PES UNIVERSITY

IBMZDATATHON

CERVICAL CANCER RISK DETECTION .

Byte Me-128

PES2UG20CS204	MOULYA SHETTY
PES2UG20CS205	MRUNMAYI DESHPANDE
PES2UG20CS208	SHREYA NADELLA
PES2UG20CS211	NAMEETA KURUWATTI
PES2UG20CS232	NIVEDITA VENKAT
PES2UG20CS333	SHRIYA YS

CERVICAL CANCER

Risk detection for Cervical Cancer.

Cervical cancer happens when cells change in women’s cervix, which connects the uterus and vagina. This cancer can affect the deeper tissues of their cervix and may spread to other parts of their body (metastasize), often the lungs, liver, bladder, vagina, and rectum.

Most cases of cervical cancer are caused by infection with human papillomavirus (HPV), which is preventable with a vaccine.

About 11,000 new cases of invasive cervical cancer are diagnosed each year.The number of new cervical cancer cases has been declining steadily over the past decades. Although it is the most preventable type of cancer, each year cervical cancer kills about 300,000 women worldwide. Risk detection of the cancer helps the at-risk women reach out for early treatment which could save their lives.

PROBLEM STATEMENT

We are all aware of cancer is a deadly disease to treat and hard to diagnose before it is too late. Its nature of recurring after a while is very difficult to cope with. So why not prevent its occurrence as a whole? On researching, we came across information that cervical cancer was one of the most preventable types of cancer and concluded to work on this project.

What it does: Classifying the risk of cervical cancer based on certain factors which do not require medical testing. In fact a short interrogation of the person who wants to test their health would be enough for the model to predict the risk.

The result of the project can guide the user and inform them of the risk. It is then the users choice to give it immediate attention and visit a hospital for a biopsy.

DATA OBSERVATION

Dataset contains 858 rows and 36 columns.
Some of these attributes contained values that were either missing at random or missing not at random

Which takes us to our next step data preprocessing

Data frame

```
[4]: df_full
```

[4]:

	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	ST Ti since I diagnc
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	
2	34	1.0	?	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	...	?	
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	...	?	
...	...	...	...	...	...	...	...	...	...	...	...	...	
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	0.0	...	?	
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?	
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?	
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	...	?	

858 rows × 36 columns

DATA PREPROCESSING

All the missing values are pre-processed
Null values of contiguous attributes are filled using the median of the respective column
Null values of the categorical attributes are filled with an identifier (1)

```
# for continuous variable
df['Number of sexual partners'] = df['Number of sexual partners'].fillna(df['Number of sexual partners'].median())
df['First sexual intercourse'] = df['First sexual intercourse'].fillna(df['First sexual intercourse'].median())
df['Num of pregnancies'] = df['Num of pregnancies'].fillna(df['Num of pregnancies'].median())
df['Smokes'] = df['Smokes'].fillna(1)
df['Smokes (years)'] = df['Smokes (years)'].fillna(df['Smokes (years)'].median())
df['Smokes (packs/year)'] = df['Smokes (packs/year)'].fillna(df['Smokes (packs/year)'].median())
df['Hormonal Contraceptives'] = df['Hormonal Contraceptives'].fillna(1)
df['Hormonal Contraceptives (years)'] = df['Hormonal Contraceptives (years)'].fillna(df['Hormonal Contraceptives (y
df['IUD'] = df['IUD'].fillna(0) # Under suggestion
df['IUD (years)'] = df['IUD (years)'].fillna(0) #Under suggestion
df['STDs'] = df['STDs'].fillna(1)
df['STDs (number)'] = df['STDs (number)'].fillna(df['STDs (number)'].median())
df['STDs:condylomatosis'] = df['STDs:condylomatosis'].fillna(df['STDs:condylomatosis'].median())
df['STDs:cervical condylomatosis'] = df['STDs:cervical condylomatosis'].fillna(df['STDs:cervical condylomatosis'].m
df['STDs:vaginal condylomatosis'] = df['STDs:vaginal condylomatosis'].fillna(df['STDs:vaginal condylomatosis'].medi
df['STDs:vulvo-perineal condylomatosis'] = df['STDs:vulvo-perineal condylomatosis'].fillna(df['STDs:vulvo-perineal
df['STDs:syphilis'] = df['STDs:syphilis'].fillna(df['STDs:syphilis'].median())
df['STDs:pelvic inflammatory disease'] = df['STDs:pelvic inflammatory disease'].fillna(df['STDs:pelvic inflammatory
df['STDs:genital herpes'] = df['STDs:genital herpes'].fillna(df['STDs:genital herpes'].median())
df['STDs:molluscum contagiosum'] = df['STDs:molluscum contagiosum'].fillna(df['STDs:molluscum contagiosum'].median(
df['STDs:AIDS'] = df['STDs:AIDS'].fillna(df['STDs:AIDS'].median())
df['STDs:HIV'] = df['STDs:HIV'].fillna(df['STDs:HIV'].median())
df['STDs:Hepatitis B'] = df['STDs:Hepatitis B'].fillna(df['STDs:Hepatitis B'].median())
df['STDs:HPV'] = df['STDs:HPV'].fillna(df['STDs:HPV'].median())
df['STDs: Time since first diagnosis'] = df['STDs: Time since first diagnosis'].fillna(df['STDs: Time since first d
df['STDs: Time since last diagnosis'] = df['STDs: Time since last diagnosis'].fillna(df['STDs: Time since last diag
```

MODEL DESIGN AND TRAINING

Using a cervical cancer dataset, correlations between each attribute and the target variable were found. Features holding a great significance over the outcome were thus identified. These features were then normalized and fit into a classification model to predict the need for a biopsy or not.

The model is a sequential linear developed from the keras library. An adam optimizer is used along with a binary crossentropy function to calculate loss. Dropouts of value 0.5 is used to prevent overfitting of the dataset in the model

A cross validation technique known as kfold split was used to split the training data into k=5 folds. Thus, 5 predictions from the model was seen and an average was taken to calculate the final accuracy

[34]:
```python
#KFOLD = 5
from sklearn.model_selection import KFold
kf=KFold(n_splits=5)
kfold_array_train=[]
kfold_array_test=[]
for train_index, test_index in kf.split(df_data.index):
    kfold_array_train.append(df_data.loc[train_index,:])
    kfold_array_test.append(df_data.loc[test_index,:])
```

```
Epoch 1/20
3/3 - 0s - loss: 0.6635 - accuracy: 0.7887 - val_loss: 0.6003 - val_accuracy: 0.9130 - 391ms/epoch - 130ms/step
Epoch 2/20
3/3 - 0s - loss: 0.5548 - accuracy: 0.9399 - val_loss: 0.4761 - val_accuracy: 0.9130 - 23ms/epoch - 8ms/step
Epoch 3/20
3/3 - 0s - loss: 0.4153 - accuracy: 0.9399 - val_loss: 0.3390 - val_accuracy: 0.9130 - 21ms/epoch - 7ms/step
Epoch 4/20
3/3 - 0s - loss: 0.2759 - accuracy: 0.9399 - val_loss: 0.2631 - val_accuracy: 0.9130 - 22ms/epoch - 7ms/step
Epoch 5/20
3/3 - 0s - loss: 0.2077 - accuracy: 0.9399 - val_loss: 0.2751 - val_accuracy: 0.9130 - 22ms/epoch - 7ms/step
Epoch 6/20
3/3 - 0s - loss: 0.2086 - accuracy: 0.9399 - val_loss: 0.3062 - val_accuracy: 0.9130 - 25ms/epoch - 8ms/step
Epoch 7/20
3/3 - 0s - loss: 0.2254 - accuracy: 0.9399 - val_loss: 0.3065 - val_accuracy: 0.9130 - 42ms/epoch - 14ms/step
Epoch 8/20
3/3 - 0s - loss: 0.2074 - accuracy: 0.9399 - val_loss: 0.2834 - val_accuracy: 0.9130 - 44ms/epoch - 15ms/step
Epoch 9/20
3/3 - 0s - loss: 0.1860 - accuracy: 0.9399 - val_loss: 0.2483 - val_accuracy: 0.9130 - 28ms/epoch - 9ms/step
Epoch 10/20
3/3 - 0s - loss: 0.1588 - accuracy: 0.9399 - val_loss: 0.2181 - val_accuracy: 0.9130 - 24ms/epoch - 8ms/step
```

```
Epoch 11/20
3/3 - 0s - loss: 0.1534 - accuracy: 0.9399 - val_loss: 0.2017 - val_accuracy: 0.9130 - 27ms/epoch - 9ms/step
Epoch 12/20
3/3 - 0s - loss: 0.1443 - accuracy: 0.9399 - val_loss: 0.1936 - val_accuracy: 0.9130 - 23ms/epoch - 8ms/step
Epoch 13/20
3/3 - 0s - loss: 0.1364 - accuracy: 0.9399 - val_loss: 0.1857 - val_accuracy: 0.9130 - 25ms/epoch - 8ms/step
Epoch 14/20
3/3 - 0s - loss: 0.1314 - accuracy: 0.9399 - val_loss: 0.1811 - val_accuracy: 0.9203 - 20ms/epoch - 7ms/step
Epoch 15/20
3/3 - 0s - loss: 0.1166 - accuracy: 0.9417 - val_loss: 0.1818 - val_accuracy: 0.9203 - 22ms/epoch - 7ms/step
Epoch 16/20
3/3 - 0s - loss: 0.1133 - accuracy: 0.9435 - val_loss: 0.1836 - val_accuracy: 0.9203 - 183ms/epoch - 61ms/step
Epoch 17/20
3/3 - 0s - loss: 0.1060 - accuracy: 0.9454 - val_loss: 0.1811 - val_accuracy: 0.9348 - 33ms/epoch - 11ms/step
Epoch 18/20
3/3 - 0s - loss: 0.1029 - accuracy: 0.9454 - val_loss: 0.1734 - val_accuracy: 0.9275 - 26ms/epoch - 9ms/step
Epoch 19/20
3/3 - 0s - loss: 0.1015 - accuracy: 0.9526 - val_loss: 0.1674 - val_accuracy: 0.9493 - 21ms/epoch - 7ms/step
Epoch 20/20
3/3 - 0s - loss: 0.0972 - accuracy: 0.9654 - val_loss: 0.1628 - val_accuracy: 0.9348 - 27ms/epoch - 9ms/step
```

MODEL ACCURACY

```
model saved to disk
6/6 [==============================] - 0s 8ms/step - loss: 0.1012 - accuracy: 0.9649


accuracy= 0.9649122953414917
total test case number:  171
total_num:  171
G1P1:  0
G0P1:  0
G1P0:  10
G0P0:  161
##########################
sensitivity:  0.0
specificity:  0.9415204678362573
false_positive_rate:  0.0
false_negative_rate:  1.0
FINAL ACCURCY = 0.9603563070297241
```

FINAL ACCURACY:96%