RA1911028010069 - Final Assignment

Model Predicting Cevical Cancer

Loading dataset

In [120]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import tree
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("kag_risk_factors_cervical_cancer.csv")
df.head()
```

Out[120]:

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STDs: Time since first diagnosis	STDs: Time since last diagnosis
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?
2	34	1.0	?	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	...	?	?
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	...	?	?

5 rows × 36 columns

In [121]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 858 entries, 0 to 857
Data columns (total 36 columns):
Age                                   858 non-null int64
Number of sexual partners             858 non-null object
First sexual intercourse              858 non-null object
Num of pregnancies                    858 non-null object
Smokes                                858 non-null object
Smokes (years)                        858 non-null object
Smokes (packs/year)                   858 non-null object
Hormonal Contraceptives               858 non-null object
Hormonal Contraceptives (years)       858 non-null object
IUD                                   858 non-null object
IUD (years)                           858 non-null object
STDs                                  858 non-null object
STDs (number)                         858 non-null object
STDs:condylomatosis                   858 non-null object
STDs:cervical condylomatosis          858 non-null object
STDs:vaginal condylomatosis           858 non-null object
STDs:vulvo-perineal condylomatosis    858 non-null object
STDs:syphilis                         858 non-null object
STDs:pelvic inflammatory disease      858 non-null object
STDs:genital herpes                   858 non-null object
STDs:molluscum contagiosum            858 non-null object
STDs:AIDS                             858 non-null object
STDs:HIV                              858 non-null object
STDs:Hepatitis B                      858 non-null object
STDs:HPV                              858 non-null object
STDs: Number of diagnosis             858 non-null int64
STDs: Time since first diagnosis      858 non-null object
STDs: Time since last diagnosis       858 non-null object
Dx:Cancer                             858 non-null int64
Dx:CIN                                858 non-null int64
Dx:HPV                                858 non-null int64
Dx                                    858 non-null int64
Hinselmann                            858 non-null int64
Schiller                              858 non-null int64
Citology                              858 non-null int64
Biopsy                                858 non-null int64
dtypes: int64(10), object(26)
memory usage: 241.4+ KB
```

Replacing '?' and dropping columns with large number of null fields

In [122]:
```
df = df.replace('?',np.nan)
df.isnull().sum()

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

Removing all null fields

In [123]:
```
mEntries = df.isnull().sum(axis=1).tolist()

nMapping = dict((x, mEntries.count(x)) for x in mEntries)
sorted(nMapping.items())

df = df.dropna()
df.isnull().sum()
```

Out[123]:

```
Age                                   0
Number of sexual partners             0
First sexual intercourse              0
Num of pregnancies                    0
Smokes                                0
Smokes (years)                        0
Smokes (packs/year)                   0
Hormonal Contraceptives               0
Hormonal Contraceptives (years)       0
IUD                                   0
IUD (years)                           0
STDs                                  0
STDs (number)                         0
STDs:condylomatosis                   0
STDs:cervical condylomatosis          0
STDs:vaginal condylomatosis           0
STDs:vulvo-perineal condylomatosis    0
STDs:syphilis                         0
STDs:pelvic inflammatory disease      0
STDs:genital herpes                   0
STDs:molluscum contagiosum            0
STDs:AIDS                             0
STDs:HIV                              0
STDs:Hepatitis B                      0
STDs:HPV                              0
STDs: Number of diagnosis             0
Dx:Cancer                             0
Dx:CIN                                0
Dx:HPV                                0
Dx                                    0
Hinselmann                            0
Schiller                              0
Citology                              0
Biopsy                                0
dtype: int64
```

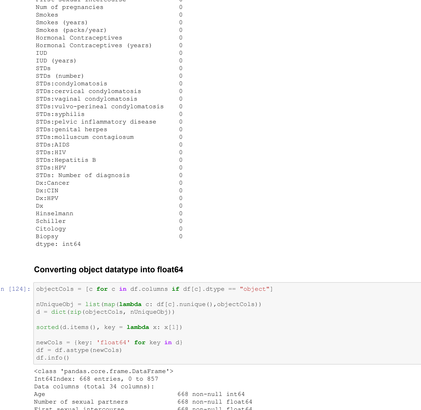
Converting object datatype into float64

In [124]:
```
objectCols = [c for c in df.columns if df[c].dtype == "object"]

nUniqueObj = list(map(lambda c: df[c].nunique(),objectCols))
d = dict(zip(objectCols, nUniqueObj))

sorted(d.items(), key = lambda x: x[1])

newCols = {key: 'float64' for key in d}
df = df.astype(newCols)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 668 entries, 0 to 857
Data columns (total 34 columns):
Age                                   668 non-null int64
Number of sexual partners             668 non-null float64
First sexual intercourse              668 non-null float64
Num of pregnancies                    668 non-null float64
Smokes                                668 non-null float64
Smokes (years)                        668 non-null float64
Smokes (packs/year)                   668 non-null float64
Hormonal Contraceptives               668 non-null float64
Hormonal Contraceptives (years)       668 non-null float64
IUD                                   668 non-null float64
IUD (years)                           668 non-null float64
STDs                                  668 non-null float64
STDs (number)                         668 non-null float64
STDs:condylomatosis                   668 non-null float64
STDs:cervical condylomatosis          668 non-null float64
STDs:vaginal condylomatosis           668 non-null float64
STDs:vulvo-perineal condylomatosis    668 non-null float64
STDs:syphilis                         668 non-null float64
STDs:pelvic inflammatory disease      668 non-null float64
STDs:genital herpes                   668 non-null float64
STDs:molluscum contagiosum            668 non-null float64
STDs:AIDS                             668 non-null float64
STDs:HIV                              668 non-null float64
STDs:Hepatitis B                      668 non-null float64
STDs:HPV                              668 non-null float64
STDs: Number of diagnosis             668 non-null int64
Dx:Cancer                             668 non-null int64
Dx:CIN                                668 non-null int64
Dx:HPV                                668 non-null int64
Dx                                    668 non-null int64
Hinselmann                            668 non-null int64
Schiller                              668 non-null int64
Citology                              668 non-null int64
Biopsy                                668 non-null int64
dtypes: float64(24), int64(10)
memory usage: 182.7 KB
```

Plotting boxplot to detect outliers

In [125]:
```
plt.figure(figsize=(20,5))
df.iloc[:,:].boxplot()
ax = plt.gca()
plt.setp(ax.get_xticklabels(), rotation=90)
plt.show()
```



Removing outliers

In [126]:
```
df.drop(df.index[df['Age'] > 50], inplace = True)
df.drop(df.index[df['Number of sexual partners'] > 10], inplace = True)
```

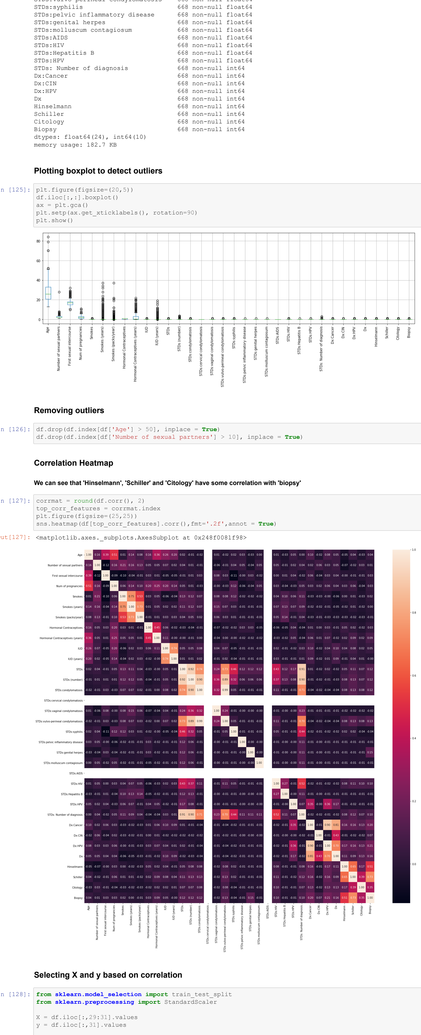
Correlation Heatmap

We can see that 'Hinselmann', 'Schiller' and 'Citology' have some correlation with 'biopsy'

In [127]:
```
corrmat = round(df.corr(), 2)
top_corr_features = corrmat.index
plt.figure(figsize=(25,25))
sns.heatmap(df[top_corr_features].corr(),fmt='.2f',annot = True)
```

Out[127]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24ff008f198>
```



Selecting X and y based on correlation

In [128]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = df.iloc[:,29:31].values
y = df.iloc[:,31].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training model using random forest classifier

In [129]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(criterion='entropy', random_state=0)
rfc.fit(X_train, y_train)
```

Out[129]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                       oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Calculating accuracy of model

In [114]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
k = 5
kf = KFold(n_splits=k, random_state=None)
result = cross_val_score(rfc , X_train, y_train, cv = kf)
print("Accuracy: {:.2f}%".format(result.mean()*100))
```

```
Accuracy: 94.14%
```