sns.heatmap(cancer_df.isnull(), yticklabels=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2021ab99b38>
```



MINI CHALLENGE #2:

- What is the age range of people involved in this study?
- What are the biopsy results for the oldest person in this study?

In [57]:
```
print('Range of Age:', cancer_df['Age'].min(), cancer_df['Age'].max())
```

```
Range of Age: 13 84
```

In [58]:
```
cancer_df[cancer_df['Age']==84]
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STDs:
668	84	3.0	20.0	11.0	1.0	24.0	0.513202	0.0	0.0	0.0	...	0.0

1 rows × 34 columns

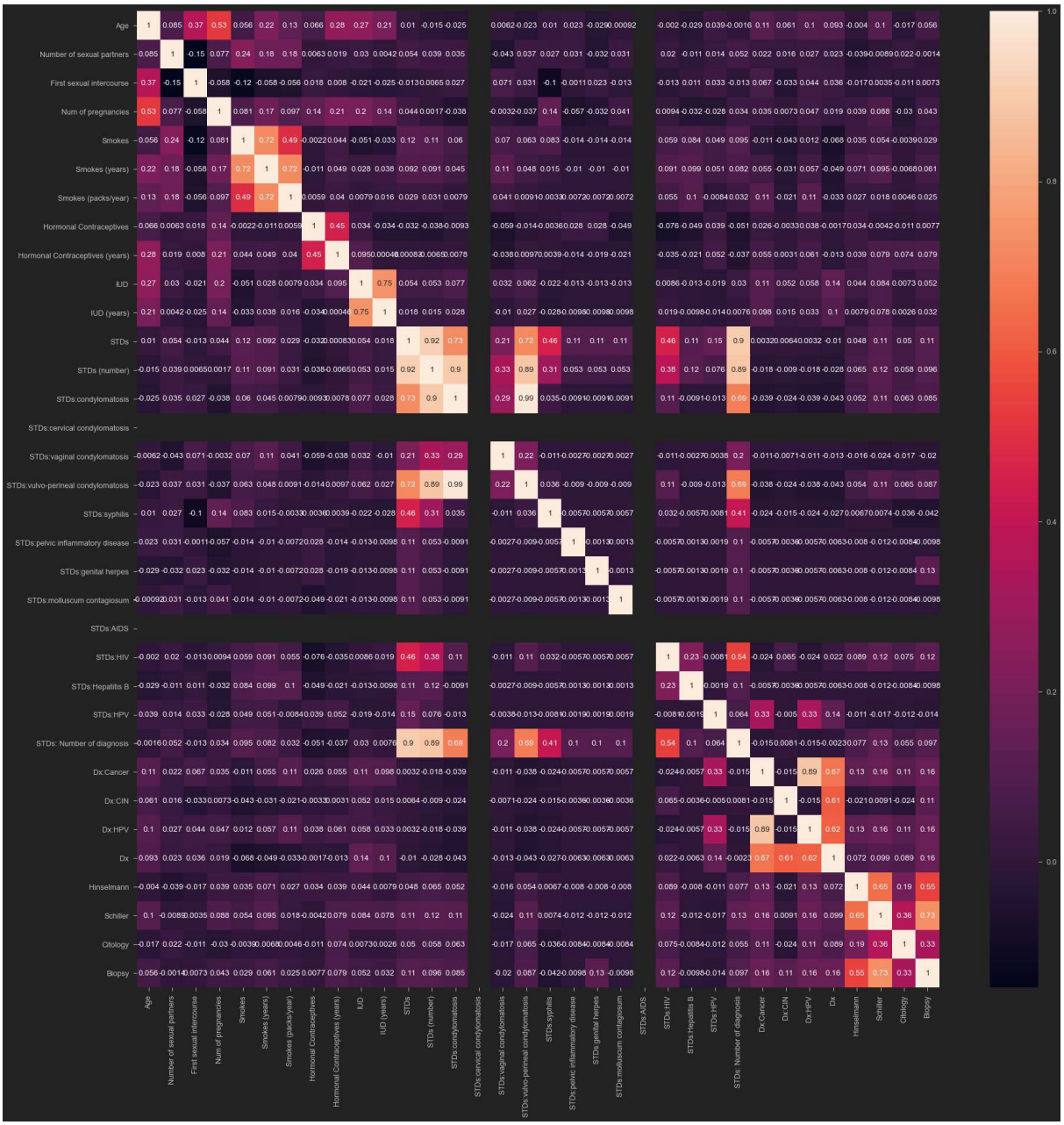
TASK #4: PERFORM DATA VISUALIZATION

In [59]:
```
# Get the correlation matrix
corr_matrix = cancer_df.corr()
corr_matrix
```

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Horm Contracept (ye
Age	1.000000	0.084896	0.369168	0.526137	0.055813	0.217349	0.131180	0.065624	0.277181
Number of sexual partners	0.084896	1.000000	-0.147937	0.076719	0.238078	0.177117	0.175153	0.006342	0.018552
First sexual intercourse	0.369168	-0.147937	1.000000	-0.058223	-0.123602	-0.058366	-0.056332	0.018344	0.008000
Num of pregnancies	0.526137	0.076719	-0.058223	1.000000	0.080768	0.174912	0.097044	0.142858	0.207839
Smokes	0.055813	0.238078	-0.123602	0.080768	1.000000	0.723128	0.493361	-0.002165	0.044157
Smokes (years)	0.217349	0.177117	-0.058366	0.174912	0.723128	1.000000	0.724116	-0.011002	0.048899
Smokes (packs/year)	0.131180	0.175153	-0.056332	0.097044	0.493361	0.724116	1.000000	0.005880	0.040112
Hormonal Contraceptives	0.065624	0.006342	0.018344	0.142858	-0.002165	-0.011002	0.005880	1.000000	0.448574
Hormonal Contraceptives (years)	0.277181	0.018552	0.008000	0.207839	0.044157	0.048899	0.040112	0.448574	1.000000
IUD	0.267662	0.030005	-0.020975	0.198550	-0.051184	0.027562	0.007891	0.033729	0.094953
IUD (years)	0.205886	0.004215	-0.024803	0.143642	-0.032996	0.037900	0.015912	-0.033752	0.000455
STDs	0.010017	0.053754	-0.013133	0.044250	0.116676	0.091611	0.029372	-0.032105	0.000829
STDs (number)	-0.015488	0.039359	0.006487	0.001706	0.105811	0.091313	0.030780	-0.038088	-0.006468
STDs:condylomatosis	-0.025012	0.034646	0.026777	-0.037999	0.059919	0.045397	0.007917	-0.009284	0.007752
STDs:cervical condylomatosis	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
STDs:vaginal condylomatosis	0.006220	-0.042924	0.071425	-0.003166	0.069631	0.114332	0.041412	-0.059222	-0.038207
STDs:vulvo-perineal condylomatosis	-0.022614	0.036750	0.031082	-0.037204	0.062775	0.047511	0.009130	-0.013714	0.009685
STDs:syphilis	0.010442	0.027178	-0.100999	0.141728	0.082684	0.015393	-0.003277	-0.003624	0.003897
STDs:pelvic inflammatory disease	0.023216	0.030616	-0.001089	-0.056542	-0.014059	-0.010337	-0.007180	0.027587	-0.014209
STDs:genital herpes	-0.029076	-0.031826	0.023398	-0.032114	-0.014059	-0.010337	-0.007180	0.027587	-0.019065
STDs:molluscum contagiosum	-0.000919	0.030616	-0.013332	0.041168	-0.014059	-0.010337	-0.007180	-0.048598	-0.021494
STDs:AIDS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
STDs:HIV	-0.002025	0.019871	-0.013430	0.009384	0.059412	0.090636	0.054577	-0.076278	-0.035472
STDs:Hepatitis B	-0.029076	-0.011012	0.011154	-0.032114	0.083551	0.099170	0.101105	-0.048598	-0.021494
STDs:HPV	0.038546	0.013871	0.033112	-0.028162	0.049171	0.050935	-0.008410	0.039040	0.052059
STDs: Number of diagnosis	-0.001606	0.051559	-0.013327	0.033514	0.095433	0.081676	0.032186	-0.050660	-0.037219
Dx:Cancer	0.110340	0.022309	0.067283	0.035123	-0.011027	0.054674	0.108476	0.026407	0.054627
Dx:CIN	0.061443	0.015691	-0.032626	0.007344	-0.042822	-0.030966	-0.021127	-0.003334	0.003086
Dx:HPV	0.101722	0.027264	0.043966	0.046753	0.012210	0.057214	0.110366	0.038038	0.061394
Dx	0.092635	0.022982	0.035750	0.019025	-0.067614	-0.048894	-0.033358	-0.001723	-0.012865
Hinselmann	-0.003967	-0.039273	-0.016546	0.038685	0.034527	0.071232	0.026662	0.033551	0.038825
Schiller	0.103283	-0.008899	0.003493	0.087687	0.053613	0.094640	0.017954	-0.004247	0.078707
Citology	-0.016862	0.021839	-0.010971	-0.029656	-0.003913	-0.006750	0.004613	-0.011030	0.074324
Biopsy	0.055956	-0.001429	0.007262	0.043460	0.029091	0.061484	0.024657	0.007711	0.078995

34 rows × 34 columns

In [60]:
```
# Plot the correlation matrix
plt.figure(figsize = (30,30))
sns.heatmap(corr_matrix, annot =True)
plt.show()
```



In [ ]:

MINI CHALLENGE #3:

- Plot the histogram for the entire DataFrame

In [62]:
```
cancer_df.hist(bins = 10, figsize = (30,30), color = 'b')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000020218562400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021E4904A8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021E471668>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021BDCA8D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021BDFBB38>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021BE30DA0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002021C176FD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C1B8278>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C1B82E8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C21F780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C2529E8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C284C50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002021C2BCEB8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C2FC160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C3303C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C363630>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C398898>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C3CCB00>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002021C400D68>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C436FD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C476278>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C4AD080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C4DE400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C510780>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002021C542B00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C576E80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C5B6240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C5E85C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C619940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C64CCC0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002021C68B048>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C6BE400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C6F0780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C721B00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C756E80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002021C795240>]],
      dtype=object)
```

TASK #5: PREPARE THE DATA BEFORE TRAINING

In [63]:
```
cancer_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:
0	18	4.0	15.0000	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
1	15	1.0	14.0000	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
2	34	1.0	16.9953	1.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
3	52	5.0	16.0000	4.0	1.0	37.0	37.0	1.0	3.00	0.0	...	0.0
4	46	3.0	21.0000	4.0	0.0	0.0	0.0	1.0	15.00	0.0	...	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0000	0.0	0.0	0.0	0.0	0.0	0.00	0.0	...	0.0
854	32	2.0	19.0000	1.0	0.0	0.0	0.0	1.0	8.00	0.0	...	0.0
855	25	2.0	17.0000	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0
856	33	2.0	24.0000	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	0.0
857	29	2.0	20.0000	1.0	0.0	0.0	0.0	1.0	0.50	0.0	...	0.0

858 rows × 34 columns

In [64]:
```
target_df = cancer_df['Biopsy']
input_df = cancer_df.drop(columns = ['Biopsy'])
```

In [66]:
```
target_df.shape
```

```
(858,)
```

In [67]:
```
input_df.shape
```

```
(858, 33)
```

In [68]:
```
X = np.array(input_df).astype('float32')
y = np.array(target_df).astype('float32')
```

In [69]:
```
# reshaping the array from (421570,) to (421570, 1)
# y = y.reshape(-1,1)
y.shape
```

```
(858,)
```

In [70]:
```
# scaling the data before feeding the model
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [71]:
```
X
```

```
array([[-1.0385634e+00,  8.9706147e-01, -7.1509570e-01, ...,
        -2.0622157e-01, -3.0722591e-01, -2.3249528e-01],
       [-1.3917956e+00, -9.3074709e-01, -1.0734857e+00, ...,
        -2.0622157e-01, -3.0722591e-01, -2.3249528e-01],
       [ 8.4534228e-01, -9.3074709e-01,  2.2945171e-07, ...,
        -2.0622157e-01, -3.0722591e-01, -2.3249528e-01],
       ...,
       [-2.1435465e-01, -3.2147753e-01,  1.6845580e-03, ...,
        -2.0622157e-01, -3.0722591e-01,  4.3011627e+00],
       [ 7.2759819e-01, -3.2147753e-01,  2.5104153e+00, ...,
        -2.0622157e-01, -3.0722591e-01, -2.3249528e-01],
       [ 2.5662178e-01, -3.2147753e-01,  1.0768549e+00, ...,
        -2.0622157e-01, -3.0722591e-01, -2.3249528e-01]], dtype=float32)
```

In [72]:
```
# spliting the data in to test and train sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size = 0.5)
```

MINI CHALLENGE #4:

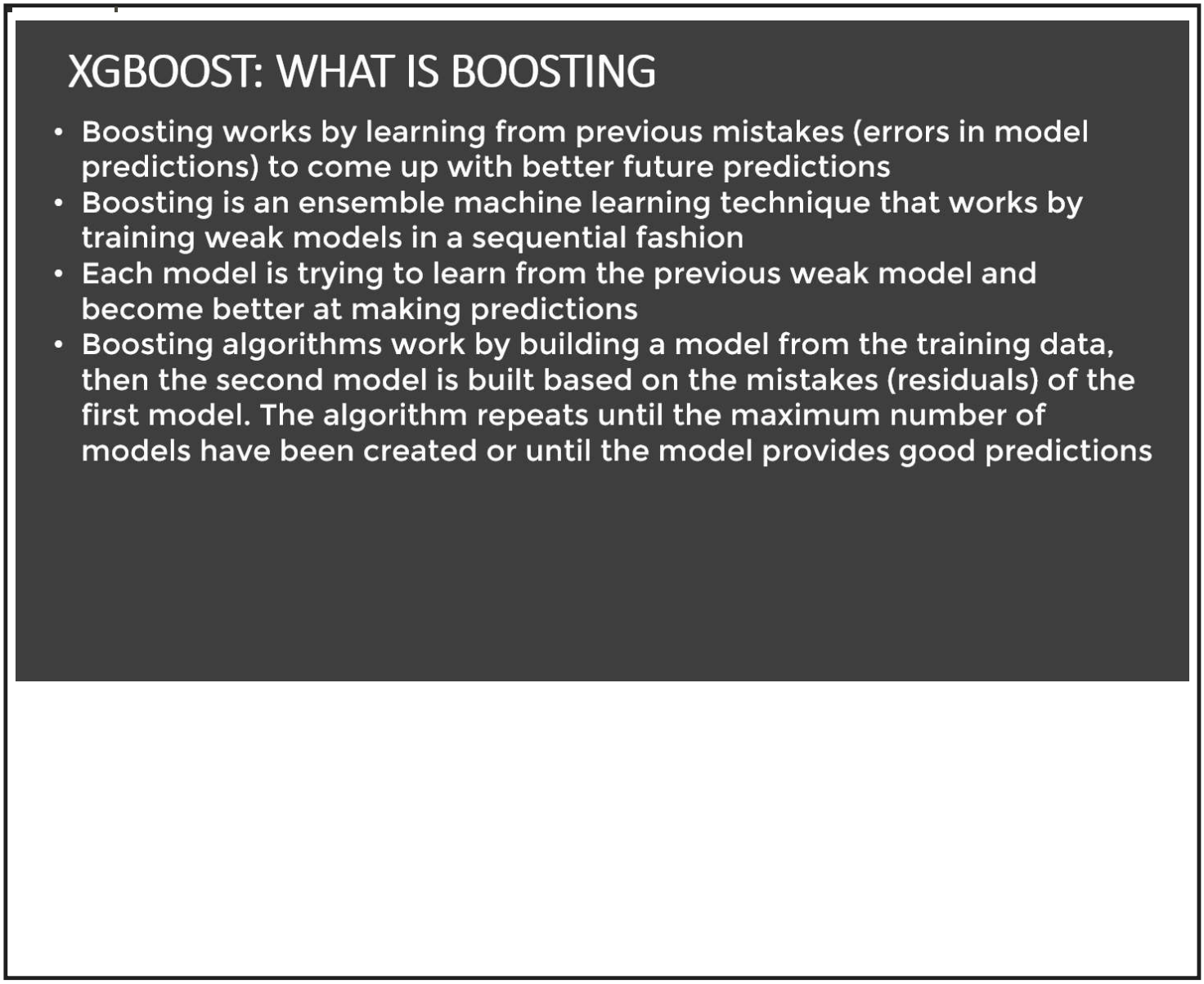
- Split the data such that the testing data is quarter the size of the training data

In [73]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

TASK #6: UNDERSTAND THE THEORY AND INTUITION BEHIND XG-BOOST ALGORITHM

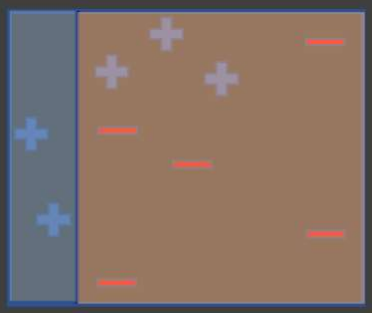
XGBOOST: INTRODUCTION

- XGBoost or Extreme Gradient Boosting is the algorithm of choice for many data scientists and could be used for regression and classification tasks
- XGBoost is a supervised learning algorithm and implements gradient boosted trees algorithm.
- The algorithm work by combining an ensemble of predictions from several weak models.
- It is robust to many data distributions and relationships and offers many hyperparameters to tune model performance.
- XGBoost offers increased speed and enhanced memory utilization.
- XGBoost is analogous to the idea of “discovering truth by building on previous discoveries”.


XGBOOST: WHAT IS BOOSTING
• Boosting works by learning from previous mistakes (errors in model predictions) to come up with better future predictions
• Boosting is an ensemble machine learning technique that works by training weak models in a sequential fashion
• Each model is trying to learn from the previous weak model and become better at making predictions
• Boosting algorithms work by building a model from the training data, then the second model is built based on the mistakes (residuals) of the first model. The algorithm repeats until the maximum number of models have been created or until the model provides good predictions

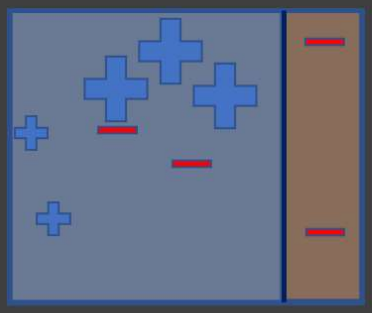
XGBOOST: BOOSTING EXAMPLE

- Model #1 works by attempting to classify the two classes (+) and (-) with the vertical line shown.
- Model #1 has assigned equal weights to all data points since it has no prior knowledge or experience from before.
- Model #1 misclassified 3 (+) samples.



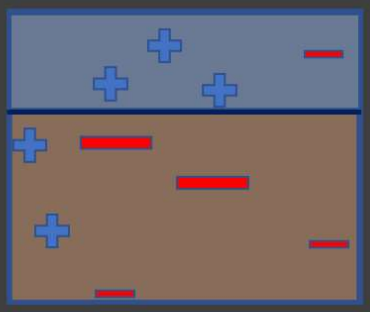
XGBOOST: BOOSTING EXAMPLE

- Model #2 learns from the mistakes of the previous model and assigns more weight to the wrongly classified data points (3 +) as shown in the figure below.
- So model #2 draws a vertical separating line and “made sure” to properly classify these points this time!
- The model did a great job correctly classifying points with higher weights but in the process, it has misclassified two red (-) samples.



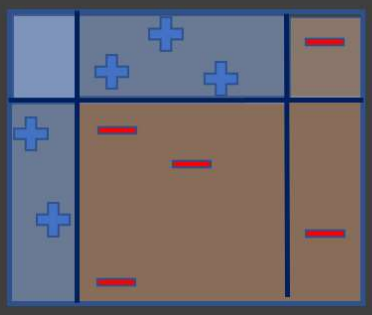
XGBOOST: BOOSTING EXAMPLE

- Model #3 learns from the mistakes of the previous model and assigns more weight to the wrongly classified data points (2 -) as shown in the figure below.
- So model #3 draws a horizontal separating line and “made sure” to properly classify these points this time!
- The model did a great job correctly classifying points with higher weights but in the process, it has misclassified two blue (+) samples.



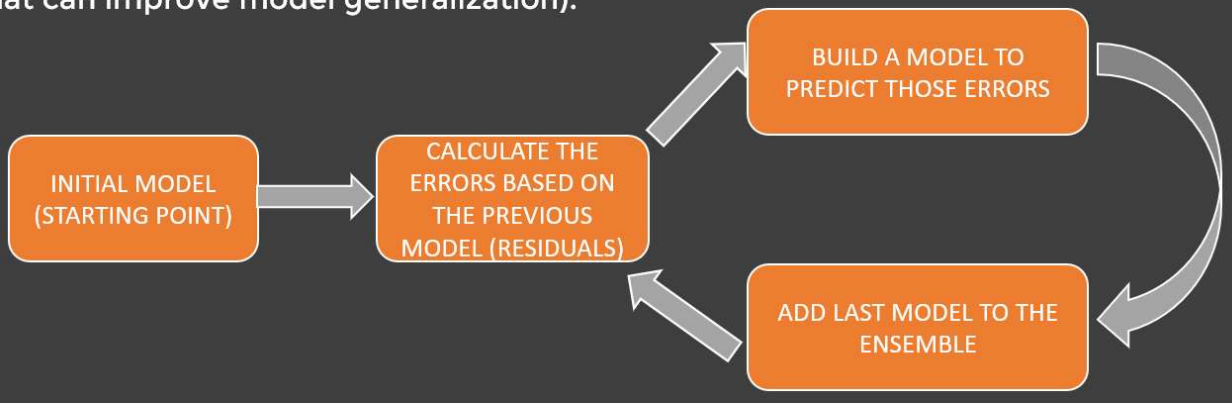
XGBOOST: BOOSTING EXAMPLE

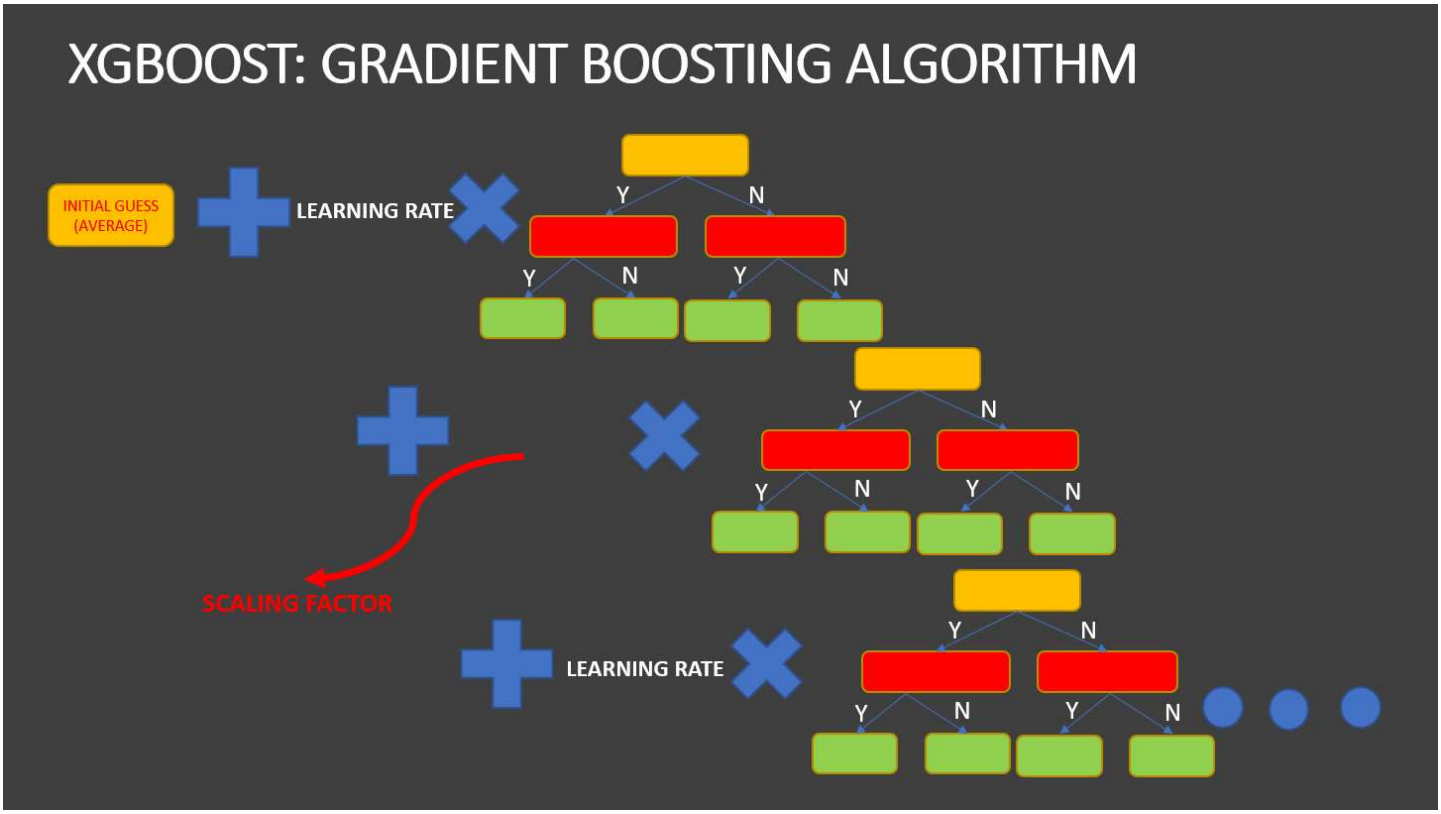
- Model #4 combines all the mistakes from all these weak models to build a much stronger model that correctly classifies all data points.



XGBOOST: STEPS

- XGBoost repeatedly builds new models and combine them into an ensemble model
- Initially build the first model and calculate the error for each observation in the dataset
- Then you build a new model to predict those residuals (errors)
- Then you add prediction from this model to the ensemble of models
- XGBoost is superior compared to gradient boosting algorithm since it offers a good balance between bias and variance (Gradient boosting only optimized for the variance so tend to overfit training data while XGBoost offers regularization terms that can improve model generalization).




XGBOOST: GRADIENT BOOSTING ALGORITHM
INITIAL GUESS (AVERAGE)
LEARNING RATE
Y
N
SCALING FACTOR
LEARNING RATE

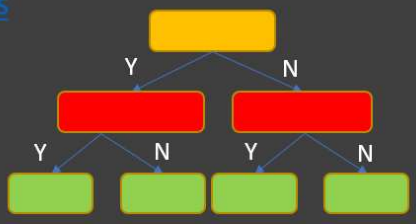
XGBOOST: GRADIENT BOOSTING ALGORITHM

- Gradient boost works by building a tree based on the error (residuals) from the previous tree.
- Gradient boost scales the trees and then adds the predictions from the new tree to the predictions from previous trees.
- Example adopted from the awesome StatQuest (by Josh Starmer): https://www.youtube.com/watch?v=3CC4N4z3GJc&t=87s

Height	Color	Gender	Weight (Kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

INPUT FEATURES

VARIABLE TO BE PREDICTED



XGBOOST: GRADIENT BOOSTING ALGORITHM

- Let's assume that the initial model predictions (starting point) is the average weight is 71.2
- Gradient boost builds a tree based on the error from the first tree.
- The tree is built by assuming that the features (heights, color, and gender) predicts the residuals (new column that we have just created).

71.2

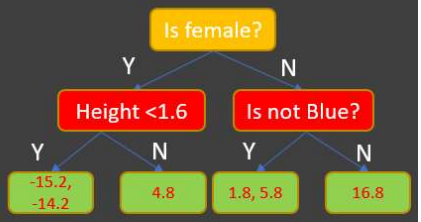
INITIAL STARTING POINT (PREDICTIONS)

Height	Color	Gender	Weight (Kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

INPUT FEATURES

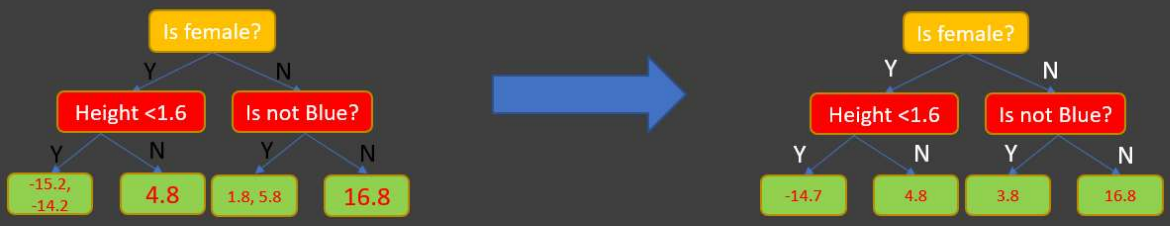
Error = True – predicted
88-71.2=16.8
76-71.2=4.8
56-71.2=-15.2
73-71.2=1.8
77-71.2=5.8
57-71.2=-14.2

ERRORS (RESIDUALS)



XGBOOST: GRADIENT BOOSTING ALGORITHM

- Note that the number of leaves is restricted to 4 in this example for the sake of simplicity.
- Let's replace the values with the average a shown below.



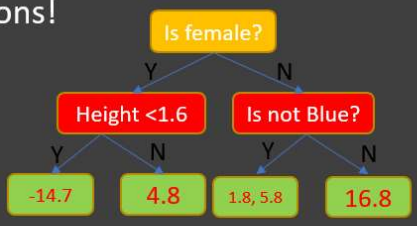
$$Average_1 = (-15.2 - 14.2)/2 = -14.7$$
$$Average_2 = (1.8 + 5.8)/2 = 3.8$$

XGBOOST: GRADIENT BOOSTING ALGORITHM

- Now that we have built a tree, let's combine the previous predictions with the new tree to generate new predictions!

71.2

AVERAGE WEIGHT (INITIAL GUESS FROM PREVIOUS STEP



Height	Color	Gender	Weight (Kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

INPUT FEATURES

New predictions
88

The model predictions match the true weight. This indicates that the model is overfitting the training data

Predictions = 71.2+16.8=88

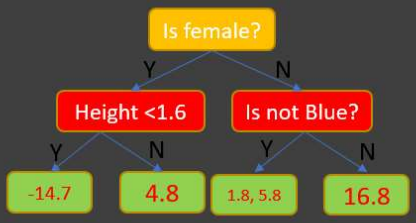
XGBOOST: GRADIENT BOOSTING ALGORITHM

- We add a learning rate (range from 0 to 1) to overcome this issue.
- This parameter is used for scaling purposes by adjusting the newly added information from the new tree.
- Adding this tree and scaling it with the learning rate helps us get a little closer to the true values.
- By taking smaller steps, the model results in better predictions on the testing dataset (low variance).

71.2

AVERAGE WEIGHT (INITIAL GUESS FROM PREVIOUS STEP

LEARNING RATE
0.1



Height	Color	Gender	Weight (Kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

New predictions
72.9

Predictions = 71.2+0.1*16.8=72.9

XGBOOST: GRADIENT BOOSTING ALGORITHM

- Now let's build another tree with the new residuals from the new predictions.

Height	Color	Gender	Weight (Kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

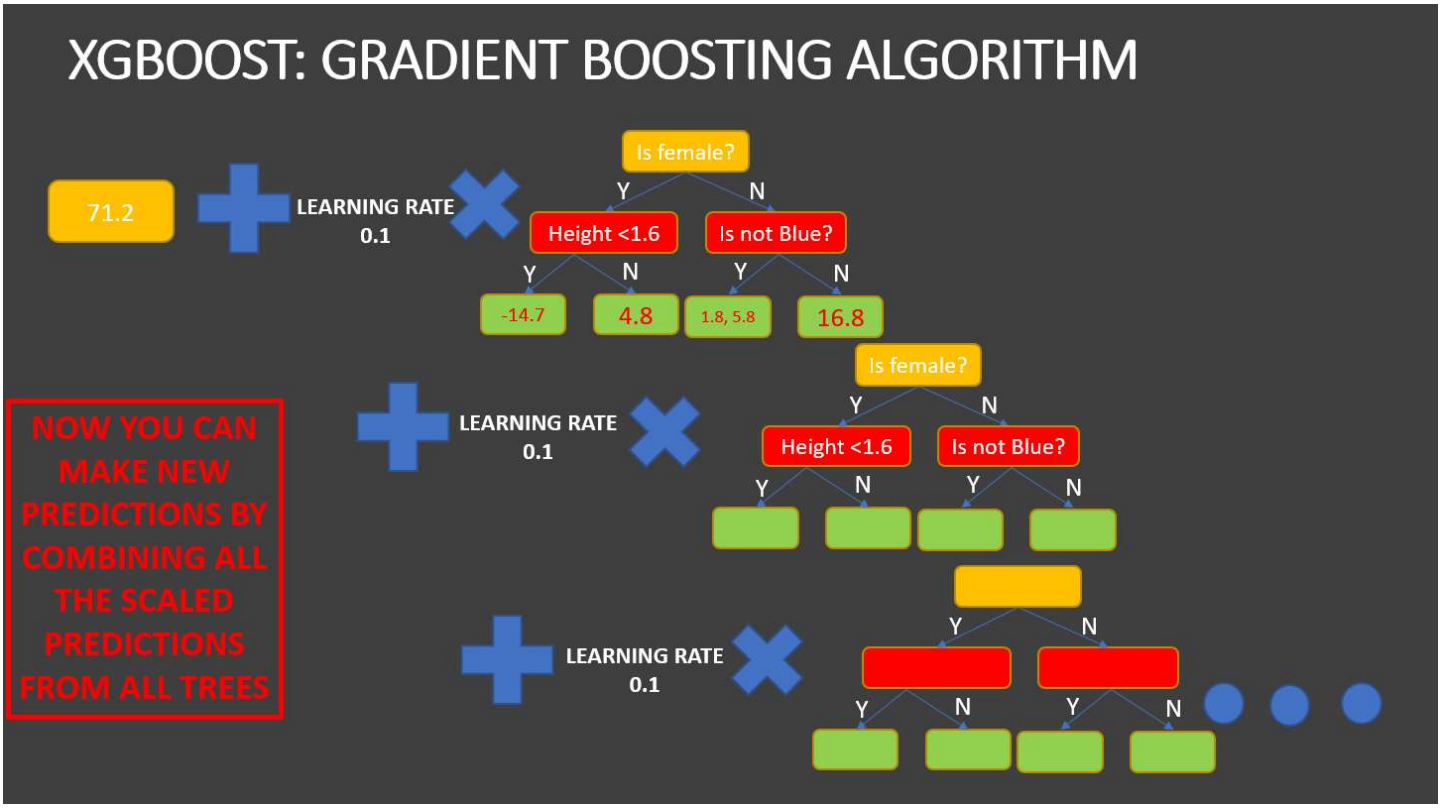
RECALL THAT THESE ARE THE INITIAL RESIDUALS

Initial Residuals
16.8
4.8
-15.2
1.8
5.8
-14.2

Predictions = 71.2+ 0.1 * 16.8=72.9

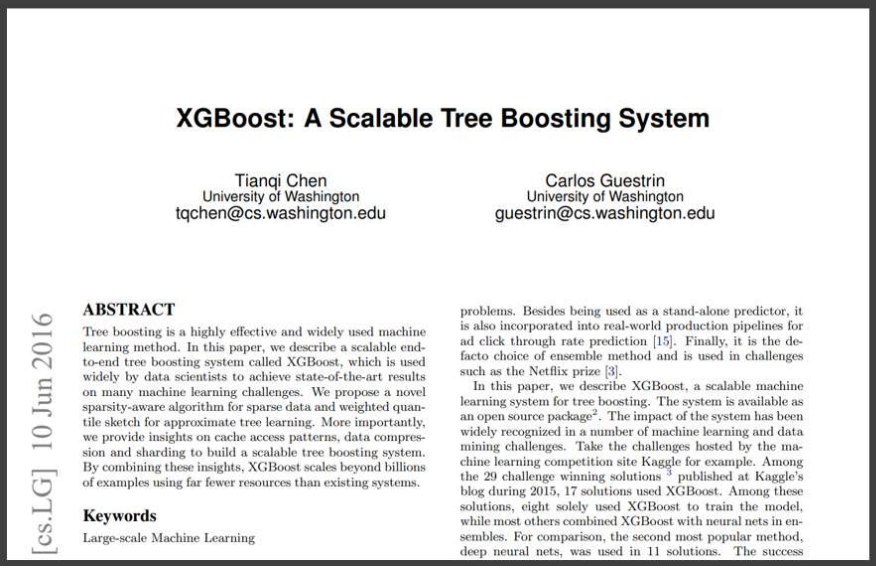
New predictions	New Residuals
72.9	88-72.9 = 15.1
	4.3
	-13.7
	1.4
	5.4
	-12.7

RESIDUALS HAVE GONE DOWN!


XGBOOST: GRADIENT BOOSTING ALGORITHM
71.2
LEARNING RATE
0.1
Is female?
Y
N
Height <1.6
Is not Blue?
Y
N
Y
N
-14.7
4.8
1.8, 5.8
16.8
Is female?
Y
N
Height <1.6
Is not Blue?
Y
N
Y
N
LEARNING RATE
0.1
LEARNING RATE
0.1
NOW YOU CAN MAKE NEW PREDICTIONS BY COMBINING ALL THE SCALED PREDICTIONS FROM ALL TREES

XGBOOST: WHAT IS THE EXTREME GRADIENT BOOSTING THEN? GREAT RESOURCES

Paper: https://arxiv.org/pdf/1603.02754.pdf
https://xgboost.readthedocs.io/en/latest/tutorials/model.html



XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT

Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Keywords

Large-scale Machine Learning

problems. Besides being used as a stand-alone predictor, it is also incorporated into real-world production pipelines for ad click through rate prediction [15]. Finally, it is the de-facto choice of ensemble method and is used in challenges such as the Netflix prize [3].

In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success

TASK #7: TRAIN AND EVALUATE XGBOOST CLASSIFIER

In [74]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages (1.3.0.post0)
Requirement already satisfied: numpy in c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages (from xgboost) (1.
18.2)
Requirement already satisfied: scipy in c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages (from xgboost) (1.
4.1)


tensorboard 2.1.1 has requirement setuptools>=41.0.0, but you'll have setuptools 39.0.1 which is incompatible.
google-auth 1.12.0 has requirement setuptools>=40.3.0, but you'll have setuptools 39.0.1 which is incompatible.
You are using pip version 10.0.1, however version 24.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

In [75]:
```
# Train an XGBoost classifier model
import xgboost as xgb

model = xgb.XGBClassifier(learning_rate = 0.1, max_depth = 5, n_estimators = 10)

model.fit(X_train, y_train)
```

```
[22:24:55] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the defau
lt evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd
like to restore the old behavior.


c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label enco
der in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_l
abel_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...,
[num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=10, n_jobs=2, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [76]:
```
result_train = model.score(X_train, y_train)

result_train
```

```
0.9766718506998445
```

In [77]:
```
# predict the score of the trained model using the testing dataset
result_test = model.score(X_test, y_test)

result_test
```

```
0.9674418604651163
```

In [78]:
```
# make predictions on the test data
y_predict = model.predict(X_test)

y_predict
```

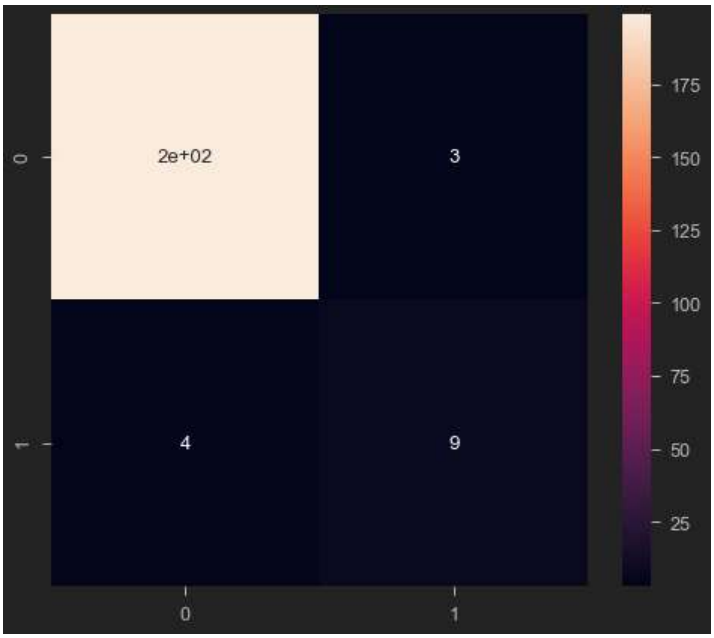
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 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., 1., 0., 0., 0., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

In [79]:
```
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.98       203
         1.0       0.69      0.75      0.72        12

    accuracy                           0.97       215
   macro avg       0.84      0.87      0.85       215
weighted avg       0.97      0.97      0.97       215
```

In [80]:
```
cm = confusion_matrix(y_predict, y_test)
sns.heatmap(cm, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2021cf26080>
```



MINI CHALLENGE #5:

- Retrain the model with 10x and 100x the number of estimators and tree depth
- Plot the confusion matrix
- Comment on the performance of the model

In [81]:
```
model1 = xgb.XGBClassifier(learning_rate = 0.1, max_depth = 50, n_estimators = 100)
model2 = xgb.XGBClassifier(learning_rate = 0.1, max_depth = 500, n_estimators = 1000)

model1.fit(X_train, y_train)
```

```
[22:37:06] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the defau
lt evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd
like to restore the old behavior.


XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=50,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=2, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [82]:
```
model2.fit(X_train, y_train)
```

```
[22:37:43] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the defau
lt evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd
like to restore the old behavior.


XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=500,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=1000, n_jobs=2, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [83]:
```
result_train1 = model1.score(X_train, y_train)
result_train2 = model2.score(X_train, y_train)
print(result_train1, result_train2)
```

```
0.9968895800933126 1.0
```

In [84]:
```
result_test1 = model1.score(X_test, y_test)
result_test2 = model2.score(X_test, y_test)
print(result_test1, result_test2)
```

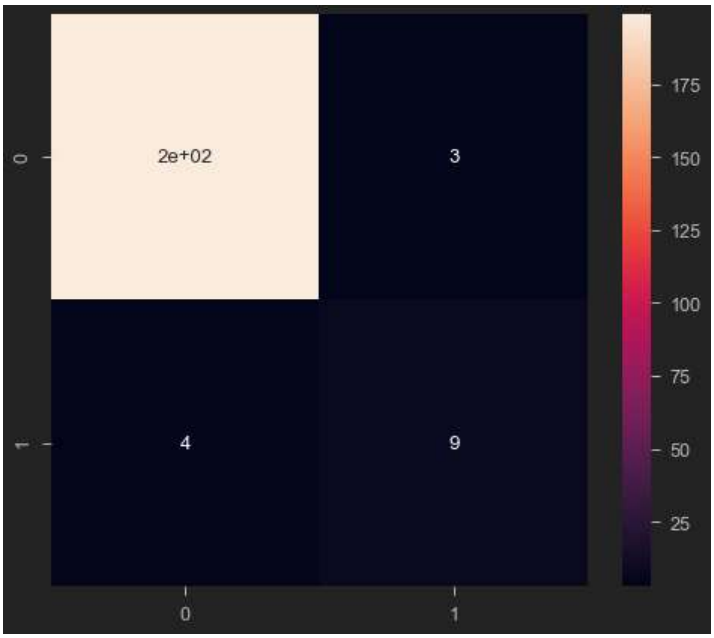
```
0.9720930232558139 0.9627906976744186
```

In [86]:
```
y_predict1 = model1.predict(X_test)
y_predict2 = model2.predict(X_test)
```

In [87]:
```
print(classification_report(y_test, y_predict1))

cm1 = confusion_matrix(y_predict1, y_test)
sns.heatmap(cm, annot = True)
```

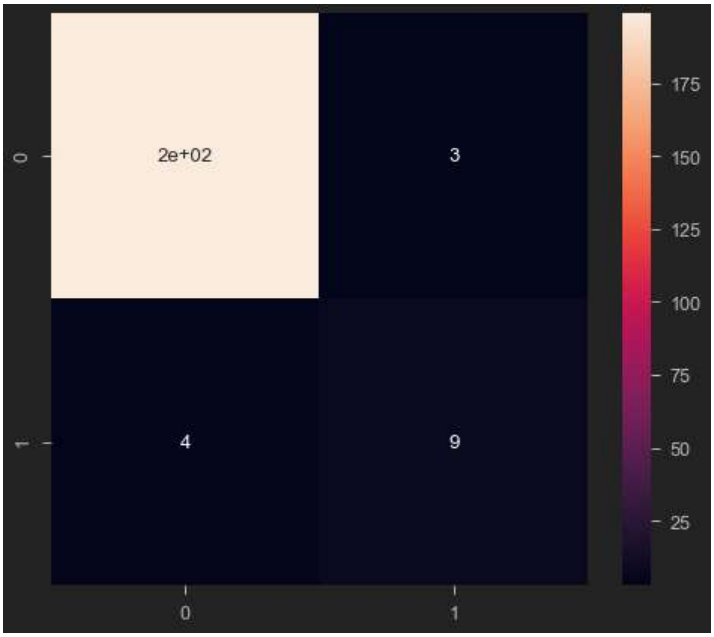
```
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.99       203
         1.0       0.71      0.83      0.77        12

    accuracy                           0.97       215
   macro avg       0.85      0.91      0.88       215
weighted avg       0.97      0.97      0.97       215
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2021e8b9ba8>
```



In [88]:
```
print(classification_report(y_test, y_predict2))

cm1 = confusion_matrix(y_predict2, y_test)
sns.heatmap(cm, annot = True)
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.98       203
         1.0       0.64      0.75      0.69        12

    accuracy                           0.96       215
   macro avg       0.81      0.86      0.84       215
weighted avg       0.97      0.96      0.96       215


<matplotlib.axes._subplots.AxesSubplot at 0x20220678400>
```



EXCELLENT JOB!

MINI CHALLENGE #1 SOLUTION:

- Print the last 20 rows in the dataframe

In [ ]:
```
cancer_df.tail(20)
```

MINI CHALLENGE #2 SOLUTION:

- What is the age range of people involved in this study?
- What are the biopsy results for the oldest person in this study?

In [ ]:
```
cancer_df['Age'].max()
```

In [ ]:
```
cancer_df['Age'].min()
```

In [ ]:
```
cancer_df[cancer_df['Age']==84]
```

MINI CHALLENGE #3 SOLUTION:

- Plot the histogram for the entire DataFrame. Set the number of bins to 10.

In [ ]:
```
# Plot the histogram
cancer_df.hist(bins = 10, figsize = (30,30), color = 'b');
```

MINI CHALLENGE #4 SOLUTION:

- Split the data such that the testing data is quarter the size of the training data

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

MINI CHALLENGE #5 SOLUTION:

- Retrain the model with 10x and 100x the number of estimators and tree depth
- Plot the confusion matrix
- Comment on the performance of the model

In [ ]:
```
model = xgb.XGBClassifier(learning_rate = 0.1, max_depth = 50, n_estimators = 100)
```

In [ ]:
```
model.fit(X_train, y_train)
```

In [ ]:
```
result_train = model.score(X_train, y_train)
print("Accuracy : {}".format(result_train))
```

In [ ]:
```
# predict the score of the trained model using the testing dataset
result = model.score(X_test, y_test)
print("Accuracy : {}".format(result))
```

In [ ]:
```
# make predictions on the test data
y_predict = model.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_predict))
```

In [ ]:
```
plt.figure(figsize=(10, 10))
cm = confusion_matrix(y_predict, y_test)
sns.heatmap(cm, annot = True,fmt = '.2f')
plt.ylabel('Predicted class')
plt.xlabel('Actual class')
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```