AN ANALYSIS OF CERVICAL CANCER RISK FACTORS USING PYSPARK MODELLING AND TABLEAU VISUALISATION

Student Name: Chiamaka Onwuka
Student ID : 12005949
Module Name: Big Data Management and Data Visualisation
Module Leader: Dr. Marwan Fuad

I can confirm that all work submitted is my own: Yes

ABSTRACT

This paper examines cervical cancer risk factors that influence the need for a screening. It uses PySpark for Big Data analysis and reliably compares three different ML classification algorithms - Logistics regression, Random Forest classifier and Naives Bayes - in predicting / calculating the likelihood of needing a cervical cancer biopsy. The project also attempts to identify observation clusters by applying K-means clustering on the data on age, age of first sexual experience, number of sexual partners, number of pregnancies, and Human papillomavirus (HPV). Tableau software is used for the data visualization to help understand the data better and make informed decisions.

Keywords – Cervical, Biopsy, Big Data, Machine Learning, PySpark, Random Forest, Tableau

Name of dataset: Cervical Cancer Risk Classification

Link: https://www.kaggle.com/datasets/loveall/cervical-cancer-risk-classification

CONTENT PAGE

1. INTRODUCTION 4

2. PYSPARK INSTALLATION & SET-UP 5

2.1 APACHE SPARK, JAVA PREREQUISITES AND PYSPARK INSTALLATION 5

3. THE DATASET 9

3.1.1. LOADING AND EXPLORING DATASET 10

3.1.2. CLEANING AND PREPARING DATASET 13

4. MACHINE LEARNING 20

4.1. VECTOR ASSEMBLER 21

4.2. DATA SCALING 22

4.3. LOGISTIC REGRESSION 23

4.4. NAÏVE BAYES 26

4.5. RANDOM FOREST 27

4.6. MODEL ACCURACY 29

4.7. K-MEANS CLUSTERING 29

5. DATA VISUALIZATION 34

6. DISCUSSION 43

7. CONCLUSION 43

8. REFERENCES 44

9. APPENDIX 45

1. INTRODUCTION

Cervical cancer is the fourth most common cancer in women and in 2018, according to the World Health Organisation, an estimated 570 000 women were diagnosed with cervical cancer globally and over 300 000 women died from the disease. Fortunately, identifying risk factors and early cervical screening/ biopsy are key to greatly reducing the mortality rate.

This project seeks to apply PySpark machine learning techniques to a cervical risk factor public Kaggle dataset to help predict the need for a biopsy and the use of Tableau software to visualize patterns. This objective is achieved by:

1. Setting up the PySpark environment to run on Jupyter Notebook
2. Dataset loading and processing
3. Using PySpark Classification Algorithms to predict Biopsy
4. Data Visualisation using Tableau
5. Reviewing Outcomes and Conclusion

2. PYSPARK INSTALLATION & SET-UP

2.1. APACHE SPARK, JAVA PREREQUISITES AND PYSPARK INSTALLATION

Using Chang Lee's guide to installing and running Pyspark on Jupyter Notebook on windows, the first step involved going to the Apache Spark download page to select the 3.2.1 release suitable for Apache Hadoop 3.3 and later. Then spark-3.2.1-bin-hadoop3.2tgz was downloaded.

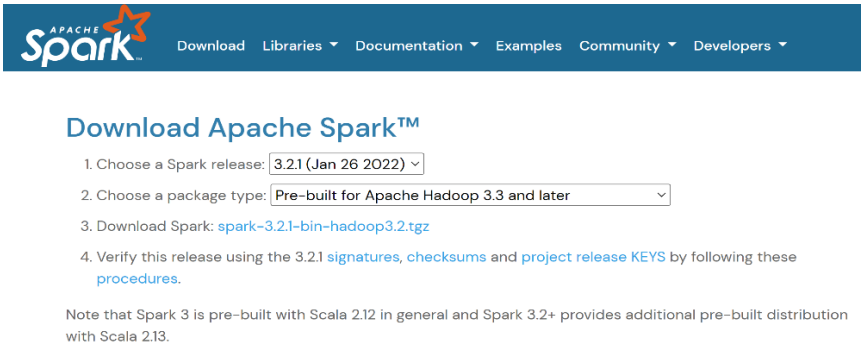


Figure 1. Spark Download

Jupyter Notebook from the Anaconda suite where the PySpark will be executed, had been previously installed on the machine.

Figure 2. Jupyter Notebook

Spark runs on Java, so windows command prompt was checked to verify if java was installed. From the screenshot below, java is not recognised, which means installation is necessary.

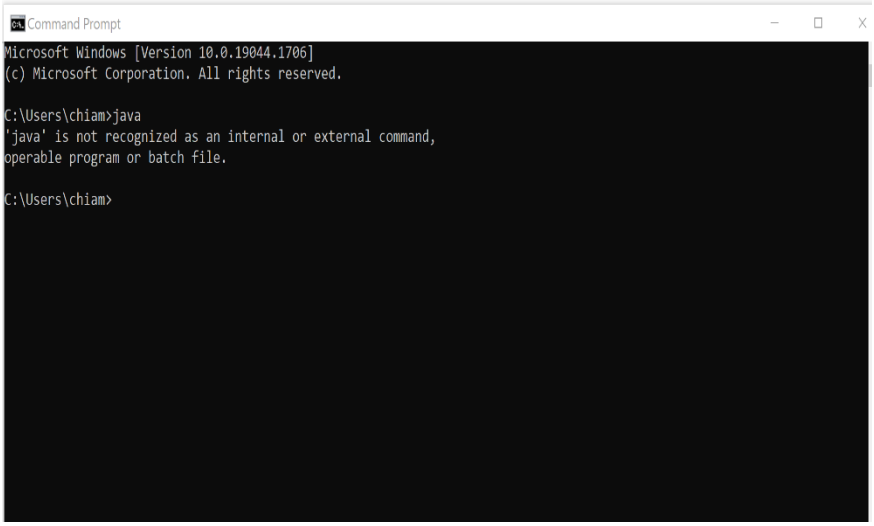
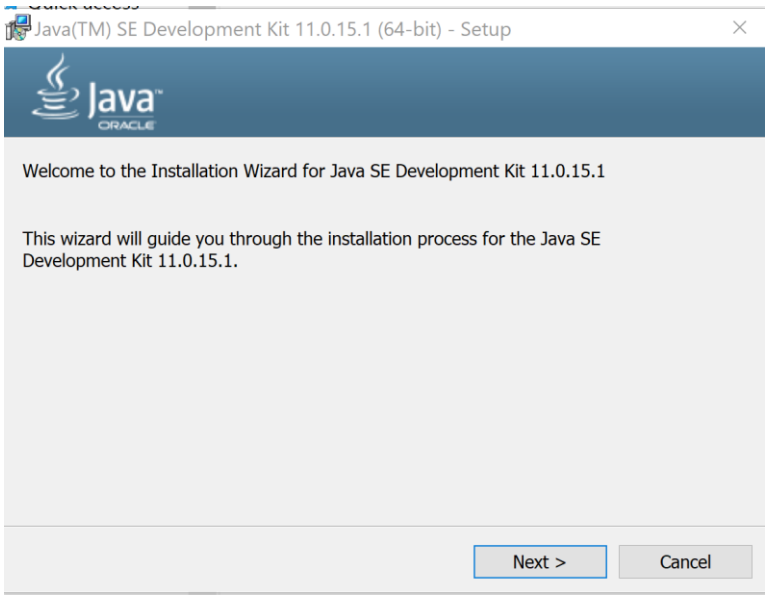


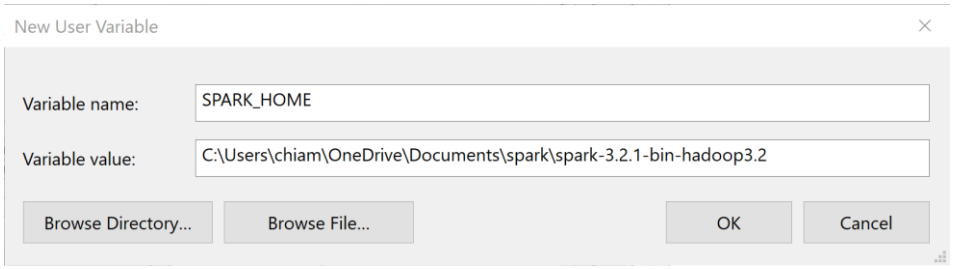
Figure 3a. Checking for Java

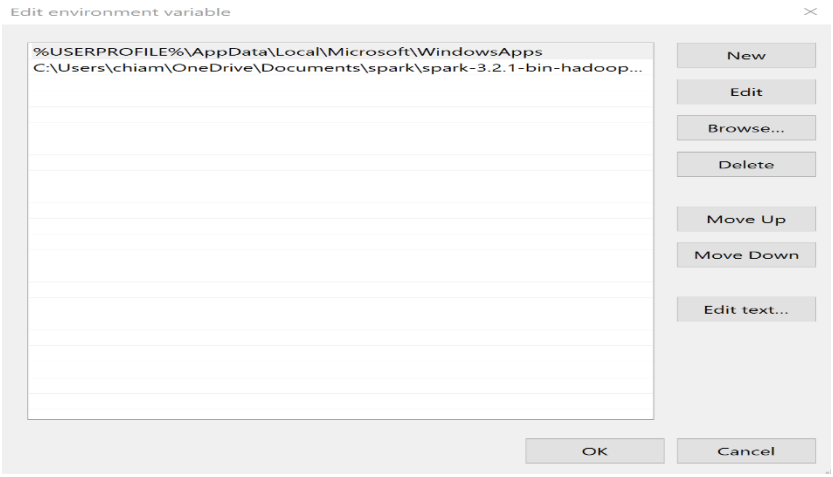
Java version 11 was downloaded and installed from oracle.

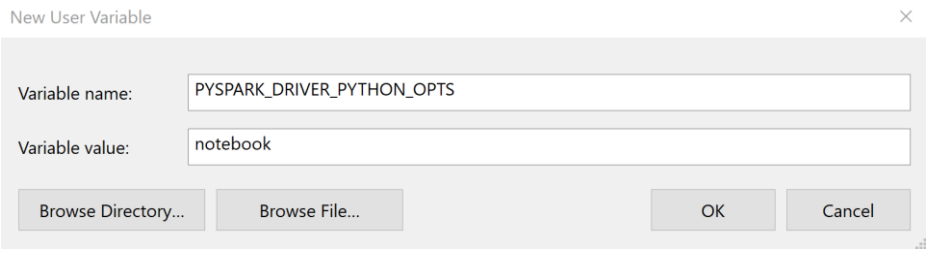


Java version was checked on the command prompt to verify that Java has been successfully installed.

Environment variables for Hadoop_Home and Spark_Home were added to let windows know the file location when the PySpark kernel is launched.







In Figure below, findspark was installed on the Anaconda Powershell Prompt

```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\chiam> pip install findspark
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
(base) PS C:\Users\chiam> pip install pyspark
Collecting pyspark
  Downloading pyspark-3.3.0.tar.gz (281.3 MB)
     |████████████████████████████████| 281.3 MB 73 kB/s
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     |████████████████████████████████| 199 kB 3.3 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.0-py2.py3-none-any.w
hl size=281764027 sha256=4874eea3fc4836c1fa573f36acd85ffb7342f556eb671
0afadffa86f9b3ea687
  Stored in directory: c:\users\chiam\appdata\local\pip\cache\wheels\0
5\75\73\81f84d174299abca38dd6a06a5b98b08ae25fce50ab8986fa1
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.0
(base) PS C:\Users\chiam>
```

A Jupyter notebook kernel was launched to test the codes. Relevant libraries were imported to run the PySpark session, and test that PySpark runs properly

```
In [1]: #Importing PySpark libraries
        import findspark
        findspark.init()
        from pyspark import SparkConf, SparkContext
        from pyspark.sql import SparkSession, SQLContext
        import pandas as pd
```

```
In [2]: spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
In [3]: #Test the spark
        df = spark.sql('''select 'chy' as sweetpea ''')
        df.show()

        +--------+
        |sweetpea|
        +--------+
        |     chy|
        +--------+
```

3. THE DATASET

This Kaggle Dataset was obtained from UCI Repository. It dataset comprises of personal details, medical records and other demographic information of 858 female patients. Some patients opted out of providing responses to some of the questions because of privacy concerns (missing values). This dataset focuses on the prediction of indicators/diagnosis of cervical cancer.

The following are the dependent attributes which predict the Biopsy (A cervical biopsy is a surgical procedure in which a small amount of tissue is removed from the cervix. A cervical biopsy is usually done after an abnormality has been found during cytology)
:

1. Age
2. Number of sexual partners
3. First sexual intercourse
4. Number of pregnancies.
5. Smokes
6. Smokes (years)
7. Smokes (packs/year)
8. Hormonal Contraceptives
9. Hormonal Contraceptives (years)
10. Intra-Uterine Device
11. IUD (years)
12. STDs
13. STDs:molluscum contagiosum
14. STDs:AIDS
15. STDs:HIV
16. STDs:Hepatitis B
17. STDs:HPV
18. STDs: Number of diagnosis
19. STDs: Time since first diagnosis
20. STDs: Time since last diagnosis
21. Dx:Cancer
22. Dx:CIN
23. Dx:HPV
24. Dx
25. Hinselmann
26. Schiller
27. Cytology

kag_risk_factors_cervical_cancer.csv (102.06 KiB)

Detail Compact Column 10 of 36 columns

About this file

Cervical Cancer Risk Factors for Biopsy

# Age	Number of sexual...	First sexual interc...	Num of pregnanci...	Smokes	Smoke
13 84	2.0 32% 3.0 24% Other (378) 44%	15.0 19% 17.0 18% Other (544) 63%	1.0 31% 2.0 28% Other (348) 41%	0.0 84% 1.0 14% Other (13) 2%	0.0 1.266972 Other (12
18	4.0	15.0	1.0	0.0	0.0
15	1.0	14.0	1.0	0.0	0.0
34	1.0	?	1.0	0.0	0.0
52	5.0	16.0	4.0	1.0	37.0
46	3.0	21.0	4.0	0.0	0.0
42	3.0	23.0	2.0	0.0	0.0
51	3.0	17.0	6.0	1.0	34.0

Figure 4. Sample of the Dataset

3.1.1. LOADING AND EXPLORING DATASET

Relevant PySpark libraries were imported. A new spark context and session was created and dataset loaded, as seen in the figures below.

```
In [4]: spark = SparkSession.builder.appName("Cervical Cancer Classification").getOrCreate()
```

Figure 5. SparkSession builder

In Figure 5 we created a spark session. It was named “Cervical Cancer Classification”, but we could have named it anything, as it is in our discretion how we name our session.

```
In [5]: #Load Dataset
        file_loc_csv = 'kag_risk_factors_cervical_cancer.csv'
        cervical = spark.read.csv(file_loc_csv, inferSchema=True, header=True)
```

Figure 6. Loading Dataset

We loaded our dataset as seen in *Figure-6 and stored it as “cervical”, we did that using PySpark, as we have to complete this project in PySpark exclusively. While loading our dataset we used “inferSchema=True” so that the system automatically understands the datatype of different columns and we set “header=True” and this way the system will take first row as the column names.*

The properties of the dataset such as type, number of rows and columns, description, etc were run, from which the details were confirmed as displayed on Kaggle(858 rows and 36 columns). For reference please check *Figure-7 to Figure-11*

```
In [6]: #Data type
        print(type(cervical))

        <class 'pyspark.sql.dataframe.DataFrame'>
```

Figure 7. Checking type of dataset

```
In [7]: #Check number of rows
        cervical.count()

Out[7]: 858
```

Figure 8. Checking number of rows

```
In [8]: #Check number of columns
        len(cervical.columns)

Out[8]: 36
```

Figure 9. Checking number of columns

```
In [11]: #Show details of the top 2 rows
         cervical.head(2)

Out[11]: [Row(Age=18, Number of sexual partners='4.0', First sexual intercourse='15.0', Num of pregnancies='1.0', Smokes='0.0', Smokes
         (years)='0.0', Smokes (packs/year)='0.0', Hormonal Contraceptives='0.0', Hormonal Contraceptives (years)='0.0', IUD='0.0', IUD
         (years)='0.0', STDs='0.0', STDs (number)='0.0', STDs:condylomatosis='0.0', STDs:cervical condylomatosis='0.0', STDs:vaginal con
         dylomatosis='0.0', STDs:vulvo-perineal condylomatosis='0.0', STDs:syphilis='0.0', STDs:pelvic inflammatory disease='0.0', STDs:
         genital herpes='0.0', STDs:molluscum contagiosum='0.0', STDs:AIDS='0.0', STDs:HIV='0.0', STDs:Hepatitis B='0.0', STDs:HPV='0.
         0', STDs: Number of diagnosis=0, STDs: Time since first diagnosis='?', STDs: Time since last diagnosis='?', Dx:Cancer=0, Dx:CIN
         =0, Dx:HPV=0, Dx=0, Hinselmann=0, Schiller=0, Citology=0, Biopsy=0),
          Row(Age=15, Number of sexual partners='1.0', First sexual intercourse='14.0', Num of pregnancies='1.0', Smokes='0.0', Smokes
         (years)='0.0', Smokes (packs/year)='0.0', Hormonal Contraceptives='0.0', Hormonal Contraceptives (years)='0.0', IUD='0.0', IUD
         (years)='0.0', STDs='0.0', STDs (number)='0.0', STDs:condylomatosis='0.0', STDs:cervical condylomatosis='0.0', STDs:vaginal con
         dylomatosis='0.0', STDs:vulvo-perineal condylomatosis='0.0', STDs:syphilis='0.0', STDs:pelvic inflammatory disease='0.0', STDs:
         genital herpes='0.0', STDs:molluscum contagiosum='0.0', STDs:AIDS='0.0', STDs:HIV='0.0', STDs:Hepatitis B='0.0', STDs:HPV='0.
         0', STDs: Number of diagnosis=0, STDs: Time since first diagnosis='?', STDs: Time since last diagnosis='?', Dx:Cancer=0, Dx:CIN
         =0, Dx:HPV=0, Dx=0, Hinselmann=0, Schiller=0, Citology=0, Biopsy=0)]
```

Figure 10. Showing details of the first 2 rows

```
In [12]: #Print top 5 rows
         cervical.show(5)
```

```
+---+-------------------------+-----------------------+------------------+------+--------------+-------------------+----------
-------------+-----------------------------+---+-----------+----+-------------+-------------------+-------------------------
--+---------------------------+-----------------------------------+-------------+-------------------------------+--------------
-----+-------------------------+---------+--------+----------------+--------+-----------------------+-----------------------
---------+-----------------------------+---------+------+------+---+----------+--------+--------+------+
|Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Smokes|Smokes (years)|Smokes (packs/year)|Hormonal C
ontraceptives|Hormonal Contraceptives (years)|IUD|IUD (years)|STDs|STDs (number)|STDs:condylomatosis|STDs:cervical condylomatos
is|STDs:vaginal condylomatosis|STDs:vulvo-perineal condylomatosis|STDs:syphilis|STDs:pelvic inflammatory disease|STDs:genital h
erpes|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|
+---+-------------------------+-----------------------+------------------+------+--------------+-------------------+----------
-------------+-----------------------------+---+-----------+----+-------------+-------------------+-------------------------
--+---------------------------+-----------------------------------+-------------+-------------------------------+--------------
-----+-------------------------+---------+--------+----------------+--------+-----------------------+-----------------------
---------+-----------------------------+---------+------+------+---+----------+--------+--------+------+
| 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|                          0.0|
0.0|                             0.0|          0.0|                                0.0|                0.0|
0.0|      0.0|     0.0|            0.0|     0.0|                          0|                                ?|
?|        0|     0|     0|  0|        0|       0|      0|     0|
| 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|                          0.0|
0.0|                             0.0|          0.0|                                0.0|                0.0|
0.0|      0.0|     0.0|            0.0|     0.0|                          0|                                ?|
?|        0|     0|     0|  0|        0|       0|      0|     0|
| 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|                          0.0|
0.0|                             0.0|          0.0|                                0.0|                0.0|
0.0|      0.0|     0.0|            0.0|     0.0|                          0|                                ?|
?|        0|     0|     0|  0|        0|       0|      0|     0|
| 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|                          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|     1|  0|        0|       0|      0|     0|
| 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|                          0.0|
0.0|                             0.0|          0.0|                                0.0|                0.0|
0.0|      0.0|     0.0|            0.0|     0.0|                          0|                                ?|
?|        0|     0|     0|  0|        0|       0|      0|     0|
+---+-------------------------+-----------------------+------------------+------+--------------+-------------------+----------
-------------+-----------------------------+---+-----------+----+-------------+-------------------+-------------------------
--+---------------------------+-----------------------------------+-------------+-------------------------------+--------------
-----+-------------------------+---------+--------+----------------+--------+-----------------------+-----------------------
---------+-----------------------------+---------+------+------+---+----------+--------+--------+------+
only showing top 5 rows
```

Figure 11. Print top 5 rows

In Figure-12 we are checking the datatypes of all our columns.

```
In [13]: #Check datatypes
         cervical.printSchema()
```

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

Figure 12. Checking Datatypes

3.1.2. CLEANING AND PREPARING DATASET

To prepare the dataset for optimal machine learning training, it was tidied by converting to relevant data type of float, removing invalid & null values, fitting null values by using Imputer library and scaling the data. Machine learning models are more suited to data whose datatype is numerical in nature, so we converted our datatype of our all columns to float for the better utilization of ML algorithm. Check Figure-13 to 22 for clarity.

```
In [14]: #DATA CLEANING

         #Convert data type from string to float
         from pyspark.sql.functions import col
         cervical = cervical.select(*(col(c).cast("float").alias(c) for c in cervical.columns))
         cervical.printSchema()

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

Figure 13. Conversion of all relevant data types to float

In the way towards cleaning the data, our first step was to convert all the invalid data entries “?” to null values so that we can drop them in the successive steps. The process of doing that can be seen in the *Figure-14*.

```
In [15]: # Replace invalid data '?' with null
         import pyspark.sql.functions as F
         from pyspark.sql.functions import col,when
         cervical=cervical.select([when(col(c)=="?",None).otherwise(col(c)).alias(c) for c in cervical.columns])
         cervical.show(5)
```

```
| Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Smokes|Smokes (years)|Smokes (packs/year)|Hormo
nal Contraceptives|Hormonal Contraceptives (years)|IUD|IUD (years)|STDs|STDs (number)|STDs:condylomatosis|STDs:cervical con
dylomatosis|STDs:vaginal condylomatosis|STDs:vulvo-perineal condylomatosis|STDs:syphilis|STDs:pelvic inflammatory disease|S
TDs: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|
+----+-----------------------+-----------------------+------------------+------+-------------+-------------------+-----
------------------+-----------------------------+---+-----------+----+-------------+-------------------+-----------------
-----------+-------------------------+-----------------------------+-------------+-----------------------------+-
------------------+-------------------------+---------+--------+----------------+--------+-------------------------+------
-------------------------+-----------------------------+---------+------+------+---+----------+--------+--------+------+
|18.0|                    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|                        0.0|
0.0|                           0.0|         0.0|                        0.0|                0.0|
0.0|     0.0|    0.0|           0.0|    0.0|                    0.0|                         null|
null|     0.0|   0.0|  0.0|0.0|      0.0|    0.0|   0.0|  0.0|
|15.0|                    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|                        0.0|
0.0|                           0.0|         0.0|                        0.0|                0.0|
0.0|     0.0|    0.0|           0.0|    0.0|                    0.0|                         null|
```

Figure 14. Replacing invalid data ‘?’ with null values

In the Figure-15 we dropped all the duplicate columns that were present in our dataset so that they do not skew our results.

```
In [16]: #Remove all duplicate
         cervical.dropDuplicates().show()
```

```
+
| Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Smokes|Smokes (years)|Smokes (packs/year)|Hormo
nal Contraceptives|Hormonal Contraceptives (years)| IUD|IUD (years)|STDs|STDs (number)|STDs:condylomatosis|STDs:cervical co
ndylomatosis|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|
+----+-----------------------+-----------------------+------------------+------+-------------+-------------------+-----
------------------+-----------------------------+---+-----------+----+-------------+-------------------+-----------------
-----------+-------------------------+-----------------------------+-------------+-----------------------------+
------------------+-------------------------+---------+--------+----------------+--------+-------------------------+------
-------------------------+-----------------------------+---------+------+------+---+----------+--------+--------+------+
+
|21.0|                    1.0|                  18.0|              1.0|   0.0|          0.0|                0.0|
1.0|                        4.0| 0.0|       0.0| 0.0|         0.0|              0.0|                        0.0|
0.0|                           0.0|         0.0|                        0.0|                0.0|
0.0|     0.0|    0.0|           0.0|    0.0|                    0.0|                         null|
null|     0.0|   0.0|  0.0|0.0|      0.0|    0.0|   0.0|  0.0|
|30.0|                    4.0|                  13.0|              8.0|   0.0|          0.0|                0.0|
null|                        null|null|       null| 1.0|         1.0|              0.0|                        0.0|
```

Figure 15. Drop duplicate columns/rows

Our next step as covered in Figure-16 was to drop all the columns that have only null values in them as they would not be adding any value to our ML model.

```
In [17]: #Remove columns that have only null values
         cervical.na.drop("all").show()
```

```
+----+-------------------------+-----------------------+------------------+------+--------------+-------------------+-----
------------------+----+-----------+----+-------------+-------------------+-----------------------+
------------+-------------------------+-------------------------------+-------------+-----------------------------+
-------------------+-------------------------+---------+--------+----------------+--------+-------------------------+-----
--------------------------+-----------------------------+---------+------+------+---+----------+--------+--------+------
+
| Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Smokes|Smokes (years)|Smokes (packs/year)|Hormo
nal Contraceptives|Hormonal Contraceptives (years)| IUD|IUD (years)|STDs|STDs (number)|STDs:condylomatosis|STDs:cervical co
ndylomatosis|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|
+----+-------------------------+-----------------------+------------------+------+--------------+-------------------+-----
------------------+----+-----------+----+-------------+-------------------+-----------------------+
------------+-------------------------+-------------------------------+-------------+-----------------------------+
-------------------+-------------------------+---------+--------+----------------+--------+-------------------------+-----
--------------------------+-----------------------------+---------+------+------+---+----------+--------+--------+------
+
|18.0|                    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|                        0.0|
```

Figure 16. Drop Columns that have only null values

After which, as seen in *Figure-17 and Figure-18,* we checked the count/number of null values still present in each column of our data using the pyspark.sql.functions library.

```
In [18]: # checking the sum/count of null values in all columns
         from pyspark.sql.functions import col, isnan, when, count

         null_count = cervical.select([count(when(isnan(i)| col(i).isNull(), i)).alias(i)for i in cervical.columns])
         null_count.show()
```

```
+---+-------------------------+-----------------------+------------------+------+--------------+-------------------+----------
------------+-------------------------------+---+-----------+----+-------------+-------------------+-------------------------
--+-------------------------+-------------------------------+-------------+-----------------------------+--------------
-----+-------------------------+---------+--------+----------------+--------+-------------------------+-----------------------
---------+-----------------------------+---------+------+-----+---+----------+--------+--------+------+
|Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Smokes|Smokes (years)|Smokes (packs/year)|Hormonal C
ontraceptives|Hormonal Contraceptives (years)|IUD|IUD (years)|STDs|STDs (number)|STDs:condylomatosis|STDs:cervical condylomatos
is|STDs:vaginal condylomatosis|STDs:vulvo-perineal condylomatosis|STDs:syphilis|STDs:pelvic inflammatory disease|STDs:genital h
erpes|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|
+---+-------------------------+-----------------------+------------------+------+--------------+-------------------+----------
------------+-------------------------------+---+-----------+----+-------------+-------------------+-------------------------
--+-------------------------+-------------------------------+-------------+-----------------------------+--------------
-----+-------------------------+---------+--------+----------------+--------+-------------------------+-----------------------
---------+-----------------------------+---------+------+-----+---+----------+--------+--------+------+
|  0|                       26|                      7|                56|    13|            13|                  13|
108|                         108|117|        117| 105|          105|                105|                          105|
105|                            105|          105|                          105|                105|
105|      105|     105|              105|     105|                        0|                            787|
787|        0|     0|     0|  0|         0|       0|       0|     0|
+---+-------------------------+-----------------------+------------------+------+--------------+-------------------+----------
------------+-------------------------------+---+-----------+----+-------------+-------------------+-------------------------
--+-------------------------+-------------------------------+-------------+-----------------------------+--------------
-----+-------------------------+---------+--------+----------------+--------+-------------------------+-----------------------
---------+-----------------------------+---------+------+-----+---+----------+--------+--------+------+
```

Figure 17. Null Count

```
In [19]: null_count = null_count.collect()[0].asDict()
         null_count

Out[19]: {'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}
```

Figure 18. Distinct Null count

In order to further clean our data, we dropped all those columns which had null values present in more than 20 percent of its rows. We didn't simple dropped all the null values because that would have left us with no data. For reference, please check Figure-19.

```
In [20]: #Drop columns with over 20% null values i.e 0.2*856 = 171
         cols_to_drop = [c for c,i in null_count.items() if i > 171]
         cervical = cervical.drop(*cols_to_drop)
         cervical.show(5)
```

```
+----+-------------------------+-------------------------+------------------+------+--------------+--------------------+---------
--------------+-------------------------------+---+-----------+----+-------------+--------------------+-------------------------
---+-----------------------------+-------------------------------+-------------+-------------------------------+-------------
------+-------------------------+---------+--------+----------------+--------+-------------------------+---------+------+-----
-+---+----------+--------+--------+------+
| 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 condylomato
sis|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:HP
V| Dx|Hinselmann|Schiller|Citology|Biopsy|
+----+-------------------------+-------------------------+------------------+------+--------------+--------------------+---------
--------------+-------------------------------+---+-----------+----+-------------+--------------------+-------------------------
---+-----------------------------+-------------------------------+-------------+-------------------------------+-------------
------+-------------------------+---------+--------+----------------+--------+-------------------------+---------+------+-----
-+---+----------+--------+--------+------+
|18.0|                     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|                         0.0|
0.0|                            0.0|         0.0|                         0.0|                0.0|
0.0|     0.0|    0.0|            0.0|    0.0|                    0.0|     0.0|  0.0|  0.0|0.0|      0.0|    0.0|
0.0|  0.0|
|15.0|                     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|                         0.0|
0.0|                            0.0|         0.0|                         0.0|                0.0|
0.0|     0.0|    0.0|            0.0|    0.0|                    0.0|     0.0|  0.0|  0.0|0.0|      0.0|    0.0|
0.0|  0.0|
|34.0|                     1.0|                     null|              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|
|52.0|                     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|                         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|
|46.0|                     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|                         0.0|
0.0|                            0.0|         0.0|                         0.0|                0.0|
0.0|     0.0|    0.0|            0.0|    0.0|                    0.0|     0.0|  0.0|  0.0|0.0|      0.0|    0.0|
0.0|  0.0|
+----+-------------------------+-------------------------+------------------+------+--------------+--------------------+---------
--------------+-------------------------------+---+-----------+----+-------------+--------------------+-------------------------
---+-----------------------------+-------------------------------+-------------+-------------------------------+-------------
------+-------------------------+---------+--------+----------------+--------+-------------------------+---------+------+-----
-+---+----------+--------+--------+------+
only showing top 5 rows
```

Figure 19. Drop Columns with over 20% null values

In Figure-20 to 21, we imported the imputer library and checked which columns have null values still present in them, so that we can fit them. We are fitting all the columns that had null values in them and them transform it with same name and output data has no null values in it.

```
In [21]: #Fitting the other null values
         #Importing relevant library
         from pyspark.ml.feature import Imputer


         #Check columns to fit
         cols_to_fit = [c for c,i in null_count.items() if i > 0]
         cols_to_fit
```

```
Out[21]: ['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: Time since first diagnosis',
          'STDs: Time since last diagnosis']
```

Figure 20. Check Columns to fit using Imputer

```
In [22]: #Fitting the null values
         imputer = Imputer(inputCols=['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'],
                         outputCols=['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'])
         model = imputer.fit(cervical)
         cervical = model.transform(cervical)
```

Figure 21. Fitting the null values

Next we check again the number of null values present in each column and it can be clearly seen in Figure-22 that there are no null values present after we used imputer to transform our dataset.

```
In [23]: #checking for null/nan values after fitting the data

         null_count = cervical.select([count(when(isnan(i)| col(i).isNull(), i)).alias(i)for i in cervical.columns])
         null_count = null_count.collect()[0].asDict()
         null_count

Out[23]: {'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}
```

Figure 22. Check null values

After doing all the data cleaning we checked how many columns we are left with. From Figure-23 we can see that we have 34 columns left. We started with 36.

```
In [24]: #Check new number of columns
         len(cervical.columns)

Out[24]: 34
```

Figure 23. Check number of columns

4. MACHINE LEARNING

In this section we will be creating three ML models for doing classification on our dataset:

1. Logistic Regression Classification
2. Naïve Bayes Classifier.
3. Random Forest Classifier.

Our first step would be to import all the libraries that we will be needing for creating and evaluating our ML models.

```
In [25]: #MACHINE LEARNING


         #Import relevant PySpark Machine Learning Libraries
         from pyspark.ml.feature import VectorAssembler
         from pyspark.ml.feature import StandardScaler
         from pyspark.ml.feature import StringIndexer

         from pyspark.ml.classification import LogisticRegression
         from pyspark.ml.classification import NaiveBayes
         from pyspark.ml.classification import RandomForestClassifier

         from pyspark.mllib.evaluation import BinaryClassificationMetrics
         from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

Figure 24. Importing relevant machine learning libraries

Next, we will be checking the count of people who need a “Biopsy” and the number of people who don’t need it. The “1” represents those people who need it and “0” represents people who don’t need it

```
In [26]: #Checking the Biopsy data.
         cervical.groupBy("Biopsy").count().show()

         +------+-----+
         |Biopsy|count|
         +------+-----+
         |   1.0|   55|
         |   0.0|  803|
         +------+-----+
```

Figure 25. Checking group of Cervical Dataset

.

4.1. VECTOR ASSEMBLER

For classification we need to convert all the predictor columns into one column. In order to achieve that we use Vector assembler which presents data in all the columns as a vector and puts it all into one single column which we named as “features”. In Figure-26 we deploy our Vector Assembler.

```
In [27]: #Vector assembling
         cols = cervical.columns
         cols.remove("Biopsy")
         assembler = VectorAssembler(inputCols = cols, outputCol = "features")
         cervical.show()
```

```
+----+-----------------------+-----------------------+-------------------+------+--------------+-------------------+-----+-----+---+-----------+--------+--------+------+
---------+------+------+---+----------+--------+--------+------+
| Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Smokes|Smokes (years)|Smokes (packs/year)|Hormo
nal Contraceptives|Hormonal Contraceptives (years)|      IUD|IUD (years)|STDs|STDs (number)|STDs:condylomatosis|STDs:cervic
al condylomatosis|STDs:vaginal condylomatosis|STDs:vulvo-perineal condylomatosis|STDs:syphilis|STDs:pelvic inflammatory dis
ease|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|
+----+-----------------------+-----------------------+-------------------+------+--------------+-------------------+-----
------------------+-----------------------------+---------+-----------+----+------------+-------------------+----------
-----------------+-------------------------+-------------------------------+-------------+-------------------------
----+-------------------+-------------------------+---------+--------+---------------+--------+-------------------------+
---------+------+------+---+----------+--------+--------+------+
|18.0|                    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|
0.0|                    0.0|                                0.0|          0.0|                          0.0|
0.0|                   0.0|      0.0|     0.0|             0.0|     0.0|                    0.0|      0.0|   0.0|
0.0|0.0|       0.0|     0.0|    0.0|   0.0|
|15.0|                    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|
0.0|                    0.0|                                0.0|          0.0|                          0.0|
```

Figure 26. Vector Assembling

In the next step shown in Figure-27 we transform our dataset using the model we created in the previous. For clarity we selected the “features” and “Biopsy” columns

```
In [28]: #Tranforming the dataset
         cervical = assembler.transform(cervical)
         cervical.select("features",'Biopsy').show(truncate = False)
```

```
+-------------------------------------------------------------------------------------------------------------------------------------
--------------+------+
|features
|Biopsy|
+-------------------------------------------------------------------------------------------------------------------------------------
--------------+------+
|(33,[0,1,2,3],[18.0,4.0,15.0,1.0])
|0.0   |
|(33,[0,1,2,3],[15.0,1.0,14.0,1.0])
|0.0   |
|(33,[0,1,2,3],[34.0,1.0,16.99530029296875,1.0])
|0.0   |
|(33,[0,1,2,3,4,5,6,7,8,26,28],[52.0,5.0,16.0,4.0,1.0,37.0,37.0,1.0,3.0,1.0,1.0])
|0.0   |
|(33,[0,1,2,3,7,8],[46.0,3.0,21.0,4.0,1.0,15.0])
|0.0   |
|(33,[0,1,2,3],[42.0,3.0,23.0,2.0])
|0.0   |
|(33,[0,1,2,3,4,5,6,9,10,30,31],[51.0,3.0,17.0,6.0,1.0,34.0,3.4000000953674316,1.0,7.0,1.0,1.0])
|1.0   |
|(33,[0,1,2,3,7,8,9,10],[26.0,1.0,26.0,3.0,1.0,2.0,1.0,7.0])
|0.0   |
|(33,[0,1,2,3,26,28,29],[45.0,1.0,20.0,5.0,1.0,1.0,1.0])
|0.0   |
|(33,[0,1,2,3,4,5,6,9,10],[44.0,3.0,15.0,2.2755610942840576,1.0,1.2669728994369507,2.799999952316284,0.11201079934835434,0.5148
043036460876])|0.0   |
|(33,[0,1,2,3,7,8],[44.0,3.0,26.0,4.0,1.0,2.0])
|0.0   |
|(33,[0,1,2,3,7,8],[27.0,1.0,17.0,3.0,1.0,8.0])
|0.0   |
|(33,[0,1,2,3,7,8,9,10],[45.0,4.0,14.0,6.0,1.0,10.0,1.0,5.0])
|0.0   |
|(33,[0,1,2,3,7,8],[44.0,2.0,25.0,2.0,1.0,5.0])
|0.0   |
|(33,[0,1,2,3,9,10],[43.0,2.0,18.0,5.0,1.0,8.0])
|0.0   |
|(33,[0,1,2,3,7,8],[40.0,3.0,18.0,2.0,1.0,15.0])
|0.0   |
|(33,[0,1,2,3,7,8],[41.0,4.0,21.0,3.0,1.0,0.25])
|0.0   |
|(33,[0,1,2,3,7,8],[43.0,3.0,15.0,8.0,1.0,3.0])
|0.0   |
|(33,[0,1,2,3,7,8,9,10,11,12,13,16],[42.0,2.0,20.0,2.2755610942840576,1.0,7.0,1.0,6.0,1.0,2.0,1.0,1.0])
|0.0   |
|(33,[0,1,2,3,9,10],[40.0,2.0,27.0,2.2755610942840576,1.0,1.0])
|0.0   |
+-------------------------------------------------------------------------------------------------------------------------------------
--------------+------+
only showing top 20 rows
```

Figure 27. Transforming Dataset

.

4.2. DATA SCALING

Next we scale our data so that we can achieve better results when processing our data for the machine learning algorithm. Please check Figure-28 and 29. In this step we are selecting the scaled column and our dependent column.

```
In [29]: #apply Scaling to achieve better results
         standardscaler = StandardScaler().setInputCol("features").setOutputCol("Scaled_features")
         cervical = standardscaler.fit(cervical).transform(cervical)
```

Figure 28. Data Scaling

```
In [30]: assembled_data = cervical.select("Scaled_features","Biopsy")
         assembled_data.show()

+--------------------+------+
|     Scaled_features|Biopsy|
+--------------------+------+
|(33,[0,1,2,3],[2....|   0.0|
|(33,[0,1,2,3],[1....|   0.0|
|(33,[0,1,2,3],[4....|   0.0|
|(33,[0,1,2,3,4,5,...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3],[4....|   0.0|
|(33,[0,1,2,3,4,5,...|   1.0|
|(33,[0,1,2,3,7,8,...|   0.0|
|(33,[0,1,2,3,26,2...|   0.0|
|(33,[0,1,2,3,4,5,...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3,7,8,...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3,9,10...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3,7,8]...|   0.0|
|(33,[0,1,2,3,7,8,...|   0.0|
|(33,[0,1,2,3,9,10...|   0.0|
+--------------------+------+
only showing top 20 rows
```

Figure 29. Assembled Data

Our next step in Figure 30 was to randomly split our data into two so that on one, we can train our models and on the next one we can test its accuracy. We put 70 percent of our rows in the training part and rest of the 30 in the testing part.

```
In [31]: #Splitting into train and test data
         train_cervical,test_cervical = assembled_data.randomSplit([0.7,0.3])

         train_cervical.show()
         test_cervical.show()

         +--------------------+------+
         |     Scaled_features|Biopsy|
         +--------------------+------+
         |(33,[0,1,2],[2.23...|   0.0|
         |(33,[0,1,2],[2.82...|   0.0|
         |(33,[0,1,2],[3.88...|   0.0|
         |(33,[0,1,2],[3.88...|   0.0|
         |(33,[0,1,2],[4.00...|   0.0|
         |(33,[0,1,2],[5.41...|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         +--------------------+------+
         only showing top 20 rows

         +--------------------+------+
         |     Scaled_features|Biopsy|
         +--------------------+------+
         |(33,[0,1,2],[1.52...|   0.0|
         |(33,[0,1,2],[2.70...|   0.0|
         |(33,[0,1,2],[4.00...|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[1....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         |(33,[0,1,2,3],[2....|   0.0|
         +--------------------+------+
         only showing top 20 rows
```

Figure 30. Data Splitting 70/30

4.3. LOGISTIC REGRESSION

In Figure-31 we are creating our first classifier model that is logistic Regression model. We named our model as “log_reg”. In the second line of code we are training our model by fitting the training part of our dataset into it.

```
In [32]: #LOGISTIC REGRESSION
         log_reg = LogisticRegression(labelCol = "Biopsy", featuresCol = "Scaled_features", maxIter=40)
         model = log_reg.fit(train_cervical)

         model

Out[32]: LogisticRegressionModel: uid=LogisticRegression_20de95b2fcf0, numClasses=2, numFeatures=33
```

Figure 31. Logistic Regression Model

In this step we are checking our model by transforming or we can say by putting the testing part of our dataset into our trained model and see its output.

It gives us five columns. The first column is the inputs, the second is the actuals outputs, third is the prediction calculations and the fourth is the probability of those calculations and the last is how its translates these raw_prediction and their probabilities into the binary output.

```
In [33]: prediction_test = model.transform(test_cervical)
         prediction_test.show()
```

```
+--------------------+------+--------------------+--------------------+----------+
|     Scaled_features|Biopsy|       rawPrediction|         probability|prediction|
+--------------------+------+--------------------+--------------------+----------+
|(33,[0,1,2],[1.52...|   0.0|[5.58824845372962...|[0.99627237153245...|       0.0|
|(33,[0,1,2],[2.70...|   0.0|[5.65702715706826...|[0.99651927250372...|       0.0|
|(33,[0,1,2],[4.00...|   0.0|[5.78602786304180...|[0.99693924393295...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.64499978484119...|[0.99647730409374...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.52500361581862...|[0.99602996968235...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.65549339595949...|[0.99651394842802...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.70695712323777...|[0.99668823627142...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.53376189117525...|[0.99606445223535...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.53202655541358...|[0.99605764375709...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.58522561845352...|[0.99626112867115...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.46175877790761...|[0.99577187171642...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.66425167131613...|[0.99654424186431...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.71571539859440...|[0.99671702015355...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[5.71571539859440...|[0.99671702015355...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[5.77593740122931...|[0.99690829911500...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[5.68003288626772...|[0.99659816579195...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[5.83615940386421...|[0.99708846598751...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[5.88762313114249...|[0.99723411010518...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[5.61150044452342...|[0.99635773434901...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[5.84318234345918...|[0.99710878297079...|       0.0|
+--------------------+------+--------------------+--------------------+----------+
only showing top 20 rows
```

Figure 32. Prediction Test

Below, we can see some of the actual values of our dependent columns and the respective predictions of our model

```
In [34]: prediction_test.select("Biopsy","prediction").show(10)
```

```
+------+----------+
|Biopsy|prediction|
+------+----------+
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
+------+----------+
only showing top 10 rows
```

Figure 33. Prediction Test Sample Comparison

.

```
In [35]: #compute raw scores on the test set
         predictionAndLabels = prediction_test.select("Biopsy","prediction").rdd
         predictionAndLabels.collect()

Out[35]: [Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
          Row(Biopsy=0.0, prediction=0.0),
```

Figure 34. Compute raw scores on test dataset

```
In [36]: metrics = BinaryClassificationMetrics(predictionAndLabels)

C:\Users\chiam\OneDrive\Documents\spark\spark-3.2.1-bin-hadoop3.2\python\pyspark\sql\context.py:125: FutureWarning: Deprecated
in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
```

Figure 35. Binary Classification Metrics

```
In [37]: #Area under ROC curve
         print("Area under ROC = %s" % metrics.areaUnderROC)

Area under ROC = 0.8374208860759493
```

Figure 36. Area under ROC curve

In this step Figure-37 we are using our evaluator create above to check the accuracy of our model. The accuracy of model comes out to be around 97 percent which is pretty good.

```
In [38]: evaluator = MulticlassClassificationEvaluator(labelCol = "Biopsy", predictionCol = "prediction", metricName = "accuracy")
         accuracy_log_reg = evaluator.evaluate(prediction_test)
         print("Logistic Reg Accuracy = ",accuracy_log_reg)

Logistic Reg Accuracy =  0.9683794466403162
```

Figure 37. Multiclass Classification Evaluator

4.4. NAÏVE BAYES

As done in the above Logistic Regression case, we have created our Naïve Bayes classification model below and trained it with our training data.

```
In [39]: #NAIVEBAYES
naive_bayes = NaiveBayes(labelCol = "Biopsy", featuresCol = "Scaled_features", smoothing = 1.0)

model = naive_bayes.fit(train_cervical)
model

Out[39]: NaiveBayesModel: uid=NaiveBayes_cf409dd69025, modelType=multinomial, numClasses=2, numFeatures=33
```

Figure 38. NaivesBayes modelling

In Figure-39, we are testing our Trained model by putting testing part of our data set in it and then we are comparing the predicted values with the actual values. In this we are simply comparing the real values and the predicted values.

```
In [40]: #select example rows to display
prediction_test = model.transform(test_cervical)
prediction_test.show()

+--------------------+------+--------------------+--------------------+----------+
|     Scaled_features|Biopsy|       rawPrediction|         probability|prediction|
+--------------------+------+--------------------+--------------------+----------+
|(33,[0,1,2],[1.52...|   0.0|[-9.5808914477509...|[0.99863451852707...|       0.0|
|(33,[0,1,2],[2.70...|   0.0|[-14.011632583146...|[0.99965887485007...|       0.0|
|(33,[0,1,2],[4.00...|   0.0|[-19.056981644766...|[0.99993557187088...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-13.705364689730...|[0.99945864115312...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-13.469717378786...|[0.99948296825971...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-12.160712023854...|[0.99929381819685...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-12.565039958075...|[0.99943396628588...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-13.679997370788...|[0.99951188313946...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-15.434930028666...|[0.99964675702078...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-14.084325305010...|[0.99960877123522...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-17.358543309822...|[0.99980433218730...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-12.370992015856...|[0.99933330419378...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-12.775319950077...|[0.99946562012443...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[-12.775319950077...|[0.99946562012443...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[-13.389927876301...|[0.99959564250066...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[-14.546484657738...|[0.99956995909379...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[-14.004535802524...|[0.99969403823113...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[-14.408863736745...|[0.99975477856916...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[-14.715165281015...|[0.99967081513849...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[-15.969748452404...|[0.99979097682311...|       0.0|
+--------------------+------+--------------------+--------------------+----------+
only showing top 20 rows
```

```
In [41]: prediction_test.select("Biopsy","prediction").show(10)

predictionAndLabels = prediction_test.select("Biopsy","prediction").rdd

+------+----------+
|Biopsy|prediction|
+------+----------+
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
+------+----------+
only showing top 10 rows
```

Figure 39. Prediction Testing

In Figure-40, we are checking the area under the ROC curve, used to check the sensitivity and efficicency of our model.

```
In [42]: metrics = BinaryClassificationMetrics(predictionAndLabels)

In [43]: #Area under ROC curve
         print("Area under ROC = %s" % metrics.areaUnderROC)

         Area under ROC = 0.7049305320230431
```

Figure 40. ROC curve

In Figure-41 we are applying the evaluator to the Naïve Bayes model as we did in the previous model to the check the accuracy of it. The accuracy of this classifier is slightly lower than the previous one, around 93%.

```
In [44]: #Select (prediction, true label) and compute test error
         evaluator = MulticlassClassificationEvaluator(labelCol = "Biopsy", predictionCol = "prediction", metricName = "accuracy")
         accuracy_naive_bayes = evaluator.evaluate(prediction_test)
         print("Naive Bayes Accuracy = ",accuracy_naive_bayes)

         Naive Bayes Accuracy =  0.9288537549407114
```

Figure 41. Multiclass Classification Evaluator

4.5. RANDOM FOREST

In this part we will be applying the Random Forest model to our dataset which is actually a specialised type of decision tree model. In this step we are creating a Random Forest classification model and then training it with our training data set. Then we are putting our testing data set to our trained model and comparing the actual values and the predicted values. See Figures 42-45.

```
In [45]: #RANDOM FOREST CLASSIFIER

         random_forest_classifier = RandomForestClassifier(labelCol = "Biopsy", featuresCol = "Scaled_features", numTrees = 40)
         model = random_forest_classifier.fit(train_cervical)
         model

Out[45]: RandomForestClassificationModel: uid=RandomForestClassifier_9cdfd271d805, numTrees=40, numClasses=2, numFeatures=33
```

Figure 42. RFC Model

```
In [46]: prediction_test = model.transform(test_cervical)
         prediction_test.show()
```

```
+--------------------+------+--------------------+--------------------+----------+
|     Scaled_features|Biopsy|       rawPrediction|         probability|prediction|
+--------------------+------+--------------------+--------------------+----------+
|(33,[0,1,2],[1.52...|   0.0|[38.5186909633311...|[0.96296727408327...|       0.0|
|(33,[0,1,2],[2.70...|   0.0|[39.5495904342305...|[0.98873976085576...|       0.0|
|(33,[0,1,2],[4.00...|   0.0|[39.450881989533,...|[0.98627204973832...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[1....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[39.4452330984588...|[0.98613082746147...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[39.4064993725661...|[0.98516248431415...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[39.5052078172636...|[0.98763019543159...|       0.0|
|(33,[0,1,2,3],[2....|   0.0|[39.4452330984588...|[0.98613082746147...|       0.0|
+--------------------+------+--------------------+--------------------+----------+
only showing top 20 rows
```

Figure 43. Prediction test

```
In [47]: prediction_test.select("Biopsy","prediction").show(10)
```

```
+------+----------+
|Biopsy|prediction|
+------+----------+
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
|   0.0|       0.0|
+------+----------+
only showing top 10 rows
```

Figure 44. Prediction test select sample

Here we are creating the Binary classification metrics to check the efficiency of our model. In the successive steps we will get the results of this model.

```
In [48]: predictionAndLabels = prediction_test.select("Biopsy","prediction").rdd.map(lambda row: row[0:])

         metrics = BinaryClassificationMetrics(predictionAndLabels)

In [49]: #Area under ROC curve
         print("Area under ROC = %s" % metrics.areaUnderROC)

         Area under ROC = 0.8529116465863453
```

Figure 45. Metrics and ROC

Finally, we are checking the accuracy of our model and it comes out to be around 95 percent which is again slightly lower than Logistic Regression model.

```
In [50]: #Select (prediction, true label) and compute test error
         evaluator = MulticlassClassificationEvaluator(labelCol = "Biopsy", predictionCol = "prediction", metricName = "accuracy")
         accuracy_random_forest = evaluator.evaluate(prediction_test)
         print("Random Forest Accuracy = ",accuracy_random_forest)

         Random Forest Accuracy =  0.9525691699604744
```

Figure 46. Model Accuracy

4.6. MODEL ACCURACY

Comparing the accuracies of all the three models we have created, we can clearly see that Logistic Regression model gives our the best accuracy. Even though other two models are not far behind.

```
In [51]: print("Accuracy of Logistics Reg = ",accuracy_log_reg)
         print("Accuracy of Naive Bayes   = ",accuracy_naive_bayes)
         print("Accuracy of Random Forest = ",accuracy_random_forest)

         Accuracy of Logistics Reg =  0.9683794466403162
         Accuracy of Naive Bayes   =  0.9288537549407114
         Accuracy of Random Forest =  0.9525691699604744
```

Figure 47. Classification Models Comparison

4.7. K-MEANS CLUSTERING

We are using K-means clustering to attempt to find groups or cluster relationships that might exist among the variables. First we import all relevant libraries, select the variables we wish to work with, assemble vectors, pepare the data, then deploy the Kmeans model. Please see Fig 48-55 for full details

```
#KMEANS Clustering
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import StringIndexer
```

Figure 48. Relevant Libraries

```
#identify observation clusters by applying K-means clustering on the data variables: age, number of sexual partners,
#age of first sexual experience,number of pregnancies,hormonal contraceptives(years), HPV STD
```

```
cervical.columns
```

```
['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',
 'features',
 'Scaled_features']
```

Figure 49. Dataset Columns

```
cervical = cervical.select('Age',
 'Number of sexual partners',
 'First sexual intercourse',
 'Num of pregnancies',
 'Hormonal Contraceptives (years)',
 'STDs:HPV',
 'Biopsy')
cervical.show()
```

Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Hormonal Contraceptives (years)	STDs:HPV	Biopsy
18.0	4.0	15.0	1.0	0.0	0.0	0.0
15.0	1.0	14.0	1.0	0.0	0.0	0.0
34.0	1.0	16.9953	1.0	0.0	0.0	0.0
52.0	5.0	16.0	4.0	3.0	0.0	0.0
46.0	3.0	21.0	4.0	15.0	0.0	0.0
42.0	3.0	23.0	2.0	0.0	0.0	0.0
51.0	3.0	17.0	6.0	0.0	0.0	1.0
26.0	1.0	26.0	3.0	2.0	0.0	0.0
45.0	1.0	20.0	5.0	0.0	0.0	0.0
44.0	3.0	15.0	2.275561	0.0	0.0	0.0
44.0	3.0	26.0	4.0	2.0	0.0	0.0
27.0	1.0	17.0	3.0	8.0	0.0	0.0
45.0	4.0	14.0	6.0	10.0	0.0	0.0
44.0	2.0	25.0	2.0	5.0	0.0	0.0
43.0	2.0	18.0	5.0	0.0	0.0	0.0
40.0	3.0	18.0	2.0	15.0	0.0	0.0
41.0	4.0	21.0	3.0	0.25	0.0	0.0
43.0	3.0	15.0	8.0	3.0	0.0	0.0
42.0	2.0	20.0	2.275561	7.0	0.0	0.0
40.0	2.0	27.0	2.275561	0.0	0.0	0.0

only showing top 20 rows

Figure 50. Selected Columns/Variables

```
#Vector assembling
cols = cervical.columns
cols.remove("Biopsy")
assembler = VectorAssembler(inputCols = cols, outputCol = "features")
cervical.show()
```

Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Hormonal Contraceptives (years)	STDs:HPV	Biopsy
18.0	4.0	15.0	1.0	0.0	0.0	0.0
15.0	1.0	14.0	1.0	0.0	0.0	0.0
34.0	1.0	16.9953	1.0	0.0	0.0	0.0
52.0	5.0	16.0	4.0	3.0	0.0	0.0
46.0	3.0	21.0	4.0	15.0	0.0	0.0
42.0	3.0	23.0	2.0	0.0	0.0	0.0
51.0	3.0	17.0	6.0	0.0	0.0	1.0
26.0	1.0	26.0	3.0	2.0	0.0	0.0
45.0	1.0	20.0	5.0	0.0	0.0	0.0
44.0	3.0	15.0	2.275561	0.0	0.0	0.0
44.0	3.0	26.0	4.0	2.0	0.0	0.0
27.0	1.0	17.0	3.0	8.0	0.0	0.0
45.0	4.0	14.0	6.0	10.0	0.0	0.0
44.0	2.0	25.0	2.0	5.0	0.0	0.0
43.0	2.0	18.0	5.0	0.0	0.0	0.0
40.0	3.0	18.0	2.0	15.0	0.0	0.0
41.0	4.0	21.0	3.0	0.25	0.0	0.0
43.0	3.0	15.0	8.0	3.0	0.0	0.0
42.0	2.0	20.0	2.275561	7.0	0.0	0.0
40.0	2.0	27.0	2.275561	0.0	0.0	0.0

only showing top 20 rows

Figure 51. Vector Assembling

```
#Tranforming the dataset
cervical = assembler.transform(cervical)
cervical.select("features",'Biopsy').show(truncate = False)
```

features	Biopsy
[18.0,4.0,15.0,1.0,0.0,0.0]	0.0
[15.0,1.0,14.0,1.0,0.0,0.0]	0.0
[34.0,1.0,16.99530029296875,1.0,0.0,0.0]	0.0
[52.0,5.0,16.0,4.0,3.0,0.0]	0.0
[46.0,3.0,21.0,4.0,15.0,0.0]	0.0
[42.0,3.0,23.0,2.0,0.0,0.0]	0.0
[51.0,3.0,17.0,6.0,0.0,0.0]	1.0
[26.0,1.0,26.0,3.0,2.0,0.0]	0.0
[45.0,1.0,20.0,5.0,0.0,0.0]	0.0
[44.0,3.0,15.0,2.275561094284057 6,0.0,0.0]	0.0
[44.0,3.0,26.0,4.0,2.0,0.0]	0.0
[27.0,1.0,17.0,3.0,8.0,0.0]	0.0
[45.0,4.0,14.0,6.0,10.0,0.0]	0.0
[44.0,2.0,25.0,2.0,5.0,0.0]	0.0
[43.0,2.0,18.0,5.0,0.0,0.0]	0.0
[40.0,3.0,18.0,2.0,15.0,0.0]	0.0
[41.0,4.0,21.0,3.0,0.25,0.0]	0.0
[43.0,3.0,15.0,8.0,3.0,0.0]	0.0
[42.0,2.0,20.0,2.275561094284057 6,7.0,0.0]	0.0
[40.0,2.0,27.0,2.275561094284057 6,0.0,0.0]	0.0

only showing top 20 rows

Figure 52. Dataset Transformation

```
cervical.groupBy("Number of sexual partners").count().show()
```

Number of sexual partners	count
9.0	1
5.0	44
7.0	7
2.0	272
3.0	208
10.0	1
2.5276442	26
1.0	206
6.0	9
8.0	4
15.0	1
4.0	78
28.0	1

Figure 53. Number of Sexual Partners

```
cervical.groupBy("Biopsy").count().show()
```

```
+------+-----+
|Biopsy|count|
+------+-----+
|   1.0|   55|
|   0.0|  803|
+------+-----+
```

Figure 54. Biopsy Count

We used k=2 for our Random Forest trees because we are attempting to compare if the clustered groups will be similar to the above Biopsy group count of 55 to 803. However from our results, the 858 was clustered into two groups of 312 and 546, which shows another relationship exists among the variables that is not related to the Biopsy.

```
kmeans = KMeans(featuresCol = "features", k =2)

model = kmeans.fit(cervical)
model
```

```
KMeansModel: uid=KMeans_6b0a42668e7e, k=2, distanceMeasure=euclidean, numFeatures=6
```

Figure 55. KMeans Modelling

```
model.transform(cervical).groupBy("prediction").count().show()
```

```
+----------+-----+
|prediction|count|
+----------+-----+
|         1|  312|
|         0|  546|
+----------+-----+
```

Figure 56. Prediction count

```
predictions = model.transform(cervical)
predictions.show()
```

```
+----+-------------------------+-----------------------+------------------+------------------------------+--------+------+---
-----------------+----------+
| Age|Number of sexual partners|First sexual intercourse|Num of pregnancies|Hormonal Contraceptives (years)|STDs:HPV|Biopsy|
features|prediction|
+----+-------------------------+-----------------------+------------------+------------------------------+--------+------+---
-----------------+----------+
|18.0|                      4.0|                   15.0|               1.0|                           0.0|     0.0|   0.0|[1
8.0,4.0,15.0,1....|         0|
|15.0|                      1.0|                   14.0|               1.0|                           0.0|     0.0|   0.0|[1
5.0,1.0,14.0,1....|         0|
|34.0|                      1.0|                16.9953|               1.0|                           0.0|     0.0|   0.0|[3
4.0,1.0,16.9953...|         1|
|52.0|                      5.0|                   16.0|               4.0|                           3.0|     0.0|   0.0|[5
2.0,5.0,16.0,4....|         1|
|46.0|                      3.0|                   21.0|               4.0|                          15.0|     0.0|   0.0|[4
6.0,3.0,21.0,4....|         1|
|42.0|                      3.0|                   23.0|               2.0|                           0.0|     0.0|   0.0|[4
2.0,3.0,23.0,2....|         1|
|51.0|                      3.0|                   17.0|               6.0|                           0.0|     0.0|   1.0|[5
1.0,3.0,17.0,6....|         1|
|26.0|                      1.0|                   26.0|               3.0|                           2.0|     0.0|   0.0|[2
6.0,1.0,26.0,3....|         0|
|45.0|                      1.0|                   20.0|               5.0|                           0.0|     0.0|   0.0|[4
5.0,1.0,20.0,5....|         1|
|44.0|                      3.0|                   15.0|          2.275561|                           0.0|     0.0|   0.0|[4
4.0,3.0,15.0,2....|         1|
|44.0|                      3.0|                   26.0|               4.0|                           2.0|     0.0|   0.0|[4
4.0,3.0,26.0,4....|         1|
|27.0|                      1.0|                   17.0|               3.0|                           8.0|     0.0|   0.0|[2
7.0,1.0,17.0,3....|         0|
|45.0|                      4.0|                   14.0|               6.0|                          10.0|     0.0|   0.0|[4
5.0,4.0,14.0,6....|         1|
|44.0|                      2.0|                   25.0|               2.0|                           5.0|     0.0|   0.0|[4
4.0,2.0,25.0,2....|         1|
|43.0|                      2.0|                   18.0|               5.0|                           0.0|     0.0|   0.0|[4
3.0,2.0,18.0,5....|         1|
|40.0|                      3.0|                   18.0|               2.0|                          15.0|     0.0|   0.0|[4
0.0,3.0,18.0,2....|         1|
|41.0|                      4.0|                   21.0|               3.0|                          0.25|     0.0|   0.0|[4
1.0,4.0,21.0,3....|         1|
|43.0|                      3.0|                   15.0|               8.0|                           3.0|     0.0|   0.0|[4
3.0,3.0,15.0,8....|         1|
|42.0|                      2.0|                   20.0|          2.275561|                           7.0|     0.0|   0.0|[4
2.0,2.0,20.0,2....|         1|
|40.0|                      2.0|                   27.0|          2.275561|                           0.0|     0.0|   0.0|[4
0.0,2.0,27.0,2....|         1|
+----+-------------------------+-----------------------+------------------+------------------------------+--------+------+---
-----------------+----------+
only showing top 20 rows
```

Figure 57. Predicitions

```
predictions.groupBy("Biopsy","prediction").count().show()
```

```
+------+----------+-----+
|Biopsy|prediction|count|
+------+----------+-----+
|   1.0|         0|   31|
|   0.0|         0|  515|
|   0.0|         1|  288|
|   1.0|         1|   24|
+------+----------+-----+
```

Figure 58. Prediction Group

5. DATA VISUALIZATION

Tableau software is an interactive data visualization software tool, useful for plotting and displaying insights in order to, optimise decision making. For our cervical cancer data set, we will use tableau to look at different organizational insights regarding the data collected from the female respondents.

To begin our data visualization on Tableau, we first import our Cervical cancer csv file. Then calculate some extra variables we need such as Number of responses and Age(bin), using the Analysis tab -→ Create Calculated Field.

Table 1. Tableau

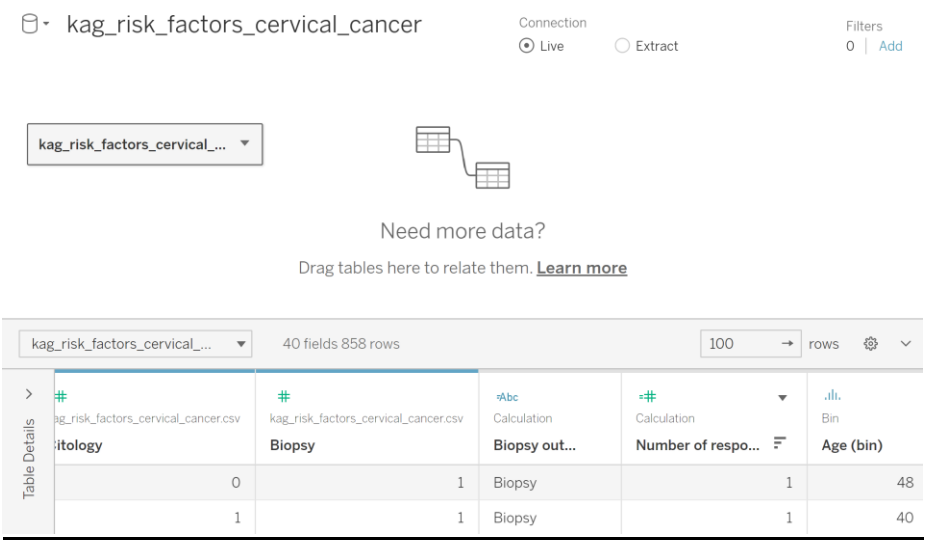
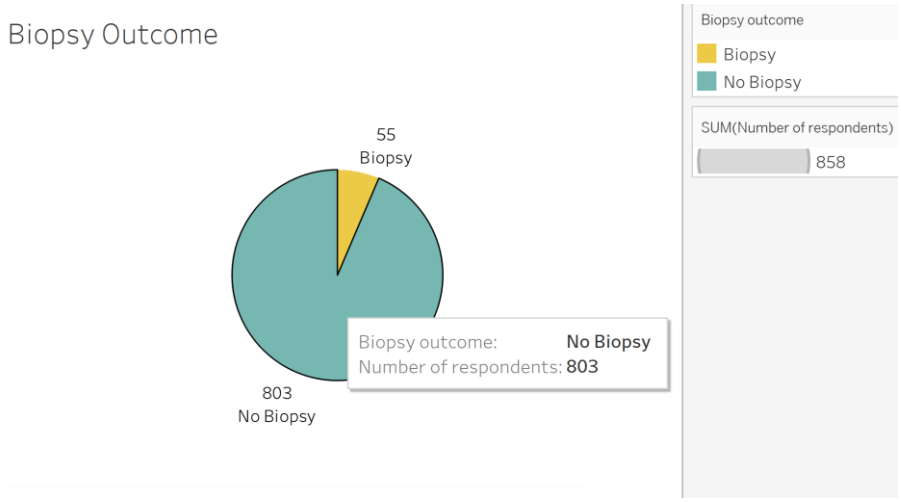


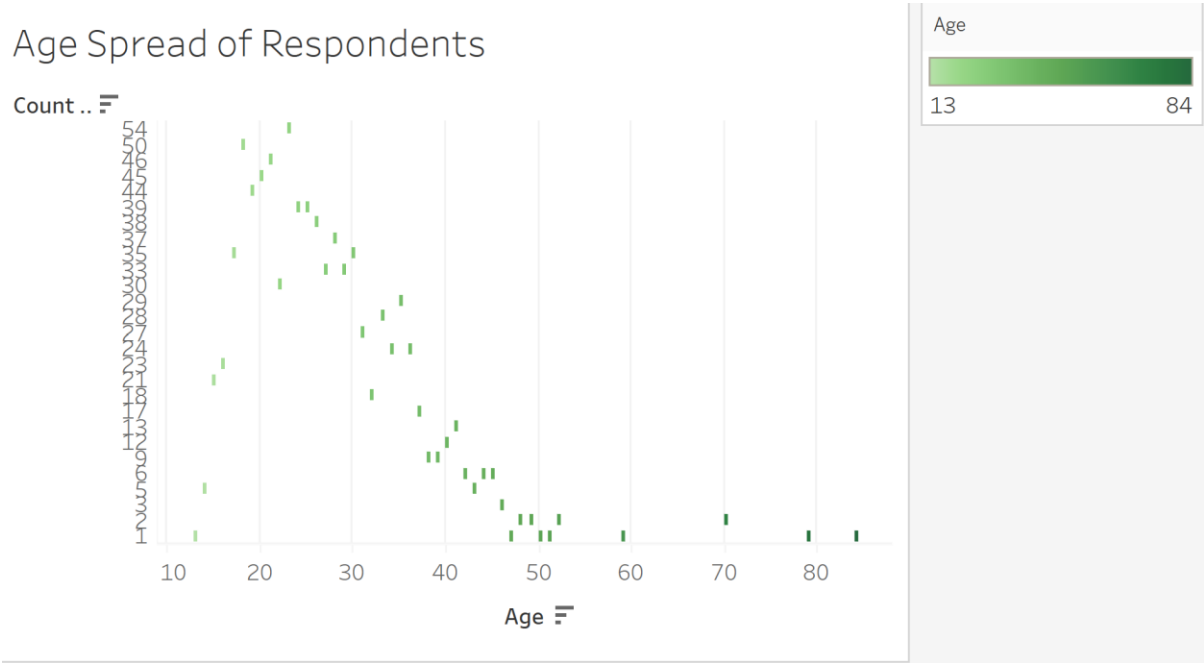
Table 2. Calculated Fields

From graph1 below, we can observe that only 55 out of the 858 respondents needed a cervical cancer biopsy. This comes to roughly 6% of the dataset and is a small representation.

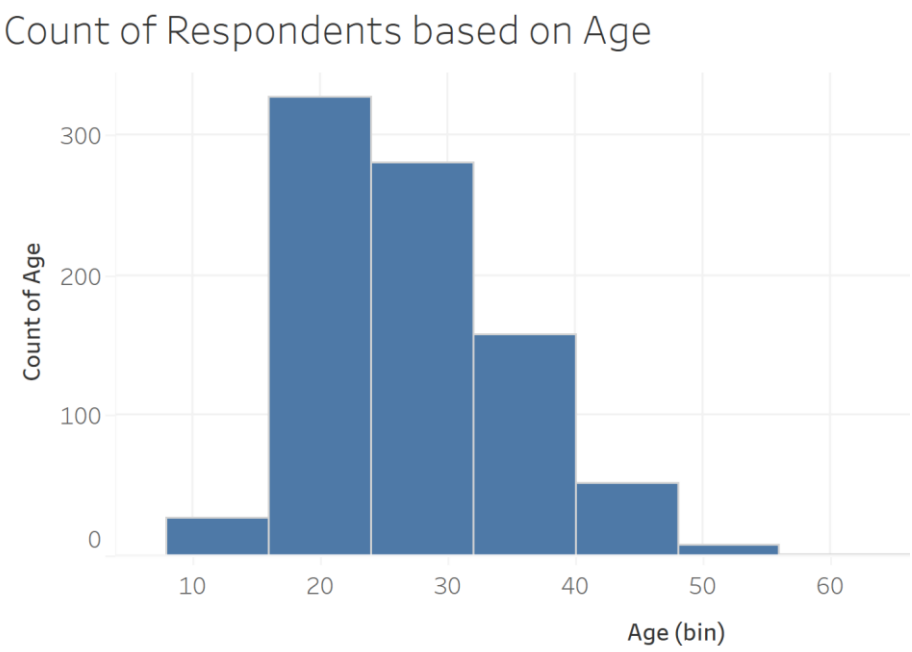


Graph 1. Biopsy Outcome

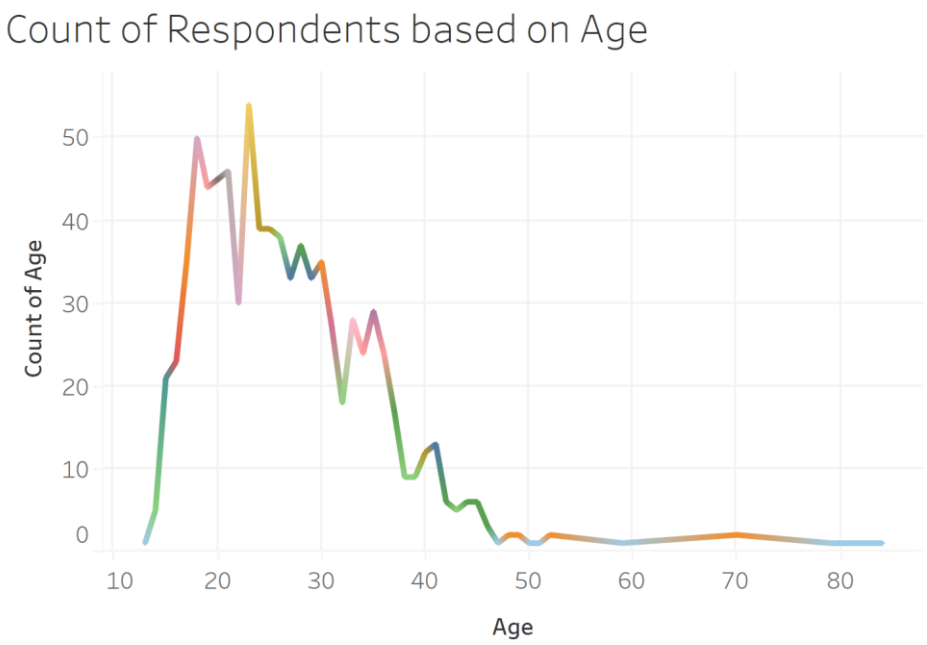
Graph 2-4 show that out of 858 respondents, the age distribution of the dataset population is densely populated among 20 year olds. This could automatically skew the result of the machine learning algorithm trained using this dataset as the most common age bracket of women affected by cervical cancer is 30-45years old.



Graph 2

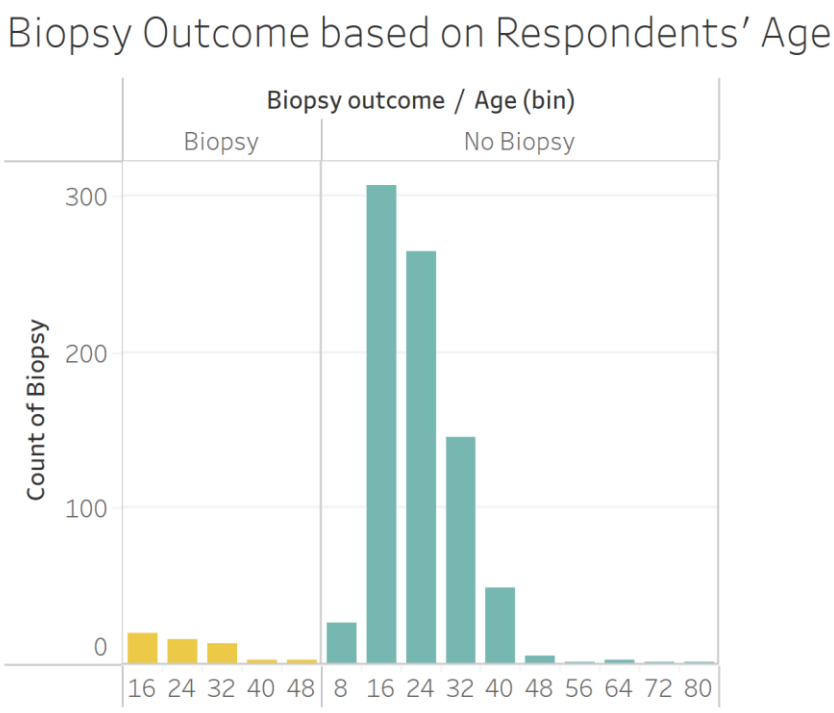


Graph 3

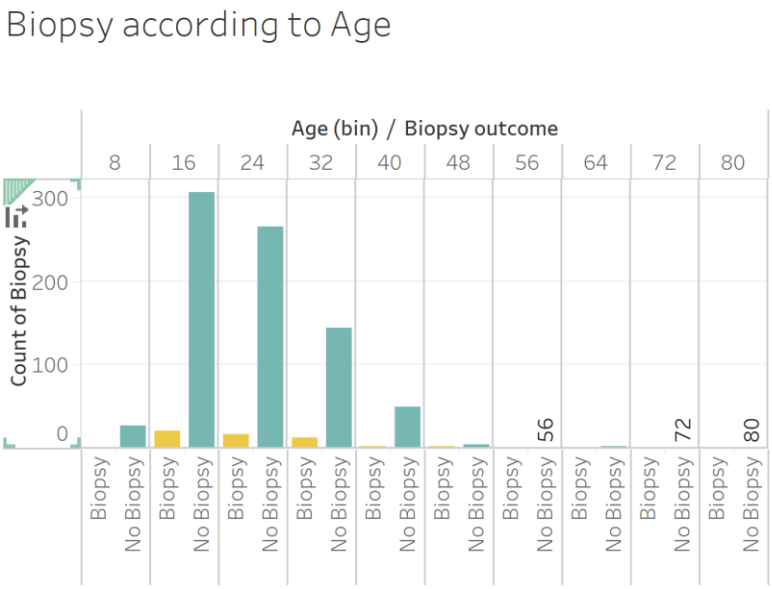


Graph 4

Graph 5 – 6, we can observe that the age group that needed Biopsy the most were between 16-32 age bin. This shows that more young people are likely to be diagnosed with cervical cancer, however this is most likely not the case as the data was disproportionately collected from the 20s age bracket and this would influence heavily the outcome of the dataset processing.

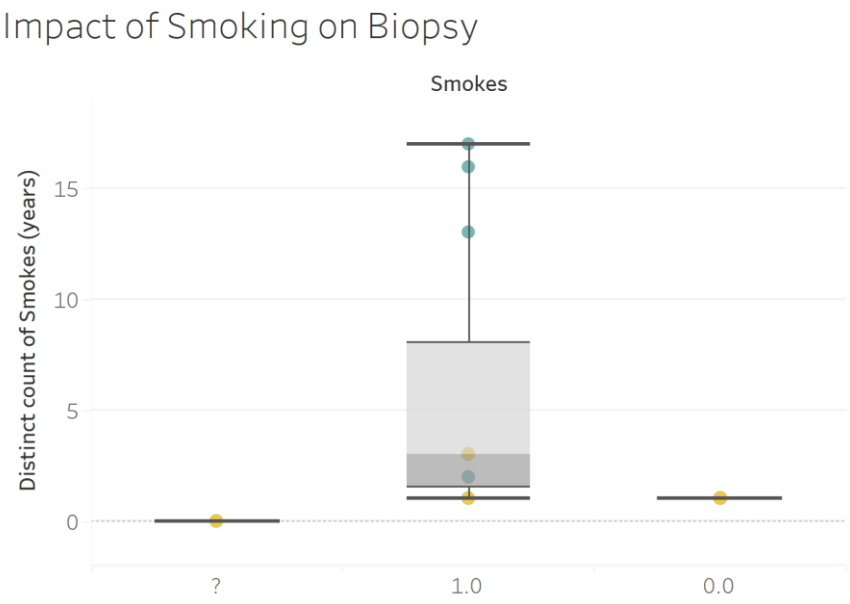


Graph 5


Biopsy according to Age
Age (bin) / Biopsy outcome
8
16
24
32
40
48
56
64
72
80
Count of Biopsy
300
200
100
0
56
72
80
Biopsy
No Biopsy

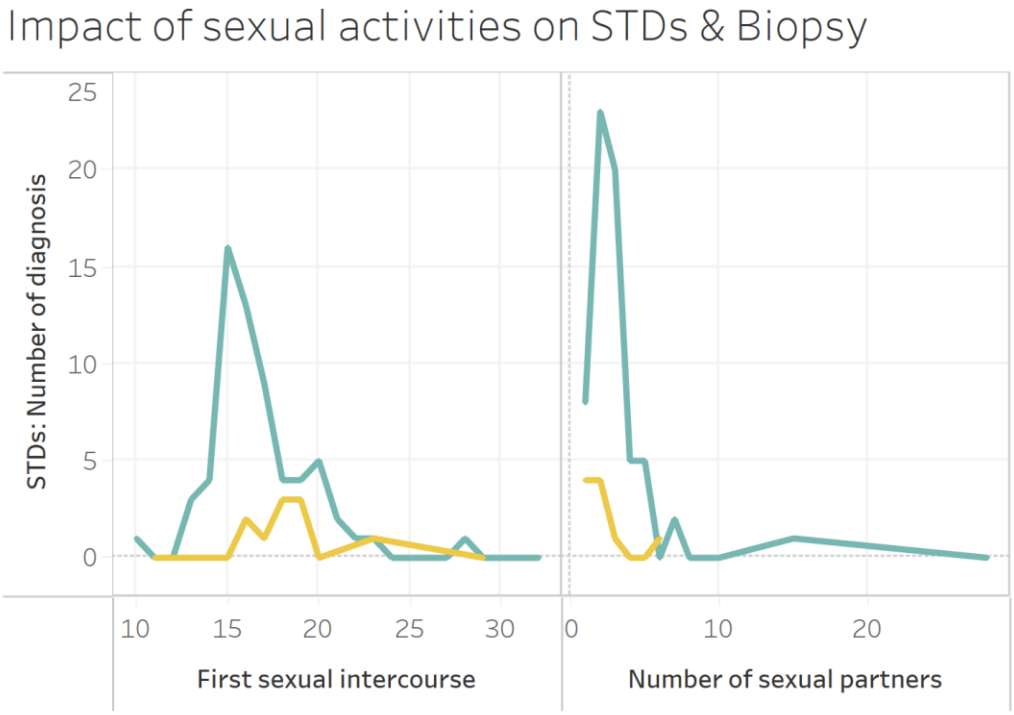
Graph 6

Graph 7 shows the relationship between smoking and the probability of a Biopsy. From the boxplot below, we can conclude that smoking does not necessarily affect Biopsy as both long-term smokers and non-smokers seem affected with no discernible relationship. Infact, we have outliers of long-term smokers who don't need a biopsy.



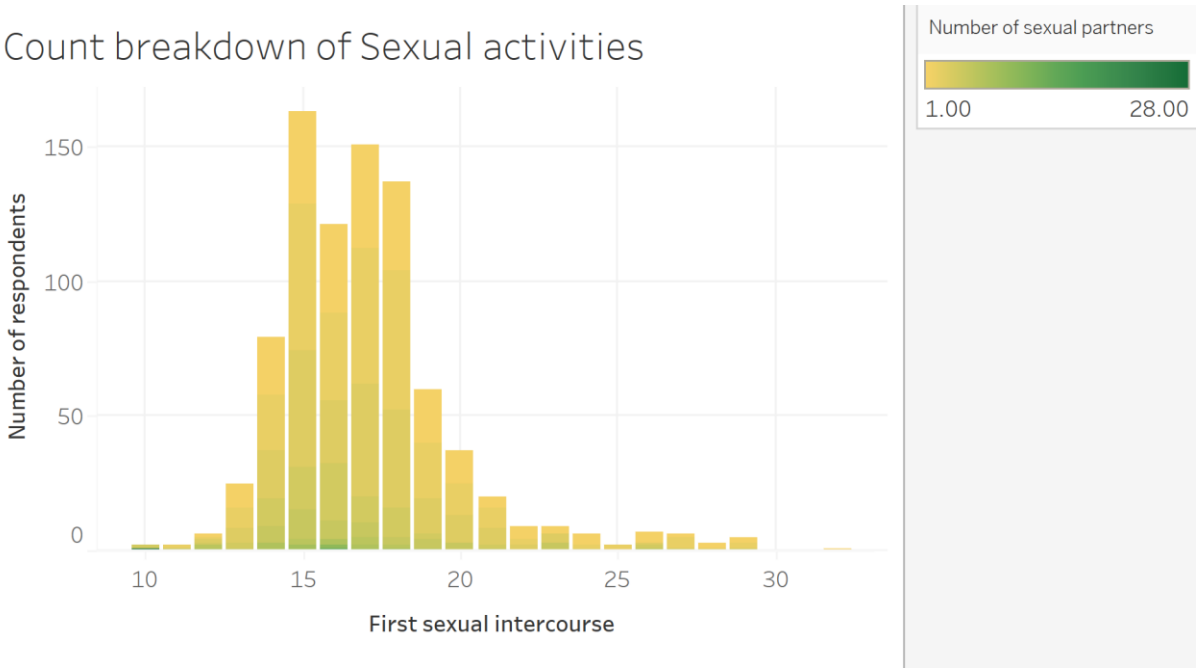
Graph 7

Graph 8 shows the impact of having sex at a young age and having multiple sex partners on the Biopsy and STD diagnosis. It appears that starting sexual relations at an early age increases the risk of STDs and Biopsy need. However having more sexual partners does not show same positive relationship.



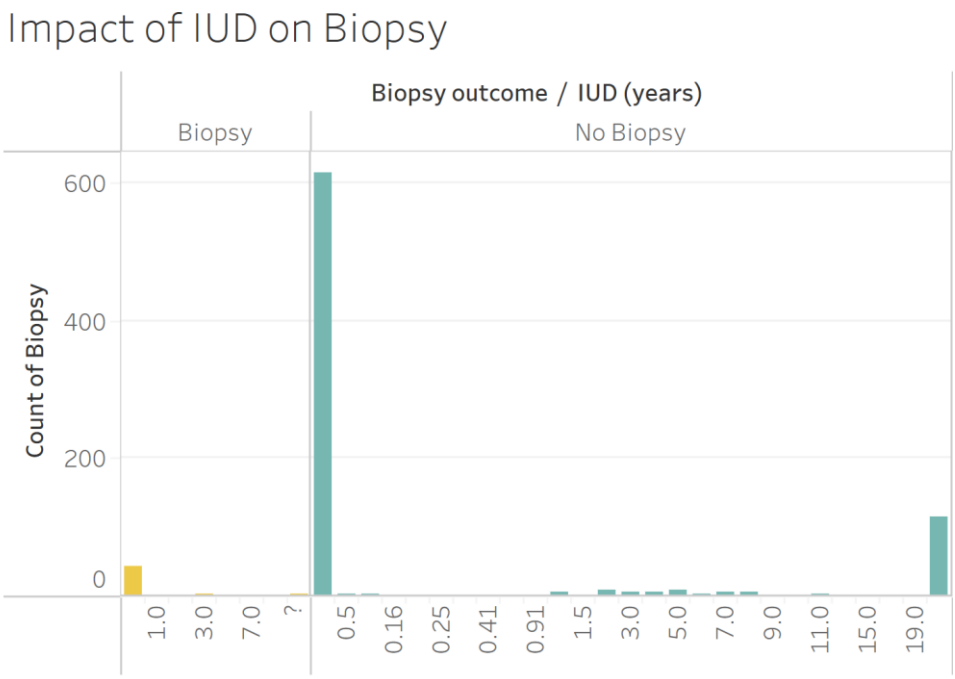
Graph 8

Graph 9 shows that most of the respondents started having sex from the age of 15-18 but have maintained single digit sexual partners.



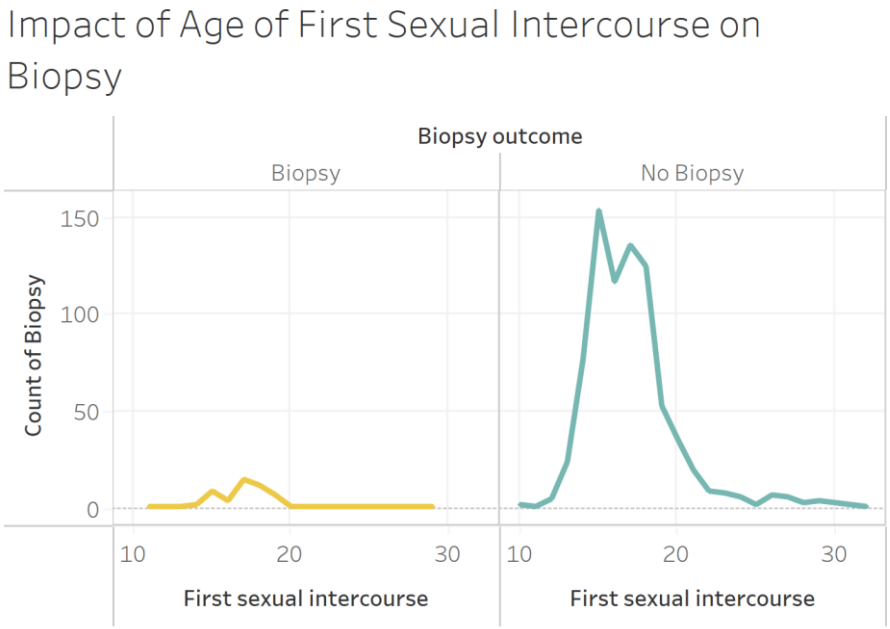
Graph 9

Graph 10 shows the relationship of using the IUD contraceptive for years, on Biopsy. There does not appear to be any clear impact of the length of usage of IUDs on Biopsy. If anything, most of the respondents who required a Biopsy have been on the IUD for less than a year .



Graph 10

Graph 11 looks at the impact of age of first sexual intercourse on the need for a Biopsy and it appears that sex before 20 could increase chances of needing Biopsy.



Graph 11

6. DISCUSSION

From the processing and analysing of this cervical dataset, it can be observed that it has abnormalities and the data collected from the respondents was largely skewed towards a particular age demography instead of being evenly distributed. It also has a lot of missing values which may be as a result of the respondents being unwilling to give out sensitive personal information regarding their sexual habits and any sexually transmitted diseases they might have.

Inferences deduced from the dataset include that:

- Most of the respondents are aged 20 -45.
- Non-smokers make up 80% of the dataset and smoking does not have any direct correlation with need for a biopsy.
- Having sex at a young age increases the likelihood of the Biopsy and STD diagnosis but multiple sex partners does not appear to have any significant effect
- Contraceptives such as IUD had no significant impact on the need for a biopsy

7. CONCLUSION

The classification models used gave very high accuracies in predicting the data but despite this seemingly favourable outcomes, including cleaning and inputting the missing values before training the dataset for prediction analysis, we cannot confidently rely because of the inherent collection biases mentioned above.

8. REFERENCES

Lee, C. (2022). How to Install and Run PySpark in Jupyter Notebook on Windows. Retrieved 1 July 2022, from https://changhsinlee.com/install-pyspark-windows-jupyter/

Cervical Cancer Risk Classification. (2022). Retrieved 1 July 2022, from https://www.kaggle.com/datasets/loveall/cervical-cancer-risk-classification/code

Drop column in pyspark – drop single & multiple columns - DataScience Made Simple. (2022). Retrieved 1 July 2022, from https://www.datasciencemadesimple.com/drop-column-in-pyspark-drop-single-multiple-columns/#

Wolff, R. (2022). 5 Types of Classification Algorithms in Machine Learning. Retrieved 1 July 2022, from https://monkeylearn.com/blog/classification-algorithms/

Watson, A. (2022). How to remove null values in Tableau - TAR Solutions. Retrieved 1 July 2022, from https://tarsolutions.co.uk/blog/tableau-hide-null-values/

Get Started with Calculations in Tableau. (2022). Retrieved 1 July 2022, from https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_create.htm

Holland, L. (2022). The Data School - Making a violin plot in Tableau. Retrieved 1 July 2022, from https://www.thedataschool.co.uk/liam-holland/understanding-and-making-violin-plot-in-tableau

pyspark, d. (2022). drop column based on condition pyspark. Retrieved 1 July 2022, from https://stackoverflow.com/questions/70393826/drop-column-based-on-condition-pyspark

9. APPENDIX

PySpark Code Syntax

```
#Importing PySpark libraries
import findspark
findspark.init()
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, SQLContext
import pandas as pd

spark = SparkSession.builder.master("local[*]").getOrCreate()

#Test the spark
df = spark.sql('''select 'chy' as sweetpea ''')
df.show()

spark = SparkSession.builder.appName("Cervical Cancer
Classification").getOrCreate()

#Load Dataset
file_loc_csv = 'kag_risk_factors_cervical_cancer.csv'
cervical = spark.read.csv(file_loc_csv, inferSchema=True, header=True)

#Data type
print(type(cervical))

#Check number of rows
cervical.count()

#Check number of columns
len(cervical.columns)

#Show details of the top 2 rows
cervical.head(2)

#Print top 5 rows
cervical.show(5)

#Check datatypes
cervical.printSchema()



#DATA CLEANING

#Convert data type from string to float
from pyspark.sql.functions import col
cervical = cervical.select(*(col(c).cast("float").alias(c) for c in
cervical.columns))
cervical.printSchema()
```

```
# Replace invalid data '?' with null
import pyspark.sql.functions as F
from pyspark.sql.functions import col,when
cervical=cervical.select([when(col(c)=="?",None).otherwise(col(c)).alias(c)
for c in cervical.columns])
cervical.show(5)

#Remove all duplicate
cervical.dropDuplicates().show()

#Remove columns that have only null values
cervical.na.drop("all").show()

# checking the sum/count of null values in all columns
from pyspark.sql.functions import col, isnan, when, count

null_count = cervical.select([count(when(isnan(i)| col(i).isNull(),
i)).alias(i)for i in cervical.columns])
null_count.show()

null_count = null_count.collect()[0].asDict()
null_count

#Drop columns with over 20% null values i.e 0.2*856 = 171
cols_to_drop = [c for c,i in null_count.items() if i > 171]
cervical = cervical.drop(*cols_to_drop)
cervical.show(5)

#Fitting the other null values
#Importing relevant library
from pyspark.ml.feature import Imputer

#Check columns to fit
cols_to_fit = [c for c,i in null_count.items() if i > 0]
cols_to_fit

#Fitting the null values
imputer = Imputer(inputCols=['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'],
                outputCols=['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'])
model = imputer.fit(cervical)
cervical = model.transform(cervical)

#checking for null/nan values after fitting the data
null_count = cervical.select([count(when(isnan(i)| col(i).isNull(),
i)).alias(i)for i in cervical.columns])
null_count = null_count.collect()[0].asDict()
null_count

#Check new number of columns
len(cervical.columns)




#MACHINE LEARNING

#Import relevant PySpark Machine Learning Libraries
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import StringIndexer

from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.classification import RandomForestClassifier

from pyspark.mllib.evaluation import BinaryClassificationMetrics
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
#Checking the Biopsy data.
cervical.groupBy("Biopsy").count().show()


#Vector assembling
cols = cervical.columns
cols.remove("Biopsy")
print(cols)
assembler = VectorAssembler(inputCols = cols, outputCol = "features")
cervical.show()


#Tranforming the dataset
cervical = assembler.transform(cervical)
cervical.select("features",'Biopsy').show(truncate = False)


#apply Scaling to achieve better results
standardscaler =
StandardScaler().setInputCol("features").setOutputCol("Scaled_features")
cervical = standardscaler.fit(cervical).transform(cervical)


assembled_data = cervical.select("Scaled_features","Biopsy")
assembled_data.show()


#Splitting into train and test data
train_cervical,test_cervical = assembled_data.randomSplit([0.7,0.3])

train_cervical.show()
test_cervical.show()


#LOGISTIC REGRESSION
log_reg = LogisticRegression(labelCol = "Biopsy", featuresCol =
"Scaled_features", maxIter=40)
model = log_reg.fit(train_cervical)

model

prediction_test = model.transform(test_cervical)
prediction_test.show()


prediction_test.select("Biopsy","prediction").show(10)


#compute raw scores on the test set
predictionAndLabels = prediction_test.select("Biopsy","prediction").rdd
predictionAndLabels.collect()


metrics = BinaryClassificationMetrics(predictionAndLabels)
```

```
#Area under ROC curve
print("Area under ROC = %s" % metrics.areaUnderROC)


evaluator = MulticlassClassificationEvaluator(labelCol = "Biopsy",
predictionCol = "prediction", metricName = "accuracy")
accuracy_log_reg = evaluator.evaluate(prediction_test)
print("Logistic Reg Accuracy = ",accuracy_log_reg)



#NAIVEBAYES
naive_bayes = NaiveBayes(labelCol = "Biopsy", featuresCol =
"Scaled_features", smoothing = 1.0)

model = naive_bayes.fit(train_cervical)
model


#select example rows to display
prediction_test = model.transform(test_cervical)
prediction_test.show()

prediction_test.select("Biopsy","prediction").show(10)

predictionAndLabels = prediction_test.select("Biopsy","prediction").rdd

metrics = BinaryClassificationMetrics(predictionAndLabels)
#Area under ROC curve
print("Area under ROC = %s" % metrics.areaUnderROC)


#Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(labelCol = "Biopsy",
predictionCol = "prediction", metricName = "accuracy")
accuracy_naive_bayes = evaluator.evaluate(prediction_test)
print("Naive Bayes Accuracy = ",accuracy_naive_bayes)


#RANDOM FOREST CLASSIFIER

random_forest_classifier = RandomForestClassifier(labelCol = "Biopsy",
featuresCol = "Scaled_features", numTrees = 40)
model = random_forest_classifier.fit(train_cervical)
model

prediction_test = model.transform(test_cervical)
prediction_test.show()

prediction_test.select("Biopsy","prediction").show(10)


predictionAndLabels =
prediction_test.select("Biopsy","prediction").rdd.map(lambda row: row[0:])

metrics = BinaryClassificationMetrics(predictionAndLabels)
```

```
#Area under ROC curve
print("Area under ROC = %s" % metrics.areaUnderROC)

#Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(labelCol = "Biopsy",
predictionCol = "prediction", metricName = "accuracy")
accuracy_random_forest = evaluator.evaluate(prediction_test)
print("Random Forest Accuracy = ",accuracy_random_forest)


print("Accuracy of Logistics Reg = ",accuracy_log_reg)
print("Accuracy of Naive Bayes   = ",accuracy_naive_bayes)
print("Accuracy of Random Forest = ",accuracy_random_forest)




#KMEANS Clustering
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import StringIndexer

cervical.columns
cervical = cervical.select('Age',
 'Number of sexual partners',
 'First sexual intercourse',
 'Num of pregnancies',
 'Hormonal Contraceptives (years)',
 'STDs:HPV',
 'Biopsy')
cervical.show()

#Vector assembling
cols = cervical.columns
cols.remove("Biopsy")
assembler = VectorAssembler(inputCols = cols, outputCol = "features")
cervical.show()

#Tranforming the dataset
cervical = assembler.transform(cervical)
cervical.select("features",'Biopsy').show(truncate = False)

cervical.groupBy("Number of sexual partners").count().show()
cervical.groupBy("Biopsy").count().show()

kmeans = KMeans(featuresCol = "features", k =2)

model = kmeans.fit(cervical)
model


model.transform(cervical).groupBy("prediction").count().show()
```

```
predictions = model.transform(cervical)
predictions.show()
predictions.groupBy("Biopsy","prediction").count().show()
```

Spark Implementation

After downloading Spark, Next is the execution of Hadoop binary for windows OS – winutils.exe – from Steve Loughran's GitHub repo.

steveloughran / winutils Public

Notificati

Code | Issues 3 | Pull requests | Actions | Projects | Wiki | Security | Insights

master | 1 branch | 3 tags | Go to file | Code

steveloughran link to cdarlint/winutils ... d4f7151 on 1 Aug 2019 25 commits

hadoop-2.6.0/bin	Add Hadoop-2.6.0/HDP-2.2 windows binaries	7 years ago
hadoop-2.6.3/bin	add gpg2 signatures	7 years ago
hadoop-2.6.4	add 2.6.4 and 2.7.1 windows binaries	6 years ago
hadoop-2.7.1	add 2.6.4 and 2.7.1 windows binaries	6 years ago
hadoop-2.8.0-RC3/bin	sign Hadoop artifacts	5 years ago
hadoop-2.8.1	sign Hadoop artifacts	5 years ago
hadoop-2.8.3/bin	Windows binaries for hadoop-2.8.3	5 years ago
hadoop-3.0.0/bin	Hadoop 3.0.0 windows binaries; off the release 3.0 tag, patched with ...	5 years ago

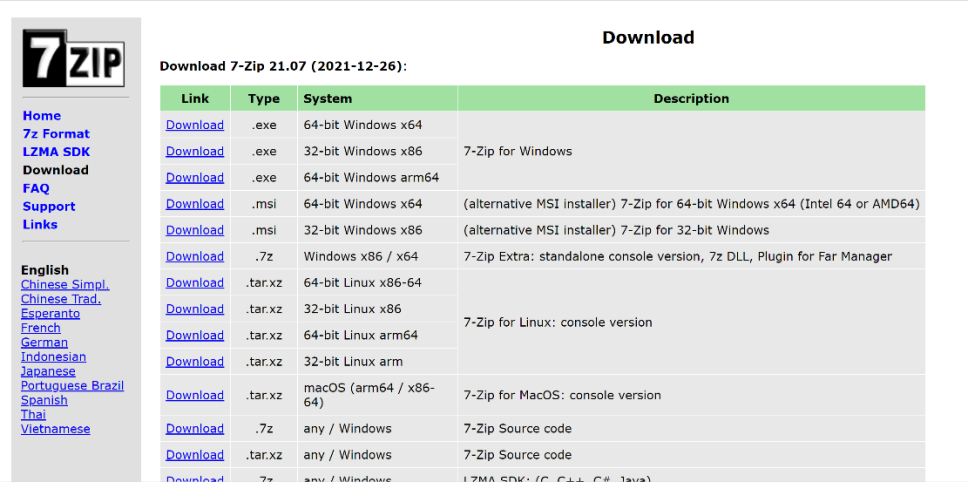
JDK 11 software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE.

JDK 11.0.15.1 checksum

Linux | macOS | Solaris | Windows

Product/file description	File size	Download
x64 Installer	140.41 MB	jdk-11.0.15.1_windows-x64_bin.exe
x64 Compressed Archive	158.1 MB	jdk-11.0.15.1_windows-x64_bin.zip

7-zip for Windows was downloaded and installed to unpack the Hadoop .gtz file on windows.



7ZIP

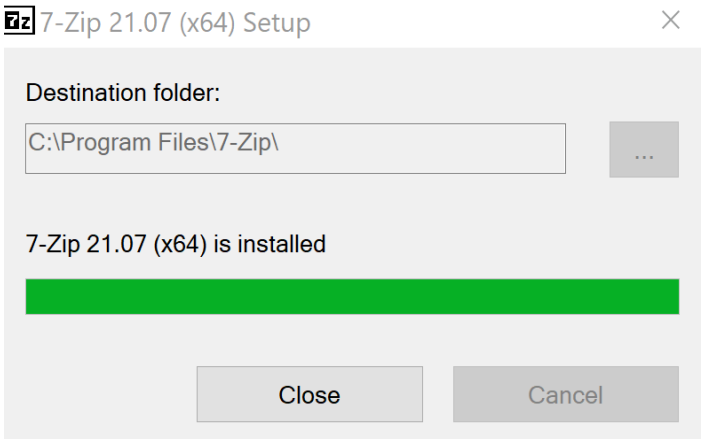
Home
7z Format
LZMA SDK
Download
FAQ
Support
Links

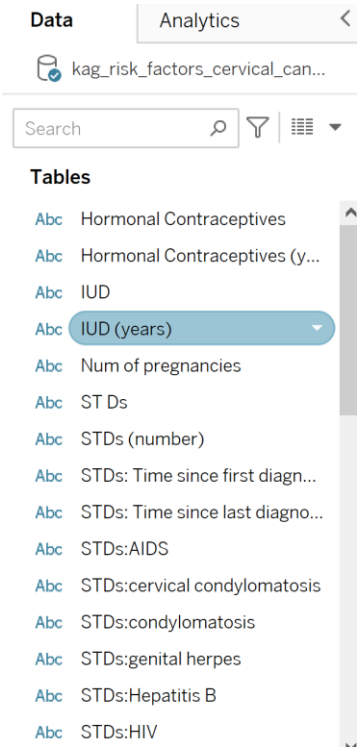
English
Chinese Simpl.
Chinese Trad.
Esperanto
French
German
Indonesian
Japanese
Portuguese Brazil
Spanish
Thai
Vietnamese

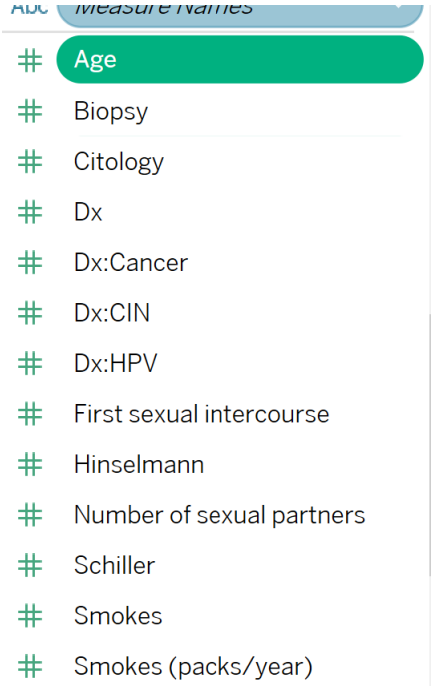
Download

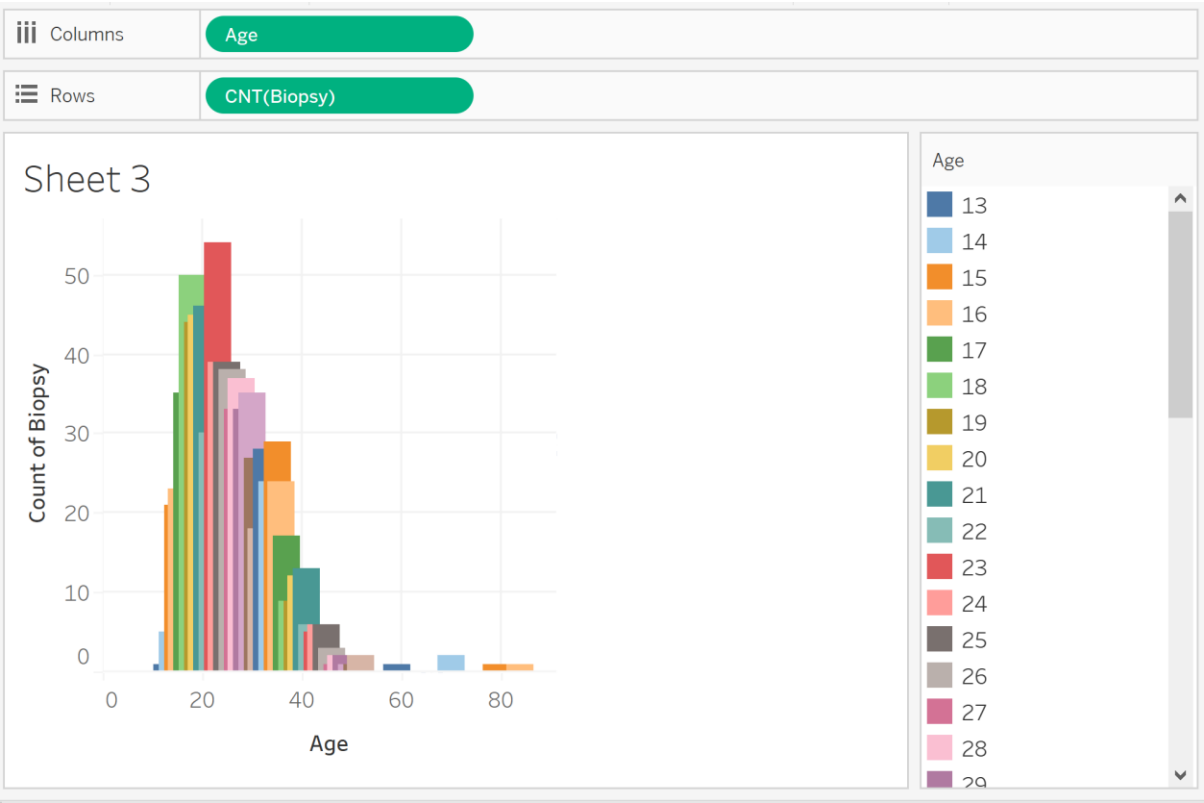
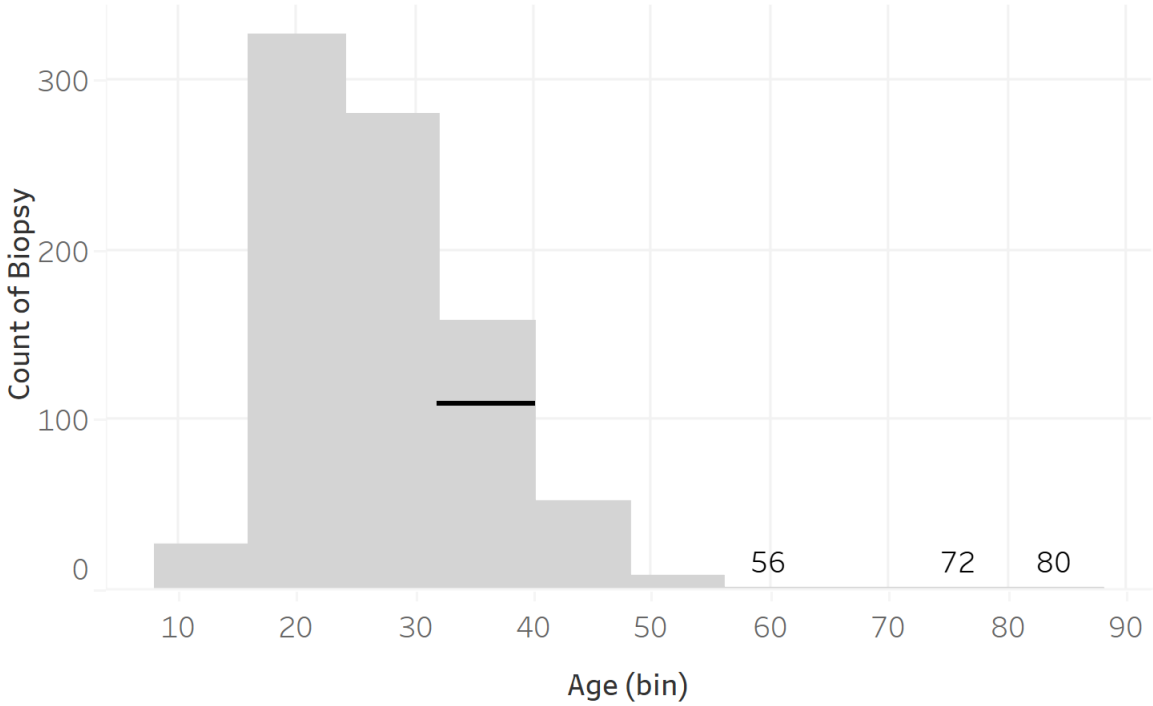
Download 7-Zip 21.07 (2021-12-26):

Link	Type	System	Description
Download	.exe	64-bit Windows x64	7-Zip for Windows
Download	.exe	32-bit Windows x86	
Download	.exe	64-bit Windows arm64	
Download	.msi	64-bit Windows x64	(alternative MSI installer) 7-Zip for 64-bit Windows x64 (Intel 64 or AMD64)
Download	.msi	32-bit Windows x86	(alternative MSI installer) 7-Zip for 32-bit Windows
Download	.7z	Windows x86 / x64	7-Zip Extra: standalone console version, 7z DLL, Plugin for Far Manager
Download	.tar.xz	64-bit Linux x86-64	7-Zip for Linux: console version
Download	.tar.xz	32-bit Linux x86	
Download	.tar.xz	64-bit Linux arm64	
Download	.tar.xz	32-bit Linux arm	
Download	.tar.xz	macOS (arm64 / x86-64)	7-Zip for MacOS: console version
Download	.7z	any / Windows	7-Zip Source code
Download	.tar.xz	any / Windows	7-Zip Source code
Download	.7z	any / Windows	LZMA SDK: (C, C++, C#, Java)




Data
Analytics
kag_risk_factors_cervical_can...
Search
Tables
Abc Hormonal Contraceptives
Abc Hormonal Contraceptives (y...
Abc IUD
Abc IUD (years)
Abc Num of pregnancies
Abc ST Ds
Abc STDs (number)
Abc STDs: Time since first diagn...
Abc STDs: Time since last diagno...
Abc STDs:AIDS
Abc STDs:cervical condylomatosis
Abc STDs:condylomatosis
Abc STDs:genital herpes
Abc STDs:Hepatitis B
Abc STDs:HIV


Age
Biopsy
Citology
Dx
Dx:Cancer
Dx:CIN
Dx:HPV
First sexual intercourse
Hinselmann
Number of sexual partners
Schiller
Smokes
Smokes (packs/year)



Biopsy according to Age