Cervical cancer is ranked as the most frequent cancer in women next to breast cancer across worldwide. Prediction of indicating cervical cancer's schiller test output by using the various factors that cause the disease. Data used here is obtained from 'Hospital Universitario de Caracas' in Caracas, Venezuela by UCI in 2017.

GROUP – 5
PREDICTION OF INDICATORS OF CERVICAL CANCER USING SCHILLR TEST

TEAM MEMEBERS:
DINESHKUMAR M
HARIPRAKASHAM K
MIRTHULA BABU
SUJITHA GANESH

Guidance:
Anjana Agrawal

GREAT LAKES INSTITUTE OF MANAGEMENT

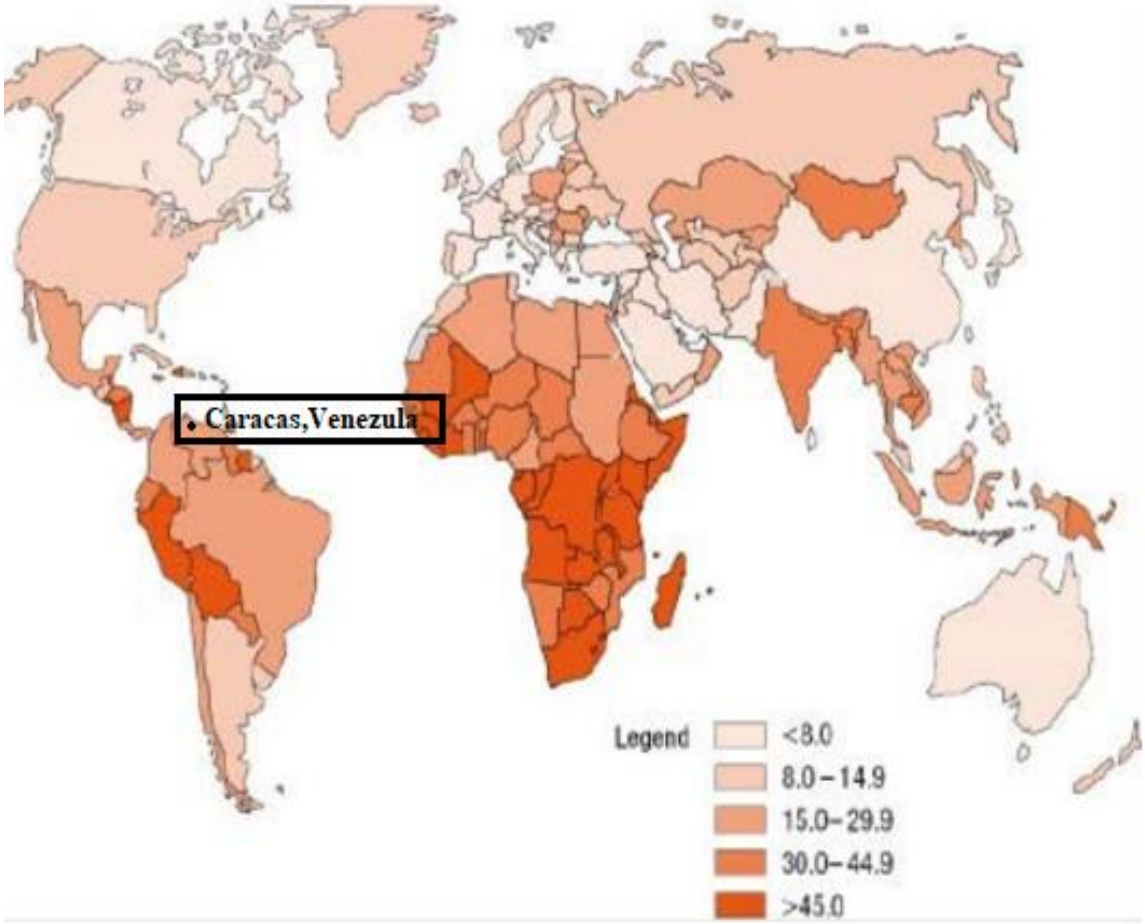
TABLE OF CONTENTS

1.INTRODUCTION

Cervical cancer occurs when the cells of the cervix grow abnormally and invade other tissues and organs of the body. When it is invasive, this cancer affects the deeper tissues of the cervix and may have spread to other parts of the body (metastasis), most notably the lungs, liver, bladder, vagina, and rectum. It is possible to prevent or detect & treat early cervical cancer since it is slow-growing.

World Health Organization (WHO) calls cancer as a generic term for a large group of diseases that can affect any part of the body. Sometimes it could be the cause of loss of patients. Cancer mortality can be reduced if cases are detected and treated early. In this fact, it is important to determine someone has highly cancer risk by using a survey. In this study, a classification of patients due to their answers to a survey has been done to determine someone who has highly cervical cancer risk. The dataset has been obtained from the dataset archive belongs to the University of California, Irvine.

Distribution of Global Burden of Cervical Cancer: Age-standardized incidence rates per 100,000 women.



CERVICAL CANCER MORTALITY WORLDWIDE

- 4th cause in female cancer deaths.
- 2nd most common cancer deaths in women aged 15-44 years.
- 2,65,672 annual deaths approximately due to cervical cancer.
- 728 women die every day approximately.
- 30 women die every hour approximately.
- 1 woman dies every 2 minutes approximately.

SYMPTOMS OF CERVICAL CANCER

As in many cancers, you may have no signs or symptoms of cervical cancer until it has progressed to a dangerous stage. They may include:

- Pain, when the cancer is advanced.
- Abnormal vaginal bleeding (other than during menstruation).
- Abnormal vaginal discharge.
- Pelvic pain.
- Kidney failure due to a urinary tract or bowel obstruction, when the cancer is advanced.

CERVICAL CANCER STAGE	SURVIVAL RATE
STAGE 0 (Early stage)	>90 %
STAGE I	80-93%
STAGE II	58-63%
STAGE III	32-35 %
STAGE IV	<16%

2.DATASET

2.1 DATASET INFORMATION

The dataset was collected at 'Hospital Universitario de Caracas' in Caracas, Venezuela. The dataset comprises demographic information, habits, and historic medical records of 858 patients. Several patients decided not to answer some of the questions because of privacy concerns (missing values). This dataset focuses on the prediction of indicators/diagnosis of cervical cancer.

2.2 DATSET ATTRIBUTES

Attribute name	Type
Age	int
Number of sexual partners	int
First sexual intercourse (age)	int
Number of pregnancies	int
Smokes	bool
Smokes (years)	int
Smokes (packs/year)	int
Hormonal Contraceptives	bool
Hormonal Contraceptives (years)	int
IUD	bool
IUD (years)	int
STDs	bool
STDs (number)	int
STDs: condylomatosis	bool
STDs: cervical condylomatosis	bool
STDs: vaginal condylomatosis	bool
STDs: vulvo-perineal condylomatosis	bool
STDs: syphilis	bool

Attribute name	Type
STDs: pelvic inflammatory disease	Bool
STDs: genital herpes	Bool
STDs: molluscum contagiosum	Bool
STDs: AIDS	Bool
STDs: HIV	Bool
STDs: Hepatitis B	Bool
STDs: HPV	Bool
STDs: Number of diagnosis	int
STDs: Time since first diagnosis	int
STDs: Time since last diagnosis	int
Dx: Cancer	Bool
Dx: CIN	Bool
Dx: HPV	Bool
Dx	Bool
Hinselmann: target variable	Bool
Schiller: target variable	Bool
Cytology: target variable	Bool
Biopsy: target variable	Bool

The following are the description of independent and the dependent attributes:

1. **Age** - It indicates the age of a woman. It is expressed in terms of numerical values
2. **Number of sexual partners** – It indicates the total number of sexual partners encountered. It is expressed in terms of numerical values.
3. **First sexual intercourse**- It indicates the age of a woman when she had her first sexual intercourse. It is expressed in terms of the count.
4. **Number of pregnancies** – It indicates the total number of times the woman got pregnant. It is expressed in terms of the total count.
5. **Smokes**- It indicates whether the person smokes or not. It is expressed in terms of zeros (does not smoke) and ones(smokes).
6. **Smokes (years)-** It indicates the total number of years for which the woman is smoking. It is expressed in terms of total count.
7. **Smokes (packs/year)-** It indicates the total number of packets of cigarettes per year the woman smokes. It is expressed in terms of numbers.

8. **Hormonal Contraceptives -** It indicates whether the patient uses hormonal contraceptives or not.
9. **Hormonal Contraceptives (years)** – It indicates that for how many years the contraceptive method was used. It was in expressed in terms of total number of years.
10. **Intra-Uterine Device**- It indicated where the intrauterine contraceptive device was used or not. It was expressed in terms of zeros(did not used IUD) and ones(used IUD).
11. **IUD (years)** – It indicated that for how many years the IUD was used. It is expressed in terms of the total number of years.
12. **STDs -** It indicates the presence of Sexually Transmitted Diseases. It is expressed in terms of zeroes and ones.
13. **STDs (number)** – It indicates the total number of sexually transmitted disease present with the patient. It is expressed in terms of numbers.
14. **STDs:condylomatosis** – It indicates the presence of Condylomatosis with the patient.
15. **STDs:cervical condylomatosis** – It indicates the presence of Cervical condylomatosis.
16. **STDs:vaginal condylomatosis** – It indicates the presence of Vaginal condylomatosis.
17. **STDs:vulvo-perineal condylomatosis** – It indicates the presence of Vulvo-Perineal condylomatosis.
18. **STDs:syphilis** – It indicates the presence of Syphilis.
19. **STDs:pelvic inflammatory disease** - It indicates the presence of pelvic inflammatory disease.
20. **STDs:genital herpes** – It indicates the presence of Genital Herpes.
21. **STDs:molluscum contagiosum** – It indicates the presence of Molluscum Contagiosum.
22. **STDs:AIDS** – It indicates the presence of AIDS in the patient.
23. **STDs:HIV** – It indicates the presence of HIV in the patient.
24. **STDs:Hepatitis B** – It indicates the presence of Hepatitis B in the patients.
25. **STDs:HPV** – It indicates the presence of HPV in the patients.
26. **STDs: Number of diagnosis** – It indicate the total number of times the STDs have been diagnosed.
27. **STDs: Time since first diagnosis** – It indicates the total number of years since the first diagnose.

28. **STDs: Time since last diagnosis** – It indicates the total number of years elapsed since the last diagnose.
29. **Dx:Cancer** – It indicates the presence of Cancer after the diagnose.
30. **Dx:CIN** – It indicates the presence of Cervical intraepithelial neoplasia.
31. **Dx:HPV**- It indicates the presence of Human papillomaviruses.
32. **Dx** - It indicates the presence any one among cancer, CIN and HPV.
33. **SCHILLER** – It indicates the result of the test by Schiller's test is whether Positive or negative.

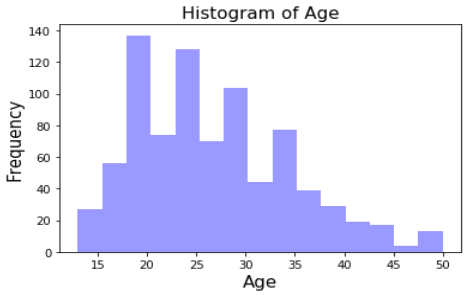
The four different target column represents the four different types of test to predict the cervical cancer. In this project, we are taking **one target column(schiller)** as per the problem statement and neglecting other three target column. So, the dataset taken is 33 attributes and 858 records.

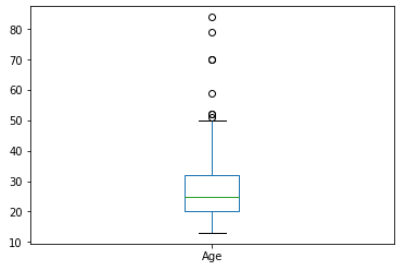
Schiller Test: Schiller's test or Schiller's Iodine test is a medical test in which iodine solution is applied to the cervix in order to diagnose cervical cancer. Visual Inspection with Lugol's Iodine (VILI), also known as "Schiller's Test". Schiller's test is named after Dr. Walter Schiller (1887 – 1960). Schiller's test is not specific for cervical cancer, as areas of inflammation, ulceration and keratosis may also not take up the stain.

3.EXPLORATORY DATA ANALYSIS:

3.1 DATASET ATTRIBUTES DESCRIPTION & ANALYSIS

AGE:





count	838
mean	26.8126
std	8.52921
min	13
25%	20
50%	25
75%	32
max	84

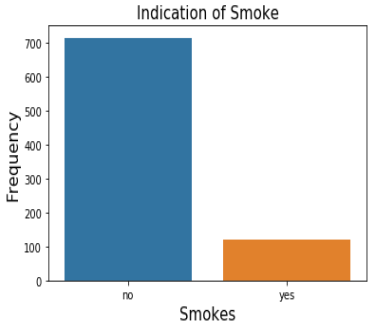
count	72
mean	29.5139
std	11.1051
min	16
25%	21
50%	28
75%	35
max	84

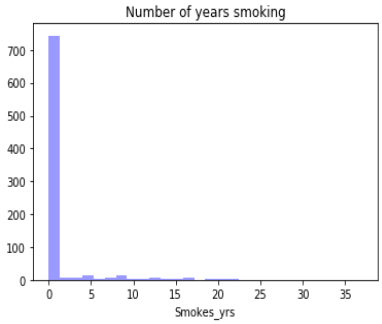
These 72 records are positive schiller records and the mean and median values of age are 29.5 and 28 respectively. The above age distribution of cervical cancer shows that the Venezuela cervical cancer standardized age is between 15-30. This five-point summary proves that the data is true representation of sample.

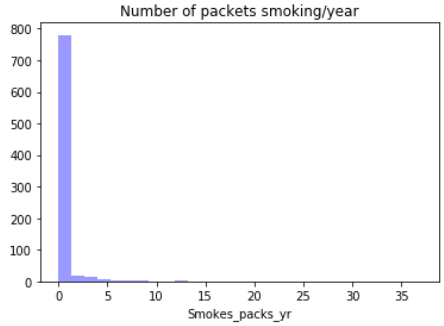
3.1.1. SMOKING HABIT ATTRIBUTES:

- Smokes – Patient having smoking habit or not.
- Smokes (years) – Number of years having smoking habit (0 for non-smokers).
- Smokes (packs/year) – Number of packets smoking per year (0 for non-smokers).

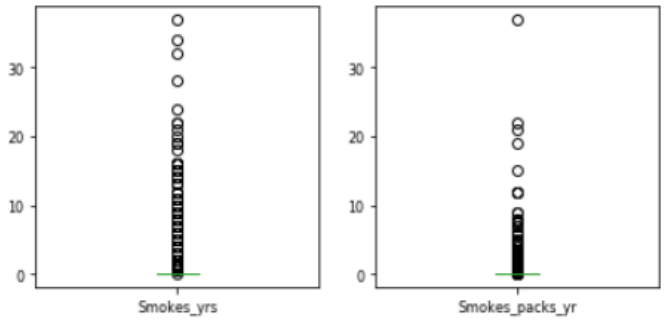
DISTRIBUTION:





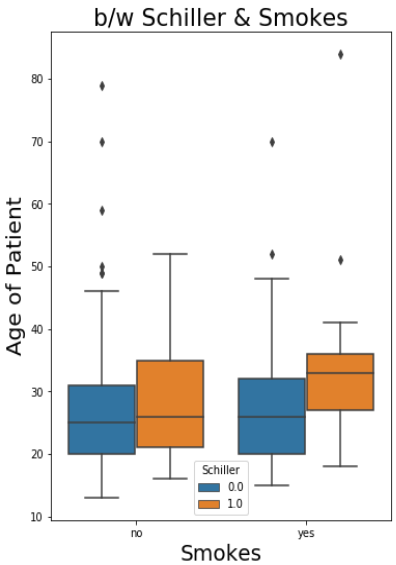


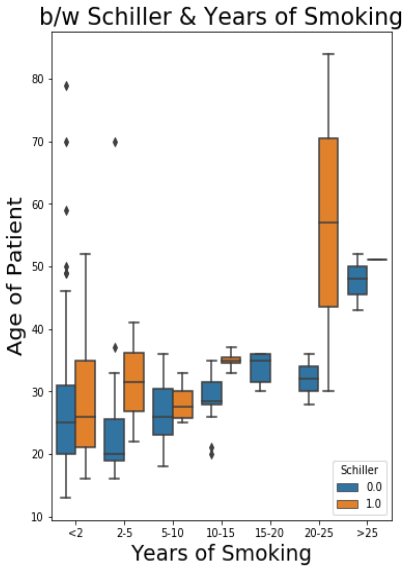
FIVE POINT SUMMARY

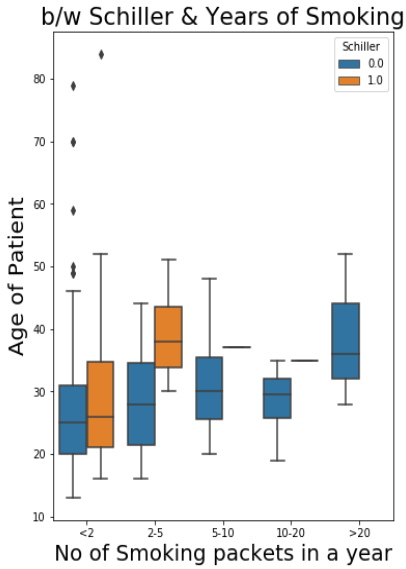


	Smokes_yrs	Smokes_packs_yr
count	838	838
mean	1.216784	0.450366
std	4.090836	2.228754
min	0	0
25%	0	0
50%	0	0
75%	0	0
max	37	37

RELATIONSHIP B/W SCHILLER AND SMOKING ATTRIBUTES







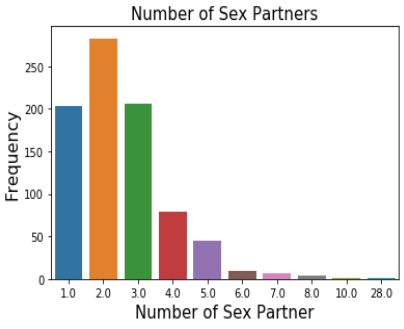
INFERENCES:

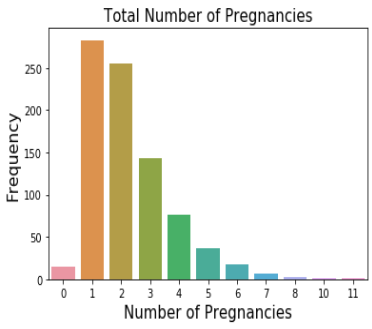
- The person who smokes over long period are prone to be tested as positive in Schiller test.
- Cancer can affect to non-smokers as well and even in the young age (around 22 years) the Schiller test can be tested as positive.
- The person who smoke for at least one year are more prone to be test as positive in schiller test.
- The person who smoke a greater number of packets a year with more age are prone to be tested as positive in Schiller test and the average age is found to be 30 years.

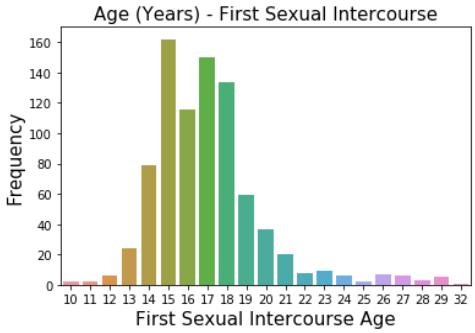
3.1.2. SEXUAL HABIT ATTRIBUTES

- Number of sexual partners – Total number of sexual partners.
- First sexual intercourse (age) – Age at which the patient had first sexual intercourse.
- Number of pregnancies – Number of times patient become pregnant.

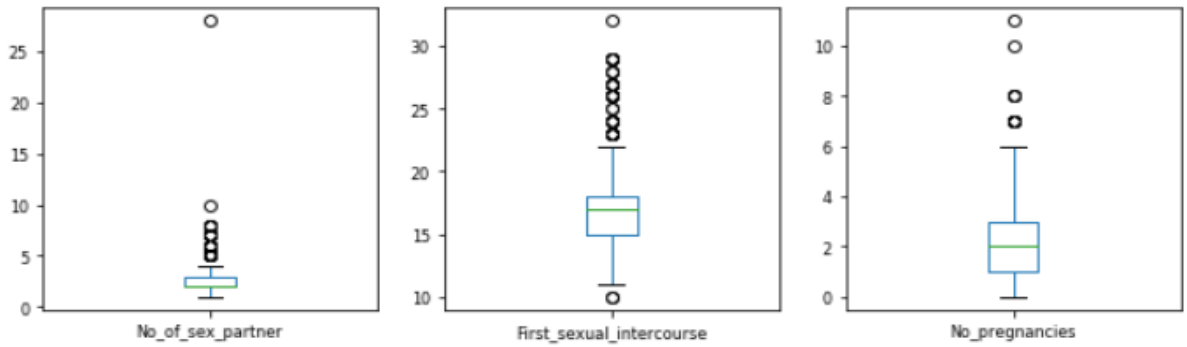
DISTRIBUTION:



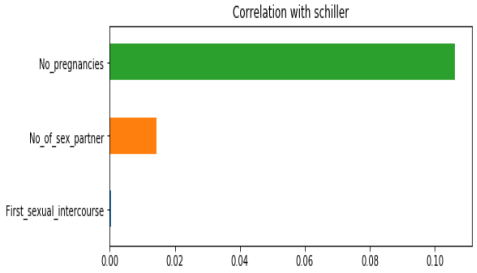




5 POINT SUMMARY:

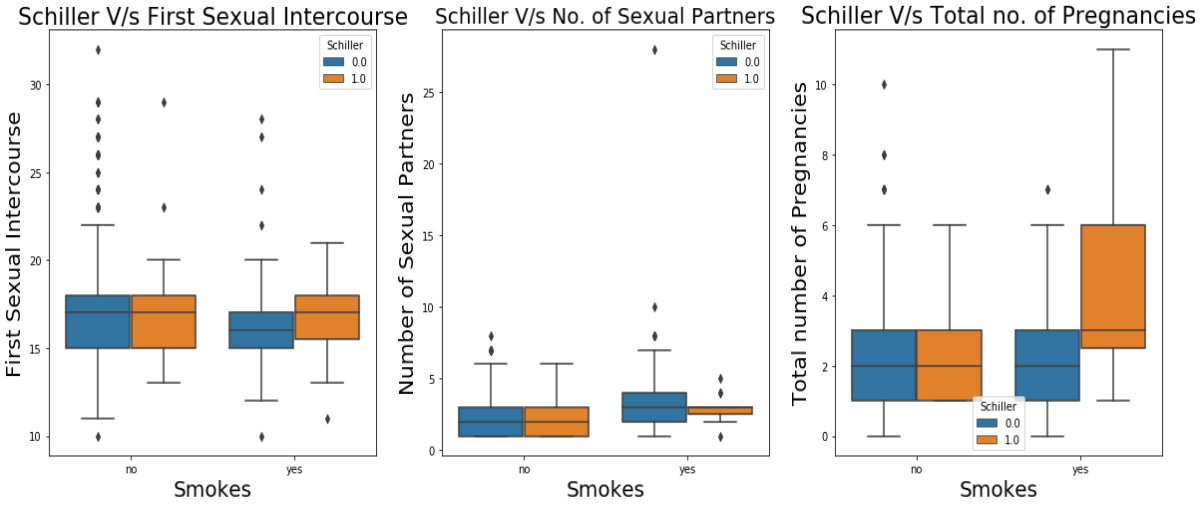


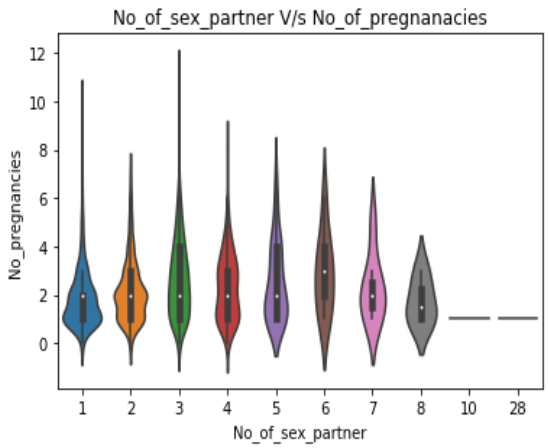
	First_sexual_intercourse	No_of_sex_partner	No_pregnancies
count	838	838	838
mean	16.844272	2.393198	2.263723
std	2.331009	1.100229	1.365524
min	10.5	1	0
25%	15	2	1
50%	17	2	2
75%	18	3	3
max	22.5	4.5	6

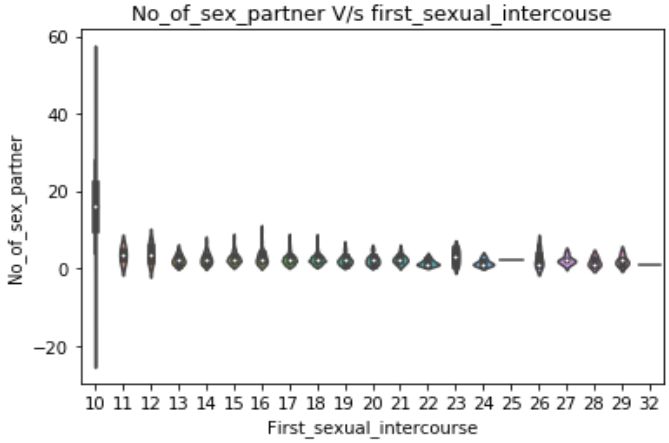


Five-point summary shows that there are some outliers in the each of the sexual attributes. The above chart explains that correlation between schiller and sexual attributes is very weak. Number of pregnancies is highly correlated from sexual attributes.

RELATION B/W SCHILLER AND SEXUAL ATTRIBUTES:







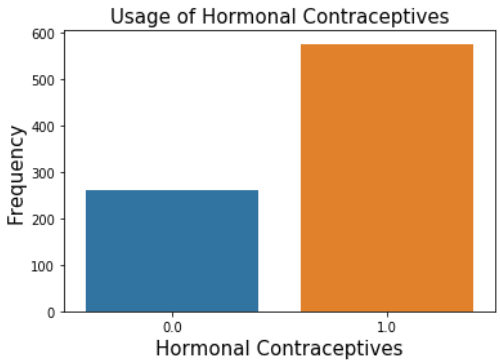
INFERENCES:

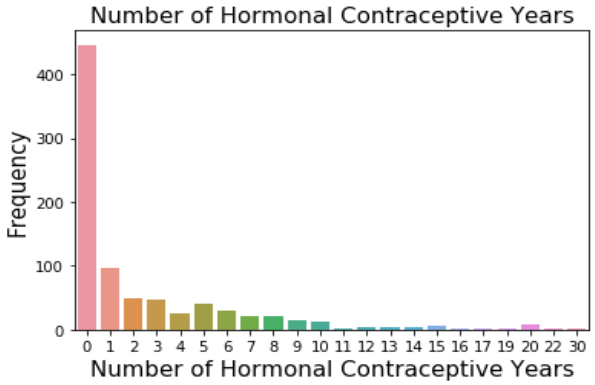
- Data is not normally distributed for all sex attributes and the data is right skewed.
- The maximum sexual partners per person is in the range of 1 to 3 and average age at which the patient had their first sexual intercourse is between 15 years and 19 years.
- Those who had sexual intercourse at very early age, have the highest number of sex partners.
- Those who smoke and the first sexual intercourse age between 15yrs and 18yrs are more prone to be test as Positive in Schiller test.
- The persons who have sexual partners between 1 and 3 are more prone to be tested as positive in Schiller test.
- The person who smokes and have higher number of pregnancies are more prone to be tested as positive in Schiller test.
- The patients who have sexual partners between 1 and 4 are getting pregnant more often. The number of sexual partners does not have any effect on the age at which they had their first sexual intercourse.

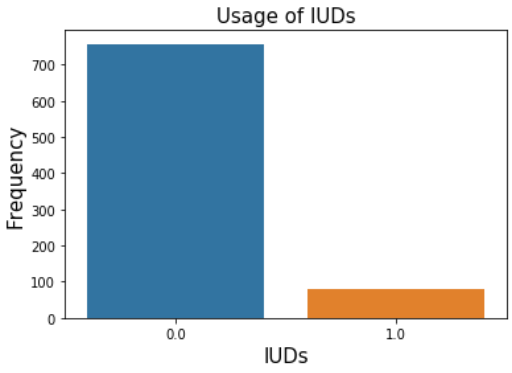
3.1.3. BIRTH CONTROL ATTRIBUTES

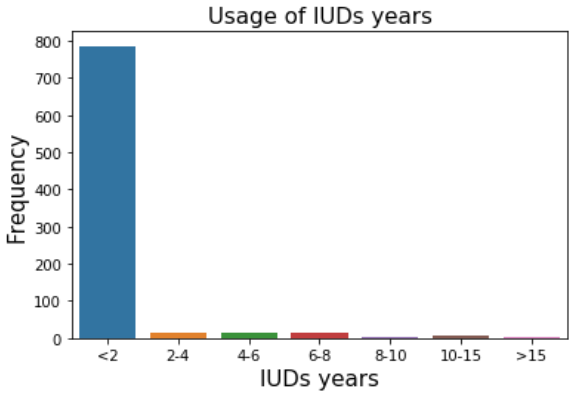
- Hormonal Contraceptives – Patient using Hormonal Contraceptives (Birth control methods that act on the endocrine system. Almost all methods are composed of steroid hormones).
- Hormonal Contraceptives (years) – Number of years patient using Hormonal contraceptives.
- IUD – Patient using IUD (A small, often T-shaped birth control device that is inserted into a woman's uterus to prevent pregnancy).
- IUD (years) – Number of years patient using IUD.

DISTRIBUTION:

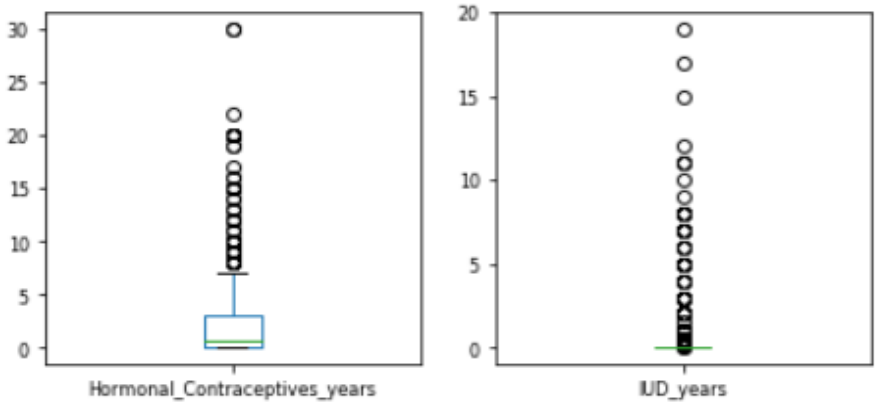




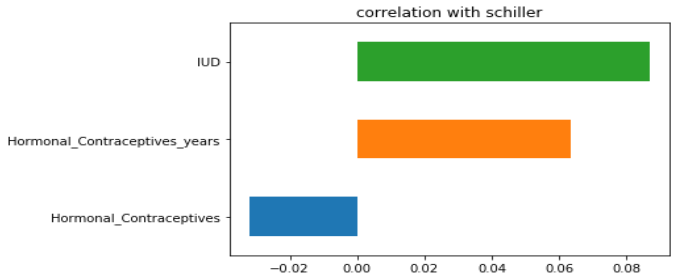


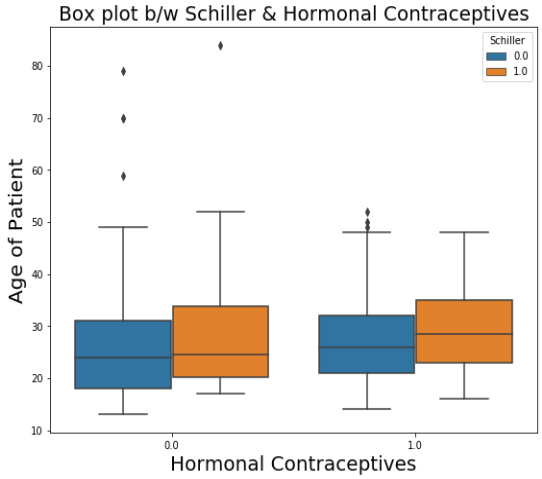


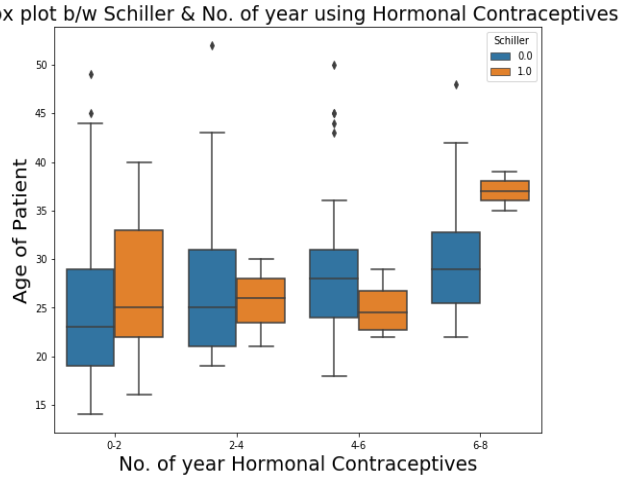
BOX PLOT:

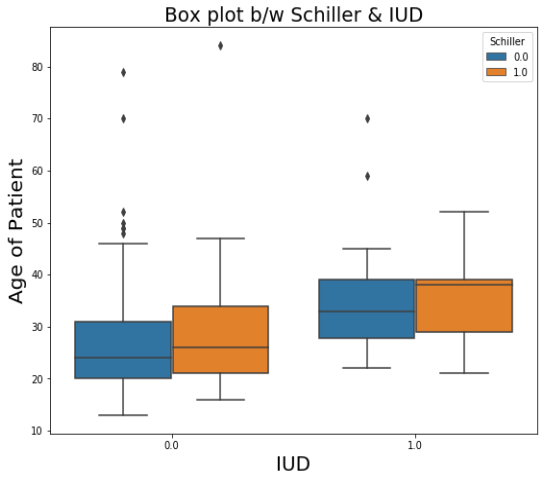


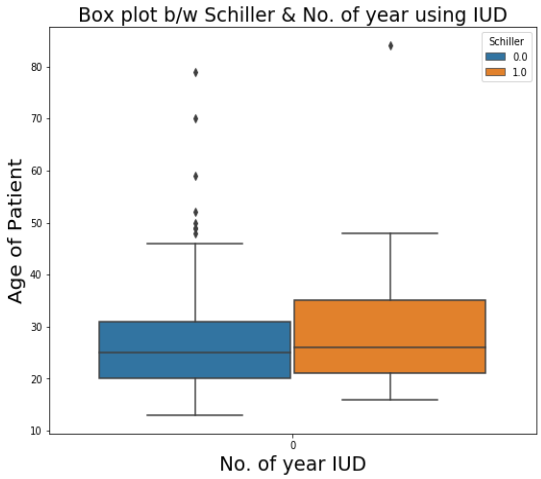
RELATION B/W SCHILLER AND BIRTH CONTROL ATTRIBUTES:











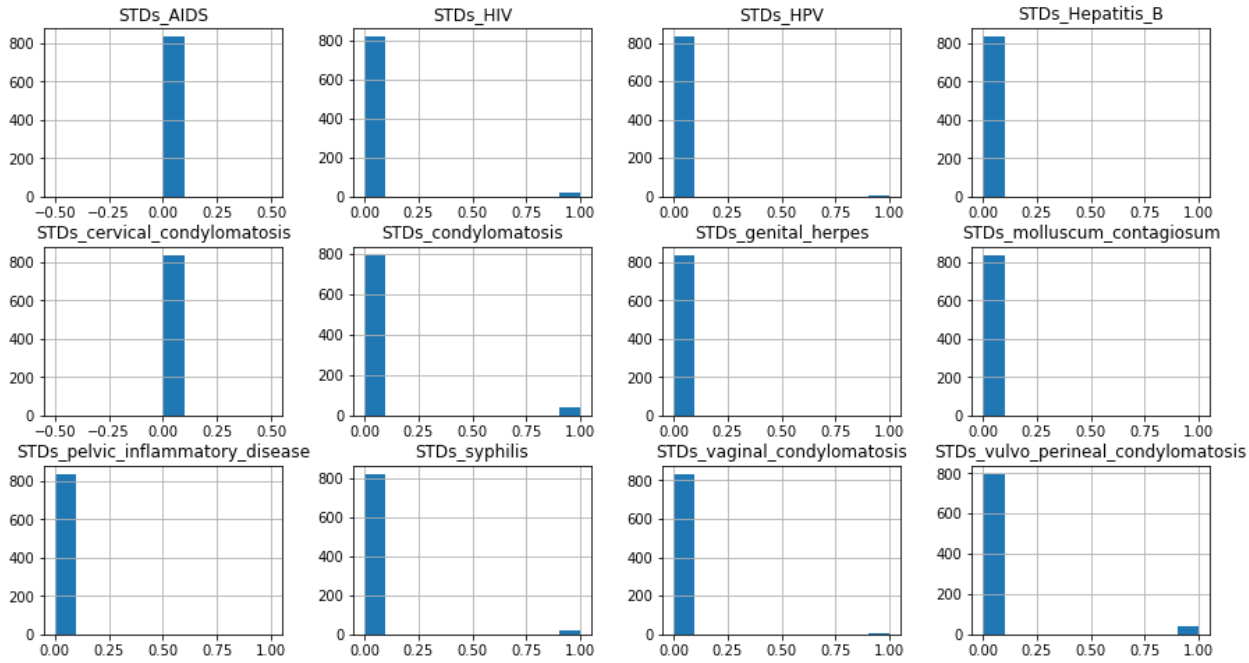
INFERENCES:

- We see that women prefer Hormonal Contraceptives when compared to the Intra Uterine Contraceptive Devices as it is safer than the latter.
- Most of the women have used the contraceptives in the period of less than 2 years.
- The persons who did not use the hormonal contraceptives and with more age are high in number, who show positive for schiller test
- The patients with 2 years of usage in hormonal contraceptives and the average age between 25yrs & 35yrs show positive for schiller test.
- The persons who did not use IUD and with age 25 years & 35 years, show positive for schiller test.

3.1.3. STDs ATTRIBUTES

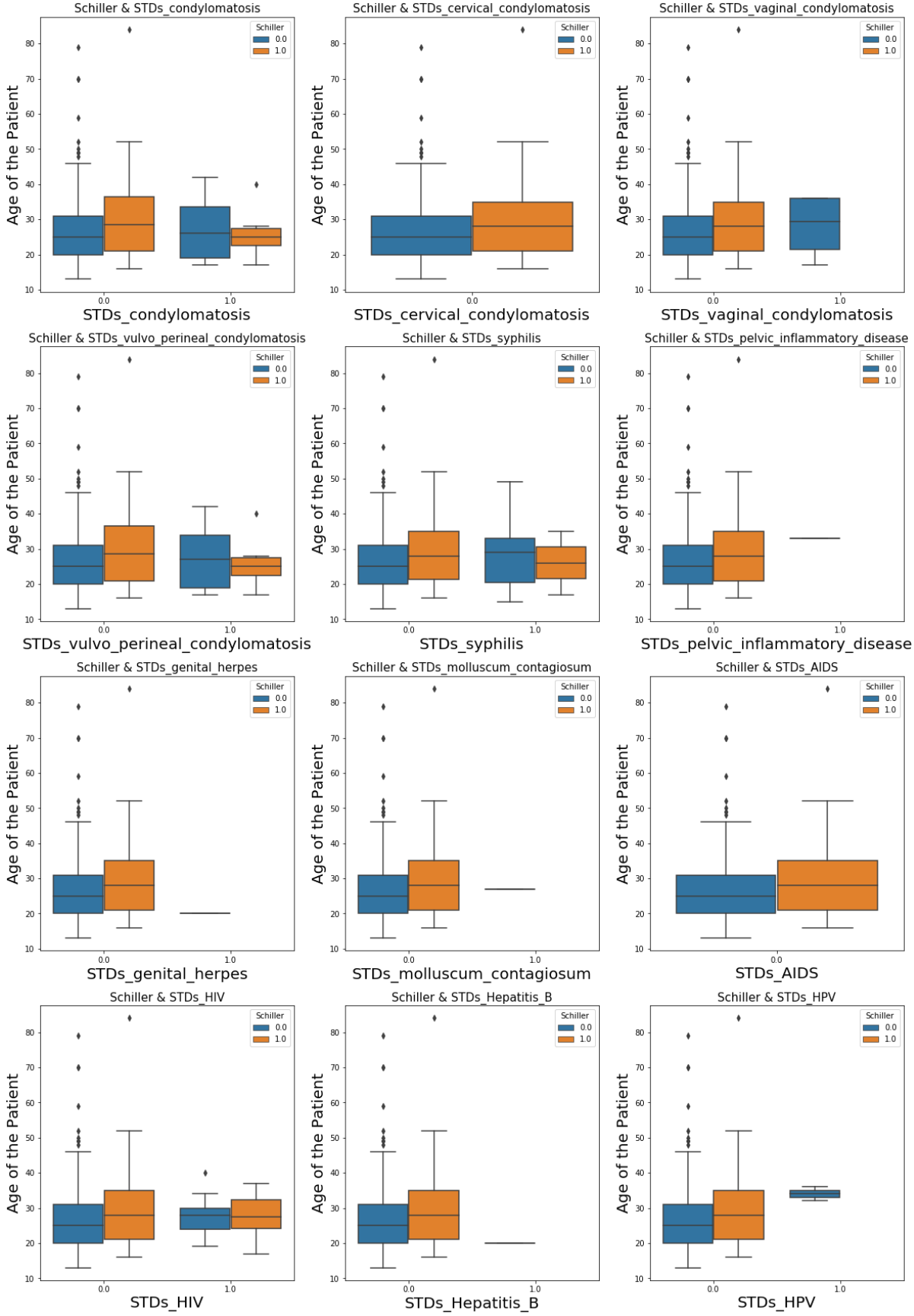
- Condylomatosis - A disease of sexual transmission caused by the virus of papilloma.
- Cervical condylomatosis - Type of Condylomatosis.
- Vaginal condylomatosis - Type of Condylomatosis.
- Vulvo-perineal condylomatosis - Type of Condylomatosis.
- Syphilis - A bacterial infection usually spread by sexual contact.
- Pelvic inflammatory disease - An infection of the female reproductive organs.
- Genital herpes - An infection marked by genital pain and sores.
- Molluscum contagiosum - A viral skin infection.
- AIDS - A disease which lowers the immunity.
- HIV - HIV causes AIDS and interferes with the body's ability to fight infections.
- Hepatitis B - A serious liver infection caused by the hepatitis B virus.
- HPV - An infection that causes warts in various parts of the body.

DISTRIBUTION:



Attribute	0 (%)	1(%)
STDs_condylomatosis	0.95	0.05
STDs_cervical_condylomatosis	1	0
STDs_vaginal_condylomatosis	0.995	0.005
STDs_vulvo_perineal_condylomatosis	0.951	0.049
STDs_syphilis	0.98	0.02
STDs_pelvic_inflammatory_disease	0.999	0.001
STDs_genital_herpes	0.999	0.001
STDs_molluscum_contagiosum	0.999	0.001
STDs_AIDS	1	0
STDs_HIV	0.98	0.02
STDs_Hepatitis_B	0.999	0.001
STDs_HPV	0.998	0.002

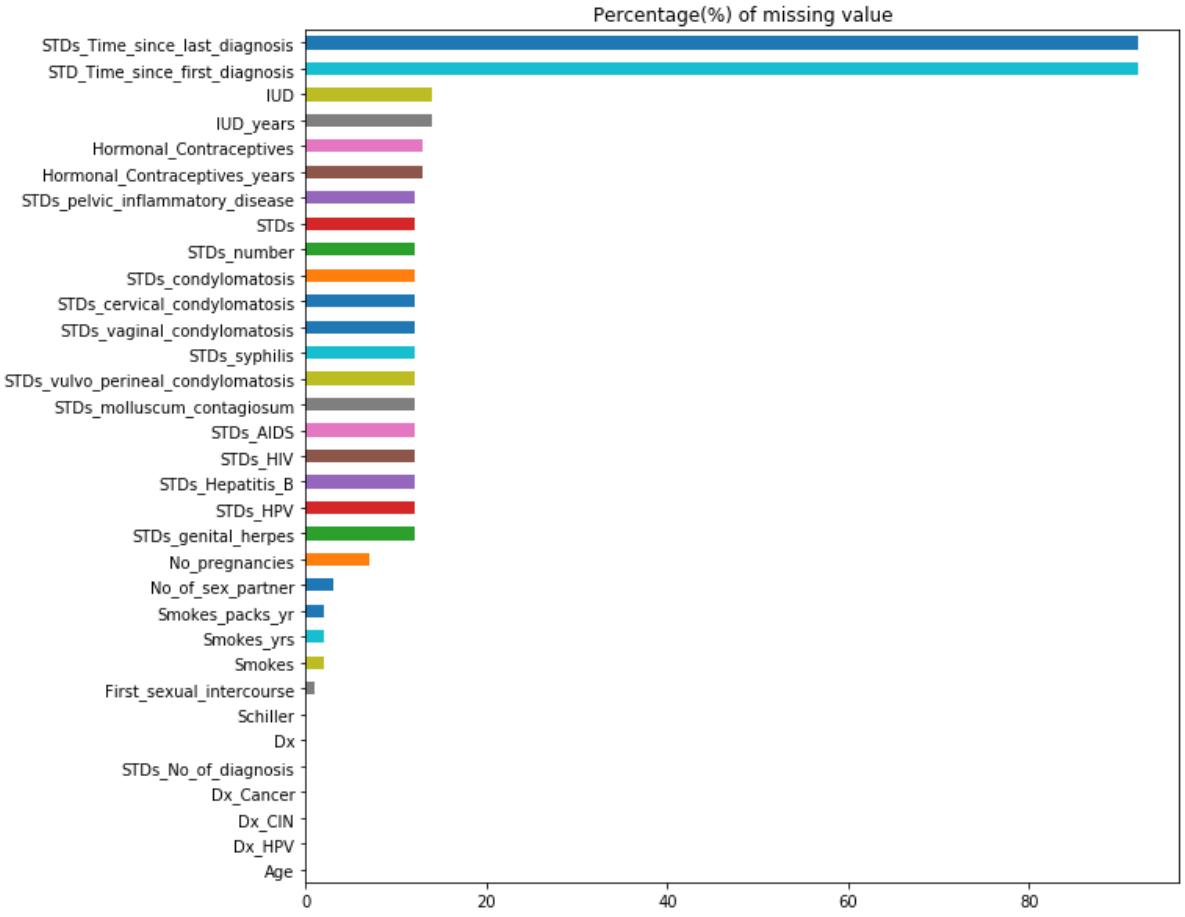
RELATION B/W STDS ATTRIBUTES AND SCHILLER:



INFERENCES:

- ❖ We can see from the distribution plot that STD cervical condolymatosis and STD AIDS have no effect on Schiller Test and hence variables will be of least importance when predicting the Cancer using Schiller test.
- ❖ In all the plots we can clearly the person with more age are prone to be tested as Positive in Schiller test.
- ❖ We cannot predict the effect of individual STDs on schiller test due to class imbalance in the dataset of each STD attributes.

3.2. TREATMENT OF MISSING VALUES



The amount of missing value percentage from the above graph shows that the attributes STDs_Time_since_last_diagnosis and STDs_Time_since_first_diagnosis has more than 80 percent of null values and hence the two columns are removed from the dataset. Also, the columns Smokes and First Sexual Intercourse whose rows having the least missing values are removed from the dataset.

Missing values are imputed by two methods:

i) Imputing missing values with median value
ii) Imputing missing values using machine learning model

3.2.1. IMPUTATION WITH MEDIAN AND MODE VALUES

Generally, the numeric and categorical variables are replaced with Median and Mode values respectively.

Categorical variables

Smokes	Hormonal_Contraceptives
STDs_AIDS	STDs_cervical_condylomatosis
STDs_HIV	STDs_vaginal_condylomatosis
STDs_Hepatitis_B	STDs_vulvo_perineal_condylomatosis
STDs_HPV	STDs_syphilis
Dx_Cancer	STDs_pelvic_inflammatory_disease
Dx_CIN	STDs_genital_herpes
Dx_HPV	STDs_molluscum_contagiosum
Dx	STDs_condylomatosis

Numeric variables

Age
No_of_sex_partner
First_sexual_intercourse
No_pregnancies
Smokes_yrs
Smokes_packs_yr
STDs_No_of_diagnosis.
Hormonal_contraceptive_years

The question marks (?) have been replaced with NAN values. We calculate the median and mode of all the features and then the NANs are replaced with median and mode values respectively for each of the above variables. This is an approximation which can add variance to the data set. But the loss of data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above two approximations are a statistical approach of handling the missing vales.

3.1.2. IMPUTATION USING MACHINE LEARNING ALGORITHMS:

Using the features which do not have missing values, we can predict the nulls with the help of machine learning algorithm. We used regression techniques for continuous target variables and classification techniques for categorical target variables. This method may result in better accuracy, unless a missing value is expected to have a very high variance.

We took each of the columns individually and treated them. We first took the variables which had maximum missing values from the remaining variables. The variable with maximum missing value was taken and a model was performed on the column, keeping the other variables as dependent variables. We did one hot encoding for the categorical variables. We predicted the missing values for that dependent column and replaced the missing values with predicted value. This iteration process of modelling the predicting and replacing the missing values was until there was no missing values in any one of the columns.

Now we have a dataset where the missing values were replaced by the predicted values from the models.

3.3. DETECTION AND TREATMENT OF OUTLIERS

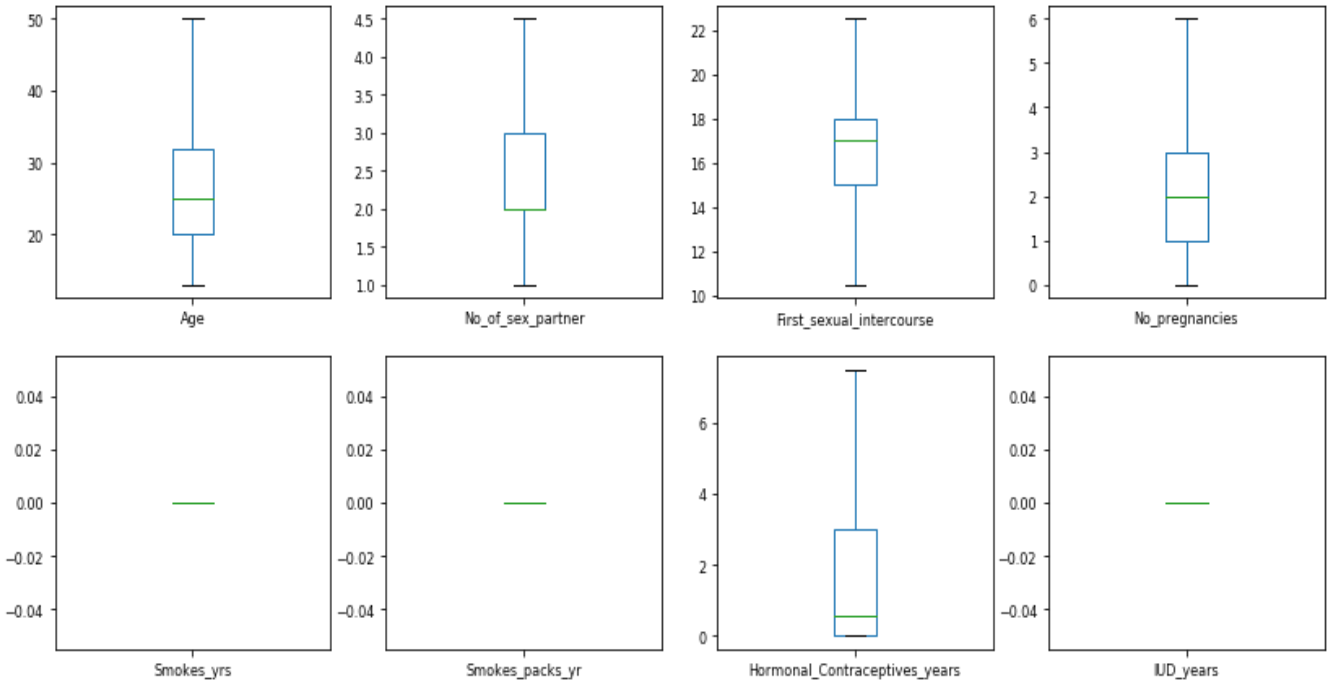
A box plot (box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables. The box shows the five-point summary of each numerical attributes. The above-mentioned numerical columns outliers are treated by using the IQR(Inter-quartile range).

IQR = 75% quartile – 25% quartile.

Range = (25% quartile -1.5*IQR) to (75% quartile +1.5*IQR).

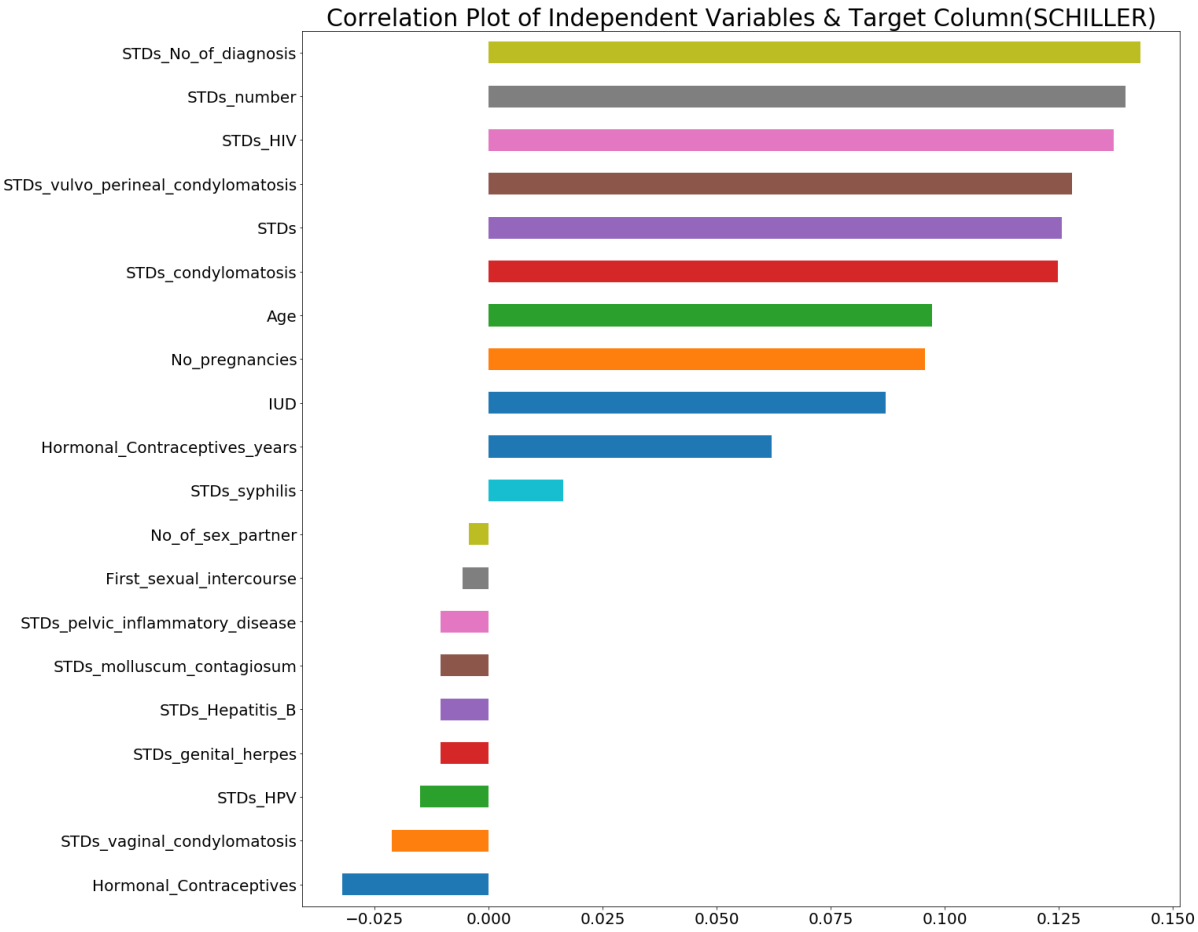
Range of Age b/w 2.0 and 50.0
Range of number of sex partner b/w 0.5 and 4.5
Range of first sexual intercourse age b/w 10.5 and 22.5
Range of number of pregnancies b/w -2.0 and 6.0
Range of Smoking years b/w 0.0 and 0.0
Range of smoking packets per year b/w 0.0 and 0.0
Range of Hormonal contraceptive usage years b/w -4.5 and 7.5
Range of IUD usage years b/w 0.0 and 0.0

AFTER OUTLIER TREATMENT



3.4. CORRELATION ANALYSIS:

Correlation analysis measures the statistical relationship between two different variables. The result will show how the change in one parameter would impact the other parameter.



The independent attributes correlation with the target column is not very strong and the maximum correlation value is around 0.15. STDs, age, Number of pregnancies are the highly influencing attributes of schiller column.

Chi-Square Analysis:

A chi-square test for independence compares two variables in a contingency table to see if they are related.

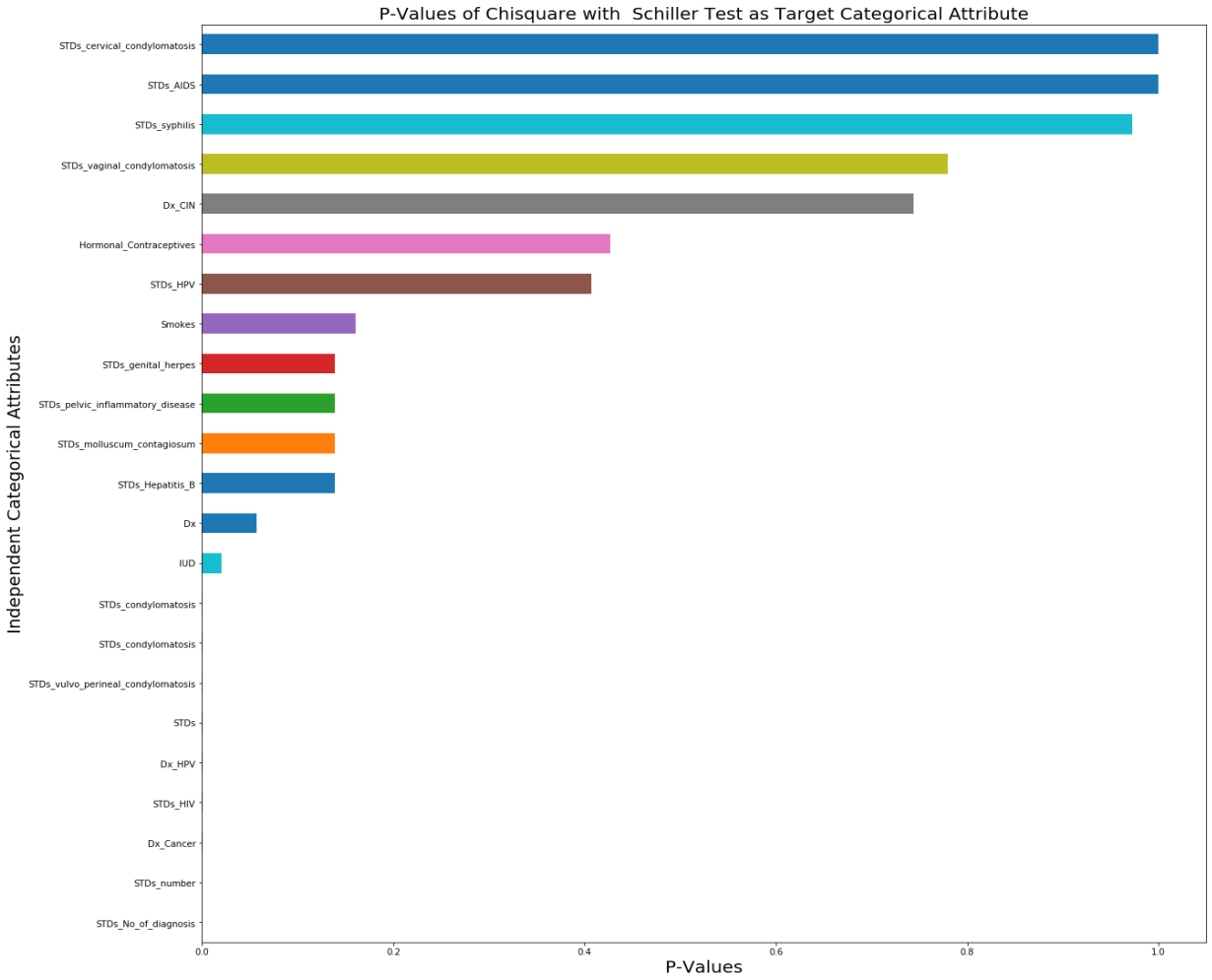
$$c^2 = \sum_{i=1}^{k} \left[\frac{(O_i - E_i)^2}{E_i} \right]$$

here “O” is the Observed value, “E” is the expected value and “i” is the “ith” position in the table.

A chi square test will give you a p-value. The p-value will tell you if your test results are significant or not.

P<0.05 – Two variables are dependent.

p>0.05 – Two variables are independent.



4.MODEL BUILDING

- The given problem statement is classification problem. So, four base classification algorithms are taken:
  - Logistic Regression
  - Decision Trees (CART)
  - Naïve Bayes.
  - K Nearest Neighbours.
- The precision and recall values were very low (below 0.30) and this could be caused by the heavy class imbalance in the data which is the ration 9:91.
- The class ratio is converted to the ratio of 40:60 using the over sampling of data of the minority class.

- Decision Tree Classifier is chosen as final base model as it given better metrics (accuracy, f1 score) than the other base models mentioned above. Feature selection is done using the Decision Tree Classifier itself.
- In order to improve the model performance, ensemble techniques are considered. We have considered the Random Forest Classifier as is the extension of Decision Tree.
- The model is finalised after confirming the hyper parameters which was determined by using Grid search Cross Validation Technique.

4.1. BASE MODEL COMPARISION

The given dataset is divided into train data and test data for each of the mentioned base models. Train dataset is used for fitting the model and test dataset is used to check how the fitted model predicts the test target column.

This is a classification problem, Base Classifier is imported. The model is fitted on the train independent attributes and dependent labels. Then the trained classifier/ model is used to predict the labels of the test attributes.

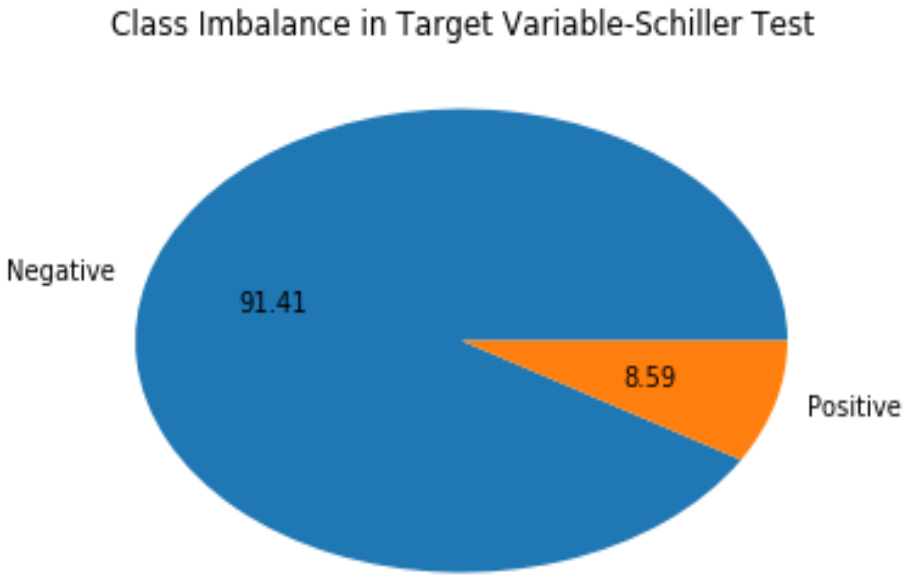
A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The matrix is generated after making predictions on the test data and then obtaining the values of True Positives, True Negatives, False Positives and False Negatives. The F1 score can be interpreted as a weighted average of the precision and recall.

The following table indicates the model performance of the different base models.

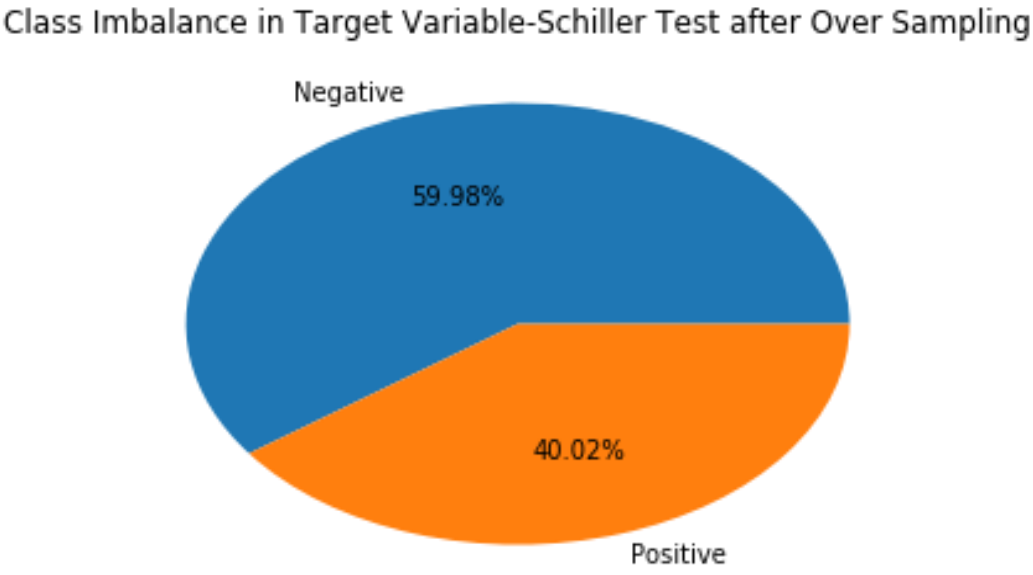
S.No	Classifier	Test Score	Recall	Precision	F1 Score
1	Decision Tree	0.845238	0.250000	0.222222	0.235294
2	K Nearest Neighbours	0.908730	0.041667	1.000000	0.080000
3	Naive Bayes	0.107143	0.958333	0.093117	0.169742
3	Logistic Regression	0.908730	0.000000	0.000000	0.000000

4.2 CLASS IMBALANCE TREATMENT

The target variable has 91.41% of negative and 8.59% of positive results. A simple way to fix imbalanced data-sets is simply to balance them, either by oversampling instance of minority class and under-sampling instance of majority class. The ideal range of class imbalance can be in ratio of 40:60.



The random over sampling is performed to the ratio of 40:60 and we get the following result. The total number of data points for the transformed data set after over sampling is 1277.



4.3 DECISION TREE CLASSIFIER AS BASE MODEL

The two base models Decision tree classifier and logistic regression are compared on the basis of metrics such as classification accuracy, precision, recall and f1score. This transformed data set is again split into train and test data. Model is trained using the train data and the metrics are calculated using the test data. The confusion matrix is again generated after making predictions on the test data and then obtaining the values of True Positives, True Negatives, False Positives and False Negatives. The F1 score can be interpreted as a weighted average of the precision and recall. The following table indicates the model performance of the two base models.

S.No	Classifier	Test Score	Recall	Precision	F1 Score
1	Logistic Regression	0.684896	0.408759	0.583333	0.480687
2	Decision Tree	0.937500	1.000000	0.850932	0.919463

Decision Tree Classifier is taken as the final base model as it is performing much better than Logistic Regression in term of the above mention metrics.

4.4 ENSEMBLE METHOD – RANDOM FOREST CLASSIFIER

An ensemble technique is built on top of Decision Tree Classifier as base model. Random forest is the extension of Decision Tree as it is collection of Decision Trees. It has the ability to limit overfitting of the model without substantially increasing error due to bias. Random Forests reduces variance by training on different samples of the data and by using a random subset of features. The model performance is further improved by doing a Grid search k-fold cross validation . The following hyper parameters are tuned in the random forest to increase the model performance.

- **n_estimators** - number of trees in the forest.
- **max_features** - maximum number of features considered for splitting a node
- **max_depth** - maximum number of levels in each decision tree
- **min_samples_split** - minimum number of data points placed in a node before the node is split
- **min_samples_leaf** - minimum number of data points allowed in a leaf node

The transformed data after over sampling is again split into train and test data. Model is trained using the train data and the metrics are calculated using the test data. The confusion matrix is generated after making predictions on the test data and then obtaining the values of True Positives, True Negatives, False Positives and False Negatives. The F1 score can be interpreted as a weighted average of the precision and recall. The following are the tuned hyperparameters.

- max_depth - 15
- max_features - 2
- min_samples_leaf - 1
- min_samples_split - 3
- n_estimators- - 30

The final random forest model gave the following metrics after predicting the trained model with test data.

S.No	Classifier	Test Score	Recall	Precision	F1 Score
1	Random Forest	0.953125	0.992701	0.888889	0.937931

5. CONCLUSION

A prediction model is built to predict the indicators of cervical cancer using Schiller as the target variable. Random forest algorithm gives the better model accuracy and f1 score of 0.95 and 0.93 respectively.

6. APPENDIX

6.1 APPENDIX-A PYTHON CODES

```
#import library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import zscore
from sklearn.model_selection import train_test_split
from sklearn import metrics
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import f1_score,precision_score,recall_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

import warnings
warnings.filterwarnings('ignore')

df  # data frame load in df

## replace ? with NaN
df = df.replace('?', np.NaN)

plt.figure(figsize=(10,10))
np.round(df.isnull().sum()/df.shape[0]*100).sort_values().plot(kind='barh')
df=df.drop(['STD_Time_since_first_diagnosis','STDs_Time_since_last_diagnosis'],axis=1)

### Removing the missing rows of Smokes and First Sexual Intercourse

df=df.drop(df.index[df.Smokes.isnull()] | df.index[df.First_sexual_intercourse.isnull()])

### Categorizing the features

x_features=list(df.columns)
x_features.remove('Schiller')

x_features_categorical=[

'Smokes','Hormonal_Contraceptives','IUD','STDs','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','Dx_Cancer','Dx_CIN','Dx_HPV','Dx']
x_features_categorical

x_features_numerical=[i for i in x_features if i not in x_features_categorical]
x_features_numerical

### Missing Value Imputation of IUD

df_iud=df.copy()

df_iud.info()

x_features_categorical.remove('IUD')
x_features_categorical

for i in x_features_categorical:
df_iud[i]=df_iud[i].fillna(df_iud[i].mode()[0])

for i in x_features_numerical:
df_iud[i]=df_iud[i].fillna(df_iud[i].median())

df_iud=df_iud.astype('float')
df_iud[x_features_categorical]=df_iud[x_features_categorical].replace(0,'no')
df_iud[x_features_categorical]=df_iud[x_features_categorical].replace(1,'yes')
df_iud.info()

df_iud=pd.get_dummies(df_iud)
df_iud.head()

train_iud=df_iud[df_iud.IUD.isnull()==False]
train_iud.shape

test_iud=df_iud[df_iud.IUD.isnull()]
test_iud.shape

train_iud_x=train_iud.drop('IUD',axis=1)
train_iud_y=train_iud['IUD']
```

```
test_iud_x=test_iud.drop('IUD',axis=1)
test_iud_y=test_iud['IUD']


dt=DecisionTreeClassifier()
iud_model=dt.fit(train_iud_x,train_iud_y)
iud_model.score(train_iud_x,train_iud_y)
test_iud['IUD']=iud_model.predict(test_iud_x)
df_impute=df.copy()
iud_complete=pd.concat([train_iud,test_iud],axis=0)
df_impute['IUD']=iud_complete['IUD'].sort_index()
df_impute.info()

### Missing Value Imputation for IUD_years

#### Here we can impute the missing values of IUD years considering the following reasons
- For all the zero values of IUD_years, the value of IUD is zero.
- For the zero values of IUD , the value of IUD_years is zero.
- The predicted value of IUD is completely zero and hence we can consider that the predicted value of IUD years will also be zero.

x_features_numerical.remove('IUD_years')
x_features_numerical

df_impute[df_impute['IUD_years']==0][['IUD','IUD_years']].sum()

df_impute[df_impute['IUD']==0][['IUD','IUD_years']].sum()

df_impute['IUD_years']=df_impute['IUD_years'].fillna(0)

df_impute.info()

### Missing Value Imputation for Hormonal_Contraceptives

df_hor=df_impute.drop(['Schiller'],axis=1)

df_impute[df_impute.Hormonal_Contraceptives.isnull()]['Schiller'].value_counts()

df_hor.shape

x_features_categorical.remove('Hormonal_Contraceptives')
x_features_categorical

for i in x_features_categorical:
df_hor[i]=df_hor[i].fillna(df_hor[i].mode()[0])

x_features_numerical

for i in x_features_numerical:
df_hor[i]=df_hor[i].fillna(df_hor[i].median())

df_hor=df_hor.astype('float')
df_hor[x_features_categorical]=df_hor[x_features_categorical].replace(0,'no')
df_hor[x_features_categorical]=df_hor[x_features_categorical].replace(1,'yes')
df_hor.info()

df_hor=pd.get_dummies(df_hor)
df_hor.head()

train_hor=df_hor[df_hor.Hormonal_Contraceptives.isnull()==False]
train_hor.shape

test_hor=df_hor[df_hor.Hormonal_Contraceptives.isnull()]
test_hor.shape

train_hor_x=train_hor.drop('Hormonal_Contraceptives',axis=1)
train_hor_y=train_hor['Hormonal_Contraceptives']
```

```
test_hor_x=test_hor.drop('Hormonal_Contraceptives',axis=1)
test_hor_y=test_hor['Hormonal_Contraceptives']

dt=DecisionTreeClassifier()
hor_model=dt.fit(train_hor_x,train_hor_y)

hor_model.score(train_hor_x,train_hor_y)

test_hor['Hormonal_Contraceptives']=hor_model.predict(test_hor_x)

hor_model.predict(test_hor_x)

hor_complete=pd.concat([train_hor,test_hor],axis=0)

df_impute['Hormonal_Contraceptives']=hor_complete['Hormonal_Contraceptives'].sort_index()

df_impute.info()

df_impute['Hormonal_Contraceptives'].value_counts()

### Missing Value Imputation for Hormonal_Contraceptives_Years

df_hor_y=df_impute.drop(['Schiller'],axis=1)

x_features_numerical.remove('Hormonal_Contraceptives_years')
x_features_numerical

for i in x_features_categorical:
df_hor_y[i]=df_hor_y[i].fillna(df_hor_y[i].mode()[0])

for i in x_features_numerical:
df_hor_y[i]=df_hor_y[i].fillna(df_hor_y[i].median())

df_hor_y.info()

df_hor_y=df_hor_y.astype('float')
df_hor_y[x_features_categorical]=df_hor_y[x_features_categorical].replace(0,'no')
df_hor_y[x_features_categorical]=df_hor_y[x_features_categorical].replace(1,'yes')
df_hor_y=pd.get_dummies(df_hor_y)
df_hor_y.head()

train_hor_yrs=df_hor_y[df_hor_y.Hormonal_Contraceptives_years.isnull()==False]
test_hor_yrs=df_hor_y[df_hor_y.Hormonal_Contraceptives_years.isnull()]
train_hor_yrs_x=train_hor_yrs.drop('Hormonal_Contraceptives_years',axis=1)
train_hor_yrs_y=train_hor_yrs['Hormonal_Contraceptives_years']
test_hor_yrs_x=test_hor_yrs.drop('Hormonal_Contraceptives_years',axis=1)
test_hor_yrs_y=test_hor_yrs['Hormonal_Contraceptives_years']
dt=DecisionTreeRegressor()
hor_yrs_model=dt.fit(train_hor_yrs_x,train_hor_yrs_y)

hor_yrs_model.score(train_hor_yrs_x,train_hor_yrs_y)

test_hor_yrs['Hormonal_Contraceptives_years']=hor_yrs_model.predict(test_hor_yrs_x)
hor_yrs_model.predict(test_hor_yrs_x)
hor_yrs_complete=pd.concat([train_hor_yrs,test_hor_yrs],axis=0)
df_hor_y['Hormonal_Contraceptives_years']=hor_yrs_complete['Hormonal_Contraceptives_years'].sort_index()
df_impute['Hormonal_Contraceptives_years']=df_hor_y['Hormonal_Contraceptives_years']

df_impute.info()

### Missing Value Imputation for STDs

df_std=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs')
x_features_categorical
```

```
for i in x_features_categorical:
df_std[i]=df_std[i].fillna(df_std[i].mode()[0])

x_features_numerical

for i in x_features_numerical:
df_std[i]=df_std[i].fillna(df_std[i].median())

df_std=df_std.astype('float')
df_std[x_features_categorical]=df_std[x_features_categorical].replace(0,'no')
df_std[x_features_categorical]=df_std[x_features_categorical].replace(1,'yes')
df_std=pd.get_dummies(df_std)
df_std.head()

train_std=df_std[df_std.STDs.isnull()==False]
test_std=df_std[df_std.STDs.isnull()]
train_std_x=train_std.drop('STDs',axis=1)
train_std_y=train_std['STDs']
test_std_x=test_std.drop('STDs',axis=1)
test_std_y=test_std['STDs']
dt=DecisionTreeClassifier()
std_model=dt.fit(train_std_x,train_std_y)

std_model.score(train_std_x,train_std_y)

test_std['STDs']=std_model.predict(test_std_x)
std_complete=pd.concat([train_std,test_std],axis=0)
df_std['STDs']=std_complete['STDs'].sort_index()
df_impute['STDs']=df_std['STDs']

df_impute.info()

### Missing Value Imputation for STDs_Number

df_std_num=df_impute.drop(['Schiller'],axis=1)

x_features_numerical.remove('STDs_number')
x_features_numerical

for i in x_features_categorical:
df_std_num[i]=df_std_num[i].fillna(df_std_num[i].mode()[0])

for i in x_features_numerical:
df_std_num[i]=df_std_num[i].fillna(df_std_num[i].median())

df_std_num=df_std_num.astype('float')
df_std_num[x_features_categorical]=df_std_num[x_features_categorical].replace(0,'no')
df_std_num[x_features_categorical]=df_std_num[x_features_categorical].replace(1,'yes')
df_std_num=pd.get_dummies(df_std_num)
df_std_num.head()

train_std_num=df_std_num[df_std_num.STDs_number.isnull()==False]
test_std_num=df_std_num[df_std_num.STDs_number.isnull()]
train_std_num_x=train_std_num.drop('STDs_number',axis=1)
train_std_num_y=train_std_num['STDs_number']
test_std_num_x=test_std_num.drop('STDs_number',axis=1)
test_std_num_y=test_std_num['STDs_number']
dt=DecisionTreeRegressor()
std_model_num=dt.fit(train_std_num_x,train_std_num_y)

std_model_num.score(train_std_num_x,train_std_num_y)

test_std_num['STDs_number']=std_model_num.predict(test_std_num_x)
std_model_num.predict(test_std_num_x)
std_num_complete=pd.concat([train_std_num,test_std_num],axis=0)
df_std_num['STDs_number']=std_num_complete['STDs_number'].sort_index()
```

```
df_impute['STDs_number']=df_std_num['STDs_number']

df_impute.info()

### Missing Value Imputation for STDs_condylomatosis

df_std_con=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_condylomatosis')
x_features_categorical

for i in x_features_categorical:
df_std_con[i]=df_std_con[i].fillna(df_std_con[i].mode()[0])

for i in x_features_numerical:
df_std_con[i]=df_std_con[i].fillna(df_std_con[i].median())

df_std_con=df_std_con.astype('float')
df_std_con[x_features_categorical]=df_std_con[x_features_categorical].replace(0,'no')
df_std_con[x_features_categorical]=df_std_con[x_features_categorical].replace(1,'yes')
df_std_con=pd.get_dummies(df_std_con)
df_std_con.head()

df_std_con[df_std_con['STDs_condylomatosis'].isnull()]['STDs_condylomatosis']

df_std_con[df_std_con.STDs_condylomatosis.isnull()]['STDs_condylomatosis']

train_std_con=df_std_con[df_std_con.STDs_condylomatosis.isnull()==False]
test_std_con=df_std_con[df_std_con.STDs_condylomatosis.isnull()]
train_std_con_x=train_std_con.drop('STDs_condylomatosis',axis=1)
train_std_con_y=train_std_con['STDs_condylomatosis']
test_std_con_x=test_std_con.drop('STDs_condylomatosis',axis=1)
test_std_con_y=test_std_con['STDs_condylomatosis']
dt=DecisionTreeClassifier()
std_model_con=dt.fit(train_std_con_x,train_std_con_y)

std_model_con.score(train_std_con_x,train_std_con_y)

test_std_con['STDs_condylomatosis']=std_model_con.predict(test_std_con_x)
std_con_complete=pd.concat([train_std_con,test_std_con],axis=0)
df_std_con['STDs_condylomatosis']=std_con_complete['STDs_condylomatosis'].sort_index()
df_impute['STDs_condylomatosis']=df_std_con['STDs_condylomatosis']

df_impute.info()

### Missing Value Imputation for STDs_cervical_condylomatosis

df_std_cerv=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_cervical_condylomatosis')
x_features_categorical

for i in x_features_categorical:
df_std_cerv[i]=df_std_cerv[i].fillna(df_std_cerv[i].mode()[0])

for i in x_features_numerical:
df_std_cerv[i]=df_std_cerv[i].fillna(df_std_cerv[i].median())

df_std_cerv=df_std_cerv.astype('float')
df_std_cerv[x_features_categorical]=df_std_cerv[x_features_categorical].replace(0,'no')
df_std_cerv[x_features_categorical]=df_std_cerv[x_features_categorical].replace(1,'yes')
df_std_cerv=pd.get_dummies(df_std_cerv)
df_std_cerv.head()

train_std_cerv=df_std_cerv[df_std_cerv.STDs_cervical_condylomatosis.isnull()==False]
test_std_cerv=df_std_cerv[df_std_cerv.STDs_cervical_condylomatosis.isnull()]
train_std_cerv_x=train_std_cerv.drop('STDs_cervical_condylomatosis',axis=1)
```

```
train_std_cerv_y=train_std_cerv['STDs_cervical_condylomatosis']
test_std_cerv_x=test_std_cerv.drop('STDs_cervical_condylomatosis',axis=1)
test_std_cerv_y=test_std_cerv['STDs_cervical_condylomatosis']
dt=DecisionTreeClassifier()
std_model_cerv=dt.fit(train_std_cerv_x,train_std_cerv_y)

std_model_cerv.score(train_std_cerv_x,train_std_cerv_y)

test_std_cerv['STDs_cervical_condylomatosis']=std_model_cerv.predict(test_std_cerv_x)
std_model_cerv.predict(test_std_cerv_x)
std_cerv_complete=pd.concat([train_std_cerv,test_std_cerv],axis=0)
df_std_cerv['STDs_cervical_condylomatosis']=std_cerv_complete['STDs_cervical_condylomatosis'].sort_index()
df_impute['STDs_cervical_condylomatosis']=df_std_cerv['STDs_cervical_condylomatosis']

df_impute.info()

### Missing Value Imputation for STDs_vaginal_condylomatosis

df_std_vagi=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_vaginal_condylomatosis')
x_features_categorical

for i in x_features_categorical:
df_std_vagi[i]=df_std_vagi[i].fillna(df_std_vagi[i].mode()[0])

for i in x_features_numerical:
df_std_vagi[i]=df_std_vagi[i].fillna(df_std_vagi[i].median())

df_std_vagi=df_std_vagi.astype('float')
df_std_vagi[x_features_categorical]=df_std_vagi[x_features_categorical].replace(0,'no')
df_std_vagi[x_features_categorical]=df_std_vagi[x_features_categorical].replace(1,'yes')
df_std_vagi=pd.get_dummies(df_std_vagi)
df_std_vagi.head()

train_std_vagi=df_std_vagi[df_std_vagi.STDs_vaginal_condylomatosis.isnull()==False]
test_std_vagi=df_std_vagi[df_std_vagi.STDs_vaginal_condylomatosis.isnull()]
train_std_vagi_x=train_std_vagi.drop('STDs_vaginal_condylomatosis',axis=1)
train_std_vagi_y=train_std_vagi['STDs_vaginal_condylomatosis']
test_std_vagi_x=test_std_vagi.drop('STDs_vaginal_condylomatosis',axis=1)
test_std_vagi_y=test_std_vagi['STDs_vaginal_condylomatosis']
dt=DecisionTreeClassifier()
std_model_vagi=dt.fit(train_std_vagi_x,train_std_vagi_y)

std_model_vagi.score(train_std_vagi_x,train_std_vagi_y)

test_std_vagi['STDs_vaginal_condylomatosis']=std_model_vagi.predict(test_std_vagi_x)
std_model_vagi.predict(test_std_vagi_x)
std_vagi_complete=pd.concat([train_std_vagi,test_std_vagi],axis=0)
df_std_vagi['STDs_vaginal_condylomatosis']=std_vagi_complete['STDs_vaginal_condylomatosis'].sort_index()
df_impute['STDs_vaginal_condylomatosis']=df_std_vagi['STDs_vaginal_condylomatosis']

df_impute.info()

### Missing Value Imputation for STDs_vulvo_perineal_condylomatosis

df_std_peri=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_vulvo_perineal_condylomatosis')
x_features_categorical

for i in x_features_categorical:
df_std_peri[i]=df_std_peri[i].fillna(df_std_peri[i].mode()[0])

for i in x_features_numerical:
df_std_peri[i]=df_std_peri[i].fillna(df_std_peri[i].median())
```

```
df_std_peri=df_std_peri.astype('float')
df_std_peri[x_features_categorical]=df_std_peri[x_features_categorical].replace(0,'no')
df_std_peri[x_features_categorical]=df_std_peri[x_features_categorical].replace(1,'yes')
df_std_peri=pd.get_dummies(df_std_peri)
df_std_peri.head()

train_std_peri=df_std_peri[df_std_peri.STDs_vulvo_perineal_condylomatosis.isnull()==False]
test_std_peri=df_std_peri[df_std_peri.STDs_vulvo_perineal_condylomatosis.isnull()]
train_std_peri_x=train_std_peri.drop('STDs_vulvo_perineal_condylomatosis',axis=1)
train_std_peri_y=train_std_peri['STDs_vulvo_perineal_condylomatosis']
test_std_peri_x=test_std_peri.drop('STDs_vulvo_perineal_condylomatosis',axis=1)
test_std_peri_y=test_std_peri['STDs_vulvo_perineal_condylomatosis']
dt=DecisionTreeClassifier()
std_model_peri=dt.fit(train_std_peri_x,train_std_peri_y)

std_model_peri.score(train_std_peri_x,train_std_peri_y)

test_std_peri['STDs_vulvo_perineal_condylomatosis']=std_model_peri.predict(test_std_peri_x)
std_model_peri.predict(test_std_peri_x)
std_peri_complete=pd.concat([train_std_peri,test_std_peri],axis=0)
df_std_peri['STDs_vulvo_perineal_condylomatosis']=std_peri_complete['STDs_vulvo_perineal_condylomatosis'].sort_index
()
df_impute['STDs_vulvo_perineal_condylomatosis']=df_std_peri['STDs_vulvo_perineal_condylomatosis']

df_impute.info()

### Missing Value Imputation for STDs_syphilis

df_std_syp=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_syphilis')

for i in x_features_categorical:
df_std_syp[i]=df_std_syp[i].fillna(df_std_syp[i].mode()[0])

for i in x_features_numerical:
df_std_syp[i]=df_std_syp[i].fillna(df_std_syp[i].median())

df_std_syp=df_std_syp.astype('float')
df_std_syp[x_features_categorical]=df_std_syp[x_features_categorical].replace(0,'no')
df_std_syp[x_features_categorical]=df_std_syp[x_features_categorical].replace(1,'yes')
df_std_syp=pd.get_dummies(df_std_syp)
df_std_syp.head()

train_std_syp=df_std_syp[df_std_syp.STDs_syphilis.isnull()==False]
test_std_syp=df_std_syp[df_std_syp.STDs_syphilis.isnull()]
train_std_syp_x=train_std_syp.drop('STDs_syphilis',axis=1)
train_std_syp_y=train_std_syp['STDs_syphilis']
test_std_syp_x=test_std_syp.drop('STDs_syphilis',axis=1)
test_std_syp_y=test_std_syp['STDs_syphilis']
dt=DecisionTreeClassifier()
std_model_syp=dt.fit(train_std_syp_x,train_std_syp_y)

std_model_syp.score(train_std_syp_x,train_std_syp_y)

test_std_syp['STDs_syphilis']=std_model_syp.predict(test_std_syp_x)
std_model_syp.predict(test_std_syp_x)
std_syp_complete=pd.concat([train_std_syp,test_std_syp],axis=0)
df_std_syp['STDs_syphilis']=std_syp_complete['STDs_syphilis'].sort_index()
df_impute['STDs_syphilis']=df_std_syp['STDs_syphilis']

df_impute.info()

### Missing Value Imputation for STDs_pelvic_inflammatory_disease

df_std_pelv=df_impute.drop(['Schiller'],axis=1)
```

```
x_features_categorical.remove('STDs_pelvic_inflammatory_disease')

for i in x_features_categorical:
df_std_pelv[i]=df_std_pelv[i].fillna(df_std_pelv[i].mode()[0])

for i in x_features_numerical:
df_std_pelv[i]=df_std_pelv[i].fillna(df_std_pelv[i].median())

df_std_pelv=df_std_pelv.astype('float')
df_std_pelv[x_features_categorical]=df_std_pelv[x_features_categorical].replace(0,'no')
df_std_pelv[x_features_categorical]=df_std_pelv[x_features_categorical].replace(1,'yes')
df_std_pelv=pd.get_dummies(df_std_pelv)
df_std_pelv.head()

train_std_pelv=df_std_pelv[df_std_pelv.STDs_pelvic_inflammatory_disease.isnull()==False]
test_std_pelv=df_std_pelv[df_std_pelv.STDs_pelvic_inflammatory_disease.isnull()]
train_std_pelv_x=train_std_pelv.drop('STDs_pelvic_inflammatory_disease',axis=1)
train_std_pelv_y=train_std_pelv['STDs_pelvic_inflammatory_disease']
test_std_pelv_x=test_std_pelv.drop('STDs_pelvic_inflammatory_disease',axis=1)
test_std_pelv_y=test_std_pelv['STDs_pelvic_inflammatory_disease']
dt=DecisionTreeClassifier()
std_model_pelv=dt.fit(train_std_pelv_x,train_std_pelv_y)

std_model_pelv.score(train_std_pelv_x,train_std_pelv_y)

test_std_pelv['STDs_pelvic_inflammatory_disease']=std_model_pelv.predict(test_std_pelv_x)
std_model_pelv.predict(test_std_pelv_x)
std_pelv_complete=pd.concat([train_std_pelv,test_std_pelv],axis=0)
df_std_pelv['STDs_pelvic_inflammatory_disease']=std_pelv_complete['STDs_pelvic_inflammatory_disease'].sort_index()
df_impute['STDs_pelvic_inflammatory_disease']=df_std_pelv['STDs_pelvic_inflammatory_disease']

df_impute.info()

### Missing Value Imputation for STDs_genital_herpes

x_features_categorical.remove('STDs_genital_herpes')
x_features_categorical

df_std_geni=df_impute.drop(['Schiller'],axis=1)

for i in x_features_categorical:
df_std_geni[i]=df_std_geni[i].fillna(df_std_geni[i].mode()[0])

for i in x_features_numerical:
df_std_geni[i]=df_std_geni[i].fillna(df_std_geni[i].median())

df_std_geni=df_std_geni.astype('float')
df_std_geni[x_features_categorical]=df_std_geni[x_features_categorical].replace(0,'no')
df_std_geni[x_features_categorical]=df_std_geni[x_features_categorical].replace(1,'yes')
df_std_geni=pd.get_dummies(df_std_geni)
df_std_geni.head()

train_std_geni=df_std_geni[df_std_geni.STDs_genital_herpes.isnull()==False]
test_std_geni=df_std_geni[df_std_geni.STDs_genital_herpes.isnull()]
train_std_geni_x=train_std_geni.drop('STDs_genital_herpes',axis=1)
train_std_geni_y=train_std_geni['STDs_genital_herpes']
test_std_geni_x=test_std_geni.drop('STDs_genital_herpes',axis=1)
test_std_geni_y=test_std_geni['STDs_genital_herpes']
dt=DecisionTreeClassifier()
std_model_geni=dt.fit(train_std_geni_x,train_std_geni_y)

std_model_geni.score(train_std_geni_x,train_std_geni_y)

test_std_geni['STDs_genital_herpes']=std_model_geni.predict(test_std_geni_x)
std_model_geni.predict(test_std_geni_x)
std_geni_complete=pd.concat([train_std_geni,test_std_geni],axis=0)
df_std_geni['STDs_genital_herpes']=std_geni_complete['STDs_genital_herpes'].sort_index()
```

```
df_impute['STDs_genital_herpes']=df_std_geni['STDs_genital_herpes']

df_impute.info()

### Missing Value Imputation for STDs_molluscum_contagiosum

df_std_mollu=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_molluscum_contagiosum')
x_features_categorical

for i in x_features_categorical:
df_std_mollu[i]=df_std_mollu[i].fillna(df_std_mollu[i].mode()[0])

for i in x_features_numerical:
df_std_mollu[i]=df_std_mollu[i].fillna(df_std_mollu[i].median())

df_std_mollu=df_std_mollu.astype('float')
df_std_mollu[x_features_categorical]=df_std_mollu[x_features_categorical].replace(0,'no')
df_std_mollu[x_features_categorical]=df_std_mollu[x_features_categorical].replace(1,'yes')
df_std_mollu=pd.get_dummies(df_std_mollu)
df_std_mollu.head()

train_std_mollu=df_std_mollu[df_std_mollu.STDs_molluscum_contagiosum.isnull()==False]
test_std_mollu=df_std_mollu[df_std_mollu.STDs_molluscum_contagiosum.isnull()]
train_std_mollu_x=train_std_mollu.drop('STDs_molluscum_contagiosum',axis=1)
train_std_mollu_y=train_std_mollu['STDs_molluscum_contagiosum']
test_std_mollu_x=test_std_mollu.drop('STDs_molluscum_contagiosum',axis=1)
test_std_mollu_y=test_std_mollu['STDs_molluscum_contagiosum']
dt=DecisionTreeClassifier()
std_model_mollu=dt.fit(train_std_mollu_x,train_std_mollu_y)

std_model_mollu.score(train_std_mollu_x,train_std_mollu_y)

test_std_mollu['STDs_molluscum_contagiosum']=std_model_mollu.predict(test_std_mollu_x)
std_model_mollu.predict(test_std_mollu_x)
std_mollu_complete=pd.concat([train_std_mollu,test_std_mollu],axis=0)
df_std_mollu['STDs_molluscum_contagiosum']=std_mollu_complete['STDs_molluscum_contagiosum'].sort_index()
df_impute['STDs_molluscum_contagiosum']=df_std_mollu['STDs_molluscum_contagiosum']

df_impute.info()

### Missing Value Imputation for STDs_AIDS

df_std_aids=df_impute.drop(['Schiller'],axis=1)

x_features_categorical.remove('STDs_AIDS')
x_features_categorical

for i in x_features_categorical:
df_std_aids[i]=df_std_aids[i].fillna(df_std_aids[i].mode()[0])

for i in x_features_numerical:
df_std_aids[i]=df_std_aids[i].fillna(df_std_aids[i].median())

df_std_aids=df_std_aids.astype('float')
df_std_aids[x_features_categorical]=df_std_aids[x_features_categorical].replace(0,'no')
df_std_aids[x_features_categorical]=df_std_aids[x_features_categorical].replace(1,'yes')
df_std_aids=pd.get_dummies(df_std_aids)
df_std_aids.head()

train_std_aids=df_std_aids[df_std_aids.STDs_AIDS.isnull()==False]
test_std_aids=df_std_aids[df_std_aids.STDs_AIDS.isnull()]
train_std_aids_x=train_std_aids.drop('STDs_AIDS',axis=1)
train_std_aids_y=train_std_aids['STDs_AIDS']
test_std_aids_x=test_std_aids.drop('STDs_AIDS',axis=1)
test_std_aids_y=test_std_aids['STDs_AIDS']
```

```
dt=DecisionTreeClassifier()
std_model_aids=dt.fit(train_std_aids_x,train_std_aids_y)

std_model_aids.score(train_std_aids_x,train_std_aids_y)

test_std_aids['STDs_AIDS']=std_model_aids.predict(test_std_aids_x)
std_model_aids.predict(test_std_aids_x)
std_aids_complete=pd.concat([train_std_aids,test_std_aids],axis=0)
df_std_aids['STDs_AIDS']=std_aids_complete['STDs_AIDS'].sort_index()
df_impute['STDs_AIDS']=df_std_aids['STDs_AIDS']

### Missing Value Imputation for STDs_HIV

df_std_hiv=df_impute.drop(['Schiller'],axis=1)
x_features_categorical.remove('STDs_HIV')

for i in x_features_categorical:
df_std_hiv[i]=df_std_hiv[i].fillna(df_std_hiv[i].mode()[0])

for i in x_features_numerical:
df_std_hiv[i]=df_std_hiv[i].fillna(df_std_hiv[i].median())

df_std_hiv=df_std_hiv.astype('float')
df_std_hiv[x_features_categorical]=df_std_hiv[x_features_categorical].replace(0,'no')
df_std_hiv[x_features_categorical]=df_std_hiv[x_features_categorical].replace(1,'yes')
df_std_hiv=pd.get_dummies(df_std_hiv)
df_std_hiv.head()

train_std_hiv=df_std_hiv[df_std_hiv.STDs_HIV.isnull()==False]
test_std_hiv=df_std_hiv[df_std_hiv.STDs_HIV.isnull()]
train_std_hiv_x=train_std_hiv.drop('STDs_HIV',axis=1)
train_std_hiv_y=train_std_hiv['STDs_HIV']
test_std_hiv_x=test_std_hiv.drop('STDs_HIV',axis=1)
test_std_hiv_y=test_std_hiv['STDs_HIV']
dt=DecisionTreeClassifier()
std_model_hiv=dt.fit(train_std_hiv_x,train_std_hiv_y)

std_model_hiv.score(train_std_hiv_x,train_std_hiv_y)

test_std_hiv['STDs_HIV']=std_model_hiv.predict(test_std_hiv_x)
std_model_hiv.predict(test_std_hiv_x)
std_hiv_complete=pd.concat([train_std_hiv,test_std_hiv],axis=0)
df_std_hiv['STDs_HIV']=std_hiv_complete['STDs_HIV'].sort_index()
df_impute['STDs_HIV']=df_std_hiv['STDs_HIV']


### Missing Value Imputation for STDs_Hepatitis_B

df_std_hepa=df_impute.drop(['Schiller'],axis=1)
x_features_categorical.remove('STDs_Hepatitis_B')

for i in x_features_categorical:
df_std_hepa[i]=df_std_hepa[i].fillna(df_std_hepa[i].mode()[0])

for i in x_features_numerical:
df_std_hepa[i]=df_std_hepa[i].fillna(df_std_hepa[i].median())

df_std_hepa=df_std_hepa.astype('float')
df_std_hepa[x_features_categorical]=df_std_hepa[x_features_categorical].replace(0,'no')
df_std_hepa[x_features_categorical]=df_std_hepa[x_features_categorical].replace(1,'yes')
df_std_hepa=pd.get_dummies(df_std_hepa)
df_std_hepa.head()

train_std_hepa=df_std_hepa[df_std_hepa.STDs_Hepatitis_B.isnull()==False]
test_std_hepa=df_std_hepa[df_std_hepa.STDs_Hepatitis_B.isnull()]
train_std_hepa_x=train_std_hepa.drop(['STDs_Hepatitis_B'],axis=1)
train_std_hepa_y=train_std_hepa['STDs_Hepatitis_B']
```

```
test_std_hepa_x=test_std_hepa.drop(['STDs_Hepatitis_B'],axis=1)
test_std_hepa_y=test_std_hepa['STDs_Hepatitis_B']
dt=DecisionTreeClassifier()
std_model_hepa=dt.fit(train_std_hepa_x,train_std_hepa_y)
std_model_hepa.score(train_std_hepa_x,train_std_hepa_y)

test_std_hepa['STDs_Hepatitis_B']=std_model_hepa.predict(test_std_hepa_x)
std_model_hepa.predict(test_std_hepa_x)
std_hepa_complete=pd.concat([train_std_hepa,test_std_hepa],axis=0)
df_std_hepa['STDs_Hepatitis_B']=std_hepa_complete['STDs_Hepatitis_B'].sort_index()
df_impute['STDs_Hepatitis_B']=df_std_hepa['STDs_Hepatitis_B']


### Missing Value Imputation for STDs_HPV

df_std_hpv=df_impute.drop(['Schiller'],axis=1)
x_features_categorical.remove('STDs_HPV')

for i in x_features_categorical:
df_std_hpv[i]=df_std_hpv[i].fillna(df_std_hpv[i].mode()[0])

for i in x_features_numerical:
df_std_hpv[i]=df_std_hpv[i].fillna(df_std_hpv[i].median())

df_std_hpv=df_std_hpv.astype('float')
df_std_hpv[x_features_categorical]=df_std_hpv[x_features_categorical].replace(0,'no')
df_std_hpv[x_features_categorical]=df_std_hpv[x_features_categorical].replace(1,'yes')
df_std_hpv=pd.get_dummies(df_std_hpv)
df_std_hpv.head()

train_std_hpv=df_std_hpv[df_std_hpv.STDs_HPV.isnull()==False]
test_std_hpv=df_std_hpv[df_std_hpv.STDs_HPV.isnull()]
train_std_hpv_x=train_std_hpv.drop(['STDs_HPV'],axis=1)
train_std_hpv_y=train_std_hpv['STDs_HPV']
test_std_hpv_x=test_std_hpv.drop(['STDs_HPV'],axis=1)
test_std_hpv_y=test_std_hpv['STDs_HPV']
dt=DecisionTreeClassifier()
std_model_hpv=dt.fit(train_std_hpv_x,train_std_hpv_y)

std_model_hpv.score(train_std_hpv_x,train_std_hpv_y)

test_std_hpv['STDs_HPV']=std_model_hpv.predict(test_std_hpv_x)
std_model_hpv.predict(test_std_hpv_x)
std_hpv_complete=pd.concat([train_std_hpv,test_std_hpv],axis=0)
df_std_hpv['STDs_HPV']=std_hpv_complete['STDs_HPV'].sort_index()
df_impute['STDs_HPV']=df_std_hpv['STDs_HPV']

df_impute.info()

### Missing Value Imputation for No_pregnancies

df_no_preg=df_impute.drop(['Schiller'],axis=1)

x_features_numerical.remove('No_pregnancies')

for i in x_features_numerical:
df_no_preg[i]=df_no_preg[i].fillna(df_no_preg[i].median())

for i in x_features_categorical:
df_no_preg[i]=df_no_preg[i].fillna(df_no_preg[i].mode()[0])

df_no_preg=df_no_preg.astype('float')
df_no_preg[x_features_categorical]=df_no_preg[x_features_categorical].replace(0,'no')
df_no_preg[x_features_categorical]=df_no_preg[x_features_categorical].replace(1,'yes')
df_no_preg=pd.get_dummies(df_no_preg)
df_no_preg.head()
```

```
train_no_preg=df_no_preg[df_no_preg.No_pregnancies.isnull()==False]
test_no_preg=df_no_preg[df_no_preg.No_pregnancies.isnull()]
train_no_preg_x=train_no_preg.drop(['No_pregnancies'],axis=1)
train_no_preg_y=train_no_preg['No_pregnancies']
test_no_preg_x=test_no_preg.drop(['No_pregnancies'],axis=1)
test_no_preg_y=test_no_preg['No_pregnancies']
dt=DecisionTreeRegressor()
model_no_preg=dt.fit(train_no_preg_x,train_no_preg_y)

model_no_preg.score(train_no_preg_x,train_no_preg_y)

test_no_preg['No_pregnancies']=model_no_preg.predict(test_no_preg_x)
no_preg_complete=pd.concat([train_no_preg,test_no_preg],axis=0)
df_no_preg['No_pregnancies']=no_preg_complete['No_pregnancies'].sort_index()
df_impute['No_pregnancies']=df_no_preg['No_pregnancies']

df_impute.info()

### Missing Value Imputation for No_of_sex_partner

df_no_sexptnr=df_impute.drop(['Schiller'],axis=1)

x_features_numerical.remove('No_of_sex_partner')

for i in x_features_numerical:
df_no_sexptnr[i]=df_no_sexptnr[i].fillna(df_no_sexptnr[i].median())

for i in x_features_categorical:
df_no_sexptnr[i]=df_no_sexptnr[i].fillna(df_no_sexptnr[i].mode()[0])

df_no_sexptnr=df_no_sexptnr.astype('float')
df_no_sexptnr[x_features_categorical]=df_no_sexptnr[x_features_categorical].replace(0,'no')
df_no_sexptnr[x_features_categorical]=df_no_sexptnr[x_features_categorical].replace(1,'yes')
df_no_sexptnr=pd.get_dummies(df_no_sexptnr)
df_no_sexptnr.head()

train_no_sexptnr=df_no_sexptnr[df_no_sexptnr.No_of_sex_partner.isnull()==False]
test_no_sexptnr=df_no_sexptnr[df_no_sexptnr.No_of_sex_partner.isnull()]
train_no_sexptnr_x=train_no_sexptnr.drop(['No_of_sex_partner'],axis=1)
train_no_sexptnr_y=train_no_sexptnr['No_of_sex_partner']
test_no_sexptnr_x=test_no_sexptnr.drop(['No_of_sex_partner'],axis=1)
test_no_sexptnr_y=test_no_sexptnr['No_of_sex_partner']
dt=DecisionTreeRegressor()
model_no_sexptnr=dt.fit(train_no_sexptnr_x,train_no_sexptnr_y)

print('The score is',model_no_sexptnr.score(train_no_sexptnr_x,train_no_sexptnr_y))

test_no_sexptnr['No_of_sex_partner']=model_no_sexptnr.predict(test_no_sexptnr_x)
no_sexptnr_complete=pd.concat([train_no_sexptnr,test_no_sexptnr],axis=0)
df_no_sexptnr['No_of_sex_partner']=no_sexptnr_complete['No_of_sex_partner'].sort_index()
df_impute['No_of_sex_partner']=df_no_sexptnr['No_of_sex_partner']


# Univariate Analysis

impute=df_impute.copy()
impute=df_impute.astype('float')
impute[x_features_categorical]=impute[x_features_categorical].replace(0,'no')
impute[x_features_categorical]=impute[x_features_categorical].replace(1,'yes')


# Age

a=sns.distplot(impute.Age,kde=False,color='blue')
a.set_title('Histogram of Age',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Age',fontsize=15)
```

```
plt.show()

# Number of Sex Partner

a=sns.countplot(impute.No_of_sex_partner)
a.set_title('Number of Sex Partners',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Number of Sex Partner',fontsize=15)
plt.show()

# First Sexual Intercourse

a=sns.countplot(impute.First_sexual_intercourse.astype('int64'))
a.set_title('Age (Years) - First Sexual Intercourse',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('First Sexual Intercourse Age',fontsize=15)
plt.show()

# Number of Pregnancies

a=sns.countplot(impute.No_pregnancies.astype('int64'))
a.set_title('Total Number of Pregnancies',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Number of Pregnancies',fontsize=15)
plt.show()

# Smokes

a=sns.countplot(impute.Smokes)
a.set_title('Indication of Smoke',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Smokes',fontsize=15)
plt.show()

a=sns.countplot(impute.Hormonal_Contraceptives)
a.set_title('Usage of Hormonal Contraceptives',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Hormonal Contraceptives',fontsize=15)
plt.show()

a=sns.countplot(impute.Hormonal_Contraceptives_years.astype('int64'))
a.set_title('Number of Hormonal Contraceptive Years',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Number of Hormonal Contraceptive Years',fontsize=15)
plt.show()

a=sns.countplot(impute.IUD)
a.set_title('Usage of IUDs',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('IUDs',fontsize=15)
plt.show()

a=sns.countplot(impute.STDs)
a.set_title('Indication of STDs',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('STDs',fontsize=15)
plt.show()

a=sns.countplot(impute.STDs_number)
a.set_title('Total number of STDs',fontsize=15)
a.set_ylabel('Frequency',fontsize=15)
a.set_xlabel('Number of STDs',fontsize=15)
plt.show()

# Multi Variate Analysis

## Types of Attributes Categories
```

```
impute=df_impute.copy()
impute=df_impute.astype('float')

impute[x_features_categorical]=impute[x_features_categorical].replace(0,'no')
impute[x_features_categorical]=impute[x_features_categorical].replace(1,'yes')

impute.Smokes_yrs=impute.Smokes_yrs.astype('float64')
impute.IUD_years=impute.IUD_years.astype('float64')
impute.Smokes_packs_yr=impute.Smokes_packs_yr.astype('float64')
impute['smokes_yr_cat']=pd.cut(impute.Smokes_yrs,[-1,2,5,10,15,20,25,50],labels=['<2','2-5','5-10','10-15','15-20','20-
25','>25'])
impute=impute.drop(['Smokes_yrs'],axis=1)
impute['smokepack_yr_cat']=pd.cut(impute.Smokes_packs_yr,[-1,2,5,10,20,50],labels=['<2','2-5','5-10','10-20','>20'])
impute=impute.drop(['Smokes_packs_yr'],axis=1)
impute['IUD_yr_cat']=pd.cut(impute.IUD_years,[-1,2,4,6,8,10,15,20],labels=['<2','2-4','4-6','6-8','8-10','10-15','>15'])
impute=impute.drop(['IUD_years'],axis=1)

impute.head()

impute.info()

## 1. Sexual Habits Attributes
- No_of_sex_partner - Total number of sexual partners the patient had.
- First_sexual_intercourse - The age when the patient had their first sexual intercourse
- No_pregnancies - Total number of pregnancies the patient had.

plt.subplots(2,2,figsize=(20,16))
plt.subplot(2,2,1)
a=sns.boxplot(y=impute['First_sexual_intercourse'],hue=impute['Schiller'],x=impute['Smokes'])
a.axes.set_title("Box Plot b/w Schiller & First Sexual Intercourse",fontsize=22)
a.set_xlabel("Smokes",fontsize=20)
a.set_ylabel("First Sexual Intercourse",fontsize=20)
plt.subplot(2,2,2)
plt.title('Box Plot b/w Schiller & Number of Sexual Partners',fontsize=20)
a=sns.boxplot(y=impute['No_of_sex_partner'],hue=impute['Schiller'],x=impute['Smokes'])
a.set_xlabel("Smokes",fontsize=20)
a.set_ylabel("Number of Sexual Partners",fontsize=20)
plt.subplot(2,2,3)
a=sns.boxplot(y=impute['No_pregnancies'],hue=impute['Schiller'],x=impute['Smokes'])
a.axes.set_title("Box Plot b/w Schiller & Total number of Pregnancies",fontsize=22)
a.set_xlabel('Smokes',fontsize=20)
a.set_ylabel("Total number of Pregnancies",fontsize=20)
plt.show()

## Inferences
- Those who smoke and age at the time first sexual intercourse is between 15yrs & 18 yrs are more prone to be test as
Positive in Schiller test.
- The persons who have sexual partners between 1 & 3 are more prone to be testes as positive in Schiller test
- The person who smokes and with higher number of pregnancies are more prone to be tested as positive in Schiller test

## 2. Smoking habits attributes
- Smokes - It indicates whether the person smokes or not
- smokes_yr_cat - It indicates for how many years the person has been smoking
- smokepack_yr_cat - It indicates how many packets per year the person is smoking

plt.subplots(2,2,figsize=(20,16))
plt.subplot(2,2,1)
b=sns.boxplot(hue=impute['Schiller'],x=impute['Smokes'],y=impute['Age'])
b.axes.set_title("Box Plot b/w Schiller & Smokes",fontsize=22)
b.set_xlabel("Smokes",fontsize=20)
b.set_ylabel("Age of Patient",fontsize=20)
plt.subplot(2,2,2)
b=sns.boxplot(hue=impute['Schiller'],x=impute['smokes_yr_cat'],y=impute['Age'])
b.axes.set_title("Box Plot b/w Schiller & Years of Smoking",fontsize=22)
b.set_xlabel("Years of Smoking",fontsize=20)
b.set_ylabel("Age of Patient",fontsize=20)
```

```
plt.subplot(2,2,3)
b=sns.boxplot(hue=impute['Schiller'],x=impute['smokepack_yr_cat'],y=impute['Age'])
b.axes.set_title("Box Plot b/w Schiller & Years of Smoking",fontsize=22)
b.set_xlabel("No of Smoking packets in a year",fontsize=20)
b.set_ylabel("Age of Patient",fontsize=20)
plt.show()
```

Inferences
- The person who smoke with more age are prone to be tested as positive in Schiller test.
- The person who smoke for atleast one year are more prone to be test as positive in schiller test
- The person who smoke more number of packets a year with more age are prone to be tested as postive in Schiller test

3. Birth control attributes
- Hormonal_Contraceptives - Indicates usage of the contraceptives or not
- Hormonal_Contraceptives_years - It indicates the years in usage of the contraceptives
- IUD - It indicates the usage of IUD contraceptives or not
- IUD_yr_cat - It indicates the years in usage of the contraceptives

```
plt.subplots(2,2,figsize=(20,16))
plt.subplot(2,2,1)
c=sns.boxplot(hue=impute['Schiller'],x=impute['Hormonal_Contraceptives'],y=impute['Age'])
c.axes.set_title('Box plot b/w Schiller & Hormonal Contraceptives',fontsize=20)
c.set_xlabel('Hormonal Contraceptives',fontsize=20)
c.set_ylabel('Age of Patient',fontsize=20)
plt.subplot(2,2,2)
years=pd.cut(impute.Hormonal_Contraceptives_years,[0,2,4,6,8],labels=['0-2','2-4','4-6','6-8'])
c=sns.boxplot(x=years,y='Age',hue='Schiller',data=impute)
c.axes.set_title('Box plot b/w Schiller & No. of year using Hormonal Contraceptives',fontsize=20)
c.set_xlabel('No. of year Hormonal Contraceptives',fontsize=20)
c.set_ylabel('Age of Patient',fontsize=20)
plt.subplot(2,2,3)
c=sns.boxplot(hue=impute['Schiller'],x=impute['IUD'],y=impute['Age'])
c.axes.set_title('Box plot b/w Schiller & IUD',fontsize=20)
c.set_xlabel('IUD',fontsize=20)
c.set_ylabel('Age of Patient',fontsize=20)
plt.subplot(2,2,4)
c=sns.boxplot(x='IUD_yr_cat',y='Age',hue='Schiller',data=impute)
c.axes.set_title('Box plot b/w Schiller & No. of year using IUD',fontsize=20)
c.set_xlabel('No. of year IUD',fontsize=20)
c.set_ylabel('Age of Patient',fontsize=20)
c=sns.boxplot()
plt.show()
```

Inferences
- The persons who did not use the hormonal contraceptives and with more age are high in number, who show positive for schiller test
- The patients with 2 years of usage in hormonal contraceptives and the average age between 25yrs & 35yrs show postive for schiller test.
- The persons who did not use IUD and with age 25yrs & 35 yrs, show positive for schiller test

4. STD attributes

```
plt.subplots(4,3,figsize=(20,30))
var=['Dummy','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']
for i in np.arange(1,13):
plt.subplot(4,3,i)
```

```
d=sns.boxplot(hue='Schiller',x=var[i],y='Age',data=impute)
d.axes.set_title('Schiller & ' + var[i],fontsize=15)
d.set_xlabel(var[i],fontsize=20)
d.set_ylabel('Age of the Patient',fontsize=20)

## Inferences
- In all the plots we can clearly the person with more age are prone to be tested as Positive in Schiller test
- We can not predict the effect of individual STDs are the data is not sufficient as there is class in imbalance

## OUTLIER TREATMENT by using IQR

numerical=['Age',
'No_of_sex_partner',
'First_sexual_intercourse',
'No_pregnancies',
'Smokes_yrs',
'Smokes_packs_yr',
'Hormonal_Contraceptives_years',
'IUD_years']

df_impute[numerical]=df_impute[numerical].astype('float64')

df_impute[numerical].plot(kind='box',subplots=True, layout=(4,4), fontsize=8, figsize=(14,14))
plt.show()

IQR=df_impute[numerical].describe().T['75%']-df_impute[numerical].describe().T['25%']

min,max=[df_impute[numerical].describe().T['25%']-(IQR*1.5),df_impute[numerical].describe().T['75%']+(IQR*1.5)]

for i in numerical:
print('range of',i,'b/w',min[i],'and',max[i])

for i in numerical:
df_impute[i][df_impute[i]>max[i]]=max[i]
df_impute[i][df_impute[i]<min[i]]=min[i]

df_impute[numerical].plot(kind='box',subplots=True, layout=(4,4), fontsize=8, figsize=(14,14))
plt.show()

## EDA - Correlation Plot

import seaborn as sns

plt.title('Correlation Plot of Independent Variables & Target Column(SCHILLER)',fontsize=30)
impute.corr()['Schiller'].sort_values()[:-6].plot(kind='barh',figsize=(20,20),fontsize=20)
plt.show()

# ChiSquare Test

from scipy.stats import chisquare,chi2_contingency

cat_col=[  'Smokes','Hormonal_Contraceptives','IUD','STDs', 'STDs_number','STDs_condylomatosis',
'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_No_of_diagnosis',
'Dx_Cancer', 'Dx_CIN', 'Dx_HPV', 'Dx']
chi_pvalue=[]
```

```
chi_name=[]
def chi_sq(cat):
cont=pd.crosstab(impute['Schiller'],impute[cat])
chi_pvalue.append(chi2_contingency(cont)[1])
chi_name.append(cat)

for cat in cat_col:
chi_sq(cat)

chi_data=pd.DataFrame()
chi_data['Pvalue']=chi_pvalue
chi_data.index=chi_name

plt.figure(figsize=(20,20))
plt.title('P-Values of Chisquare with " Schiller Test" as Target Categorical Attribute',fontsize=20)
x=chi_data.Pvalue.sort_values().plot(kind='barh')
x.set_xlabel('P-Values',fontsize=20)
x.set_ylabel('Independent Categorical Attributes',fontsize=20)
plt.show()

### Checking the Model Performance After Imputation

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report

impute=df_impute.copy()
impute=df_impute.astype('float')

impute.info()

x_features=list(df.columns)
x_features.remove('Schiller')

x_features_categorical=['Smokes','Hormonal_Contraceptives','IUD','STDs','STDs_condylomatosis','STDs_cervical_condylo
matosis',
'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','Dx_Cancer','Dx_CIN','Dx_HPV','Dx']

x_features_numerical=[i for i in x_features if i not in x_features_categorical]
len(x_features_categorical)+len(x_features_numerical)

impute['age_cat']=pd.cut(impute.Age,[0,10,20,30,40,50,60,100],labels=['<10','10-20','20-30','30-40','40-50','50-60','>60'])
impute=impute.drop(['Age'],axis=1)

impute.Smokes_yrs=impute.Smokes_yrs.astype('float64')
impute.IUD_years=impute.IUD_years.astype('float64')
impute.Smokes_packs_yr=impute.Smokes_packs_yr.astype('float64')
impute['smokes_yr_cat']=pd.cut(impute.Smokes_yrs,[-1,2,5,10,15,20,25,50],labels=['<2','2-5','5-10','10-15','15-20','20-
25','>25'])
impute=impute.drop(['Smokes_yrs'],axis=1)
impute['smokepack_yr_cat']=pd.cut(impute.Smokes_packs_yr,[-1,2,5,10,20,50],labels=['<2','2-5','5-10','10-20','>20'])
impute=impute.drop(['Smokes_packs_yr'],axis=1)
impute['IUD_yr_cat']=pd.cut(impute.IUD_years,[-1,2,4,6,8,10,15,20],labels=['<2','2-4','4-6','6-8','8-10','10-15','>15'])
impute=impute.drop(['IUD_years'],axis=1)

impute[x_features_categorical]=impute[x_features_categorical].replace(0,'no')
impute[x_features_categorical]=impute[x_features_categorical].replace(1,'yes')
impute=pd.get_dummies(impute)
impute.head()

impute.info()

x=impute.drop('Schiller',axis=1)
y=impute[['Schiller']]

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

```
## Decision Tree

dt=DecisionTreeClassifier()

dt.fit(x_train,y_train)
dt.score(x_train,y_train)

print('Confusion Matrix')
ypred=dt.predict(x_test)
confusion_matrix(y_test,ypred)

print(classification_report(y_test,ypred))
score=dt.score(x_test,y_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)
print('F1 Score',f1_score(y_test,ypred))

resultsDf = pd.DataFrame({'Classifier':['Decision Tree'], 'Test Score': [score],'Recall':[recall],'Precision':[precision],'F1
Score':[f1score]},index=[1])
resultsDf

## K-Nearest Neighbours

from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
print(knn.score(x_train,y_train))
print(knn.score(x_test,y_test))

ypred=knn.predict(x_test)
confusion_matrix(y_test,ypred)

score=knn.score(x_test,y_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)
print(classification_report(y_test,ypred))
print('F1 Score',f1_score(y_test,ypred))

tempResultsDf = pd.DataFrame({'Classifier':['K Nearest Neighbours'], 'Test Score':
[score],'Recall':[recall],'Precision':[precision],'F1 Score':[f1score]},index=[2])
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf

## Naive Bayes

from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()

nb.fit(x_train,y_train)
print(nb.score(x_train,y_train))
print(nb.score(x_test,y_test))

ypred=nb.predict(x_test)
print('Confusion Matrix \n',confusion_matrix(y_test,ypred))

score=nb.score(x_test,y_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)
print(classification_report(y_test,ypred))
print('F1 Score',f1_score(y_test,ypred))

tempResultsDf = pd.DataFrame({'Classifier':['Naive Bayes'], 'Test Score': [score],'Recall':[recall],'Precision':[precision],'F1
Score':[f1score]},index=[3])
```

```
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf

## Logistic Regression

df_impute.columns

impute=df_impute.copy()
impute['age_cat']=pd.cut(impute.Age,[0,10,20,30,40,50,60,100],labels=['<10','10-20','20-30','30-40','40-50','50-60','>60'])
impute=impute.drop(['Age'],axis=1)
impute.Smokes_yrs=impute.Smokes_yrs.astype('float64')
impute.IUD_years=impute.IUD_years.astype('float64')
impute.Smokes_packs_yr=impute.Smokes_packs_yr.astype('float64')
impute[x_features_categorical]=impute[x_features_categorical].replace(0,'no')
impute[x_features_categorical]=impute[x_features_categorical].replace(1,'yes')
impute=impute.drop(['Smokes_yrs','Smokes_packs_yr', 'IUD_years'],axis=1)
impute=pd.get_dummies(impute,drop_first=True)
x_features=list(impute)
x_features.remove('STDs_yes')
x_features.remove('Schiller')
x_features.remove('STDs_No_of_diagnosis')
x_features.remove('First_sexual_intercourse')
data_mat = impute[x_features].as_matrix()
vif = [ variance_inflation_factor( data_mat,i) for i in range(data_mat.shape[1]) ]
vif_factors = pd.DataFrame()
vif_factors['column'] = list(x_features)
vif_factors['vif'] = vif

x=impute[x_features]
#x=impute_complete.drop('Schiller',axis=1)
y=impute[['Schiller']]
test_size = 0.30 # taking 70:30 training and test set
seed = 5  # Random numbmer seeding for reapeatability of the code
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)


# Fit the model on 30%
model = LogisticRegression()
model.fit(x_train, y_train)
ypred = model.predict(x_test)
model_score = model.score(x_test, y_test)
print(model_score)
#y_predict
print(metrics.confusion_matrix(y_test, ypred))
print(classification_report(y_test, ypred))
print('F1 Score',f1_score(y_test,ypred))

score=model.score(x_test,y_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)

tempResultsDf = pd.DataFrame({'Classifier':['Logistic Regression'], 'Test Score':
[score],'Recall':[recall],'Precision':[precision],'F1 Score':[f1score]},index=[4])
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf

## Inference
- Precision and Recall for all the above the base models are not  above 0.30.
- The F1 scores for all the above base models are not above 0.30.
- The models are biased to the larger class in the target variabe as there is a significant amount of class imbalance.
- Hence there is a need of over sampling of data which has the lower class.

plt.title('Class Imbalance in Target Variable-Schiller Test')
plt.pie(impute.Schiller.value_counts(),autopct='%.2f%%',labels=['Negative','Positive'])
plt.show()
```

Random Over sampling
- The current class imbalance in the target column in the ratio 8.59:91.41 .
- The class ratio after the over sampling of the minority is 40:60 .

```
impute_sample=df_impute.copy()
impute_sample_data_0=impute_sample[impute_sample.Schiller==0]
impute_sample_data_1=impute_sample[impute_sample.Schiller==1]
count_class_0, count_class_1 = impute_sample.Schiller.value_counts()

int(np.round((count_class_0*0.4)/0.6))

df_class_1_over = impute_sample_data_1.sample(int(np.round((count_class_0*0.4)/0.6)), replace=True)
df_test_over = pd.concat([impute_sample_data_0, df_class_1_over], axis=0)

print('Random Over-sampling:')
print(df_test_over.Schiller.value_counts())

plt.title('Class Imbalance in Target Variable-Schiller Test after Over Sampling')
plt.pie(df_test_over.Schiller.value_counts(),autopct='%.2f%%',labels=['Negative','Positive'])
plt.show()

df_test_over=df_test_over.astype('float')
df_test_over[x_features_categorical]=df_test_over[x_features_categorical].replace(0,'no')
df_test_over[x_features_categorical]=df_test_over[x_features_categorical].replace(1,'yes')
df_test_over.info()

impute=df_test_over.copy()

impute=pd.get_dummies(impute,drop_first=True)
impute.info()
```

Logistic Regression

```
x_features=list(impute)
x_features.remove('Age')
x_features.remove('STDs_yes')
x_features.remove('Schiller')
x_features.remove('STDs_No_of_diagnosis')
x_features.remove('First_sexual_intercourse')
data_mat = impute[x_features].as_matrix()
data_mat.shape
vif = [ variance_inflation_factor( data_mat,i) for i in range(data_mat.shape[1]) ]
vif_factors = pd.DataFrame()
vif_factors['column'] = list(x_features)
vif_factors['vif'] = vif

x=impute[x_features]
y=impute[['Schiller']]
test_size = 0.30 # taking 70:30 training and test set
seed = 7  # Random numbmer seeding for reapeatability of the code
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)


# Fit the model on 30%
model = LogisticRegression()
model.fit(x_train, y_train)
ypred= model.predict(x_test)
model_score = model.score(x_test, y_test)
print('Model Score',model_score)
print(metrics.confusion_matrix(y_test, y_predict))
print('F1 score',f1_score(y_test, y_predict))

score=model.score(x_test,y_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)
```

```
resultsDf = pd.DataFrame({'Classifier':['Logistic Regression'], 'Test Score': [score],'Recall':[recall],'Precision':[precision],'F1 Score':[f1score]},index=[1])
resultsDf

## Decision Tree

x=impute.drop(['Schiller'],axis=1)
y=impute[['Schiller']]
test_size = 0.30 # taking 70:30 training and test set
seed = 7  # Random numbmer seeding for repeatability of the code
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)
dt=DecisionTreeClassifier(random_state=7)
dt.fit(x_train,y_train)
ypred=dt.predict(x_test)
print('Model Score',dt.score(x_test,y_test))
print(confusion_matrix(y_test,ypred))
print('F1 score',f1_score(y_test,ypred))

pd.DataFrame(dt.feature_importances_,index=x_train.columns).sort_values(by=0,ascending=True).plot(kind='barh',figsize=(15,8))

ft_selection=pd.DataFrame(dt.feature_importances_,index=x_train.columns,columns=['value'])
dt_features=list(ft_selection[ft_selection.value>0.01].index)

x=impute[dt_features]
y=impute[['Schiller']]
test_size = 0.30 # taking 70:30 training and test set
seed = 7  # Random numbmer seeding for repeatability of the code
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)
dt=DecisionTreeClassifier(random_state=7)
dt.fit(x_train,y_train)
ypred=dt.predict(x_test)
print('Model Score',dt.score(x_test,y_test))
print(confusion_matrix(y_test,ypred))
print('F1 Score',f1_score(y_test,ypred))

score=dt.score(x_test,y_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)

tempResultsDf = pd.DataFrame({'Classifier':['Decision Tree'], 'Test Score': [score],'Recall':[recall],'Precision':[precision],'F1 Score':[f1score]},index=[2])
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf

### Inferences
- We can see that the accuracy scores of Decision Tree (0.92) is more than Logistic Regression (0.66).
- The F1 scores of Decision Tree (0.90) is more than the scores of Logistic Regression (0.45).
- Hence we will confirm the base model as Decision Tree.
- We will use the ensemble techniques to improve the model performance.

## Ensemble Methods - Random Forest

rf=RandomForestClassifier()
rf.fit(x_train,y_train)
print('Model Score',rf.score(x_test,y_test))
ypred=rf.predict(x_test)
print('Confusion Matrix \n', confusion_matrix(y_test,ypred))
print('F1 Score',f1_score(y_test,ypred))

len(x_train.columns)

#### Hyper Parameter Tuning with Grid Search Cross Validation
```

```
param_grid=dict(n_estimators=np.array([10, 20, 30,
40,50]),max_depth=np.arange(1,17,2),min_samples_split=np.arange(2,6),min_samples_leaf=np.arange(1,5),max_features=n
p.arange(1,10))
rf=RandomForestClassifier(random_state=seed)
kfold = KFold(n_splits=5, random_state=seed)
gs= GridSearchCV(estimator=rf, param_grid=param_grid, scoring='accuracy', cv=kfold)
gs_result=gs.fit(x_train,y_train)
gs_result.best_params_
gs.score(x_train,y_train)

print('Model Score',gs.score(x_test,y_test))
ypred=gs.predict(x_test)
print('Confusion Matrix \n', confusion_matrix(y_test,ypred))
print('F1 Score',f1_score(y_test,ypred))

score=gs.score(x_test,y_test)
ypred=gs.predict(x_test)
f1score=f1_score(y_test,ypred)
precision=precision_score(y_test,ypred)
recall=recall_score(y_test,ypred)

ResultsDf = pd.DataFrame({'Classifier':['Random Forest'], 'Test Score': [score],'Recall':[recall],'Precision':[precision],'F1
Score':[f1score]},index=[1])
ResultsDf
```

6.2 APPENDIX B GLOSSARY

Biopsy - An examination of tissue removed from a living body to discover the presence, cause, or extent of a disease.
Cancer - Abnormal cells divide uncontrollably and destroy body tissue.
Cytology - Diagnosing diseases by looking at single cells and small clusters of cells.
False Positive - An outcome where the model incorrectly predicts the positive class.
False Negative - An outcome where the model incorrectly predicts the negative class.
Hinselmann - A medical diagnostic procedure to examine an illuminated, magnified view of the cervix and the tissues.
True Positive - An outcome where the model correctly predicts the positive class.
True Negative - An outcome where the model correctly predicts the negative class.
Outliers – The data that are not fall in between Q1 and Q3 quartile are called outliers.

6.3 APPENDIX C ACRONYMS & ABBREVATIONS

CART – Classification and Regression Trees.
IUD - Intra-Uterine Device.
STD - Sexually Transmitted Diseases.
HIV - Human Immuno-deficiency Virus.
AIDS - Acquired Immune Deficiency Syndrome.
HPV - Human Papilloma Virus.
IQR - Inter-Quartile Range.

6.4 APPENDIX D REFERENCES

https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29#

https://www.webmd.com/cancer/cervical-cancer/cervical-cancer#1

https://ir.uiowa.edu/cgi/viewcontent.cgi?article=3190&context=etd

https://www.slideshare.net/LifecareCentre/burden-of-cervical-cancer-other-hpv-related-diseases-indian-perspective-dr-sharda-jain-lifecare-centre-64387354

https://en.wikipedia.org/wiki/Cervical_cancer