Machine Learning Project

on

Breast Cancer Prevalence

Submitted by

Ranjana Chakraverty

Registration No.: 12014011

Program Name: B.Tech. (CSE - Data Science (ML and AI))

Under the guidance of

Ved Prakash Chaubey

School of Computer Science and Engineering

Lovely Professional University, Phagwara

(March-April, 2023)

DECLARATION

I hereby declare that I have completed my machine learning project from 21st March 2023 to 26th April 2023 under the guidance of Ved Prakash Chaubey. I have worked with full dedication during this one month of and my learning outcomes fulfil the requirements of training for the award of degree of B.Tech. (CSE - Data Science (ML and AI)), Lovely Professional University, Phagwara.

Ranjana Chakraverty

Registration Number: 12014011

Date: 26th April 2023

ACKNOWLEDGEMENT

I would like to thank my professors at LPU of the School of Computer Science and Engineering for their support. I would like to thank the instructors at upGrad, specially Mr. Ved Prakash Chaubey, who was always available to give clear my doubts.

TABLE OF CONTENTS

OBJECTIVE AND SCOPE

Classifying images of the tumors in the breasts of women as malignant or benign.

The menace of cancer has been one of the most pressing issues in recent times. Numerous efforts have been put in by scientists to find a cure for cancer. While a perfect cure has not been found yet, parallel progress has been made to improve the detection of cancer since it the earlier a cancer is detected, the better it is for the patient. Early treatment prevents the oncoming more harmful later stages.

This project takes into consideration one of the cancers that plagues many women worldwide which is breast cancer. This cancer develops in the breast cells and progresses in stages. The disease takes the form of tumors which are abnormal growths of tissues within the breast. Tumors, however, may or may not be cancerous. The ones that are cancerous are called malignant and indicate the presence of the cancerous growth in the breast. The ones that are not cancerous are called benign and the pose no harm to the patient. My aim, in this project, will be to classify tumors as benign or malignant.

The efficacy of this operation in the real-world medical industry is enormous. Only after a tumor is classified can it be decided whether treatment should be provided to a patient or not. Incorrectly identifying tumors can cause loss of life and resources.

The dataset has been sourced from the UCI Machine Learning Repository. I have considered supervised learning as the learning technique for my machine learning model. This will involve the model learning from labelled columns. The goal will be to correctly classify a tumor as benign or malignant based on real valued attributes derived from a cell's nuclei. This will be a classification problem and logistic regression will be utilized to create the model.

Healthcare has always been one of the most cardinal considerations in humanity's quest for survival. In its development over centuries, the goal has always been to increase the accuracy in diagnosis and betterment in treatment. In recent years, the potential of machine learning in contributing to both factors have been observed. The project will try to show the important role that technology, particularly machine learning plays in improvement of healthcare services.

INTRODUCTION

Cancer occurs when changes called mutations take place in genes that regulate cell growth. The mutations let the cells divide and multiply in an uncontrolled way.

Breast cancer is cancer that forms in the cells of the breasts.

After skin cancer, breast cancer is the most common cancer diagnosed in women in the United States. Breast cancer can occur in both men and women, but it is far more common in women. Breast cancer occurs when breast cells develop mutations and begin to divide and multiply. People may first notice a lump in the breast, discoloration, texture changes, or other symptoms.

Typically, the cancer forms in either the lobules or the ducts of the breast.

Lobules are the glands that produce milk, and ducts are the pathways that bring the milk from the glands to the nipple. Cancer can also occur in the fatty tissue or the fibrous connective tissue within your breast.

The uncontrolled cancer cells often invade other healthy breast tissue and can travel to the lymph nodes under the arms. Once the cancer enters the lymph nodes, it has access to a pathway to move to other parts of the body. When breast cancer spreads to other parts of the body, it is said to have metastasized.

Substantial support for breast cancer awareness and research funding has helped create advances in the diagnosis and treatment of breast cancer. Breast cancer survival rates have increased, and the number of deaths associated with this disease is steadily declining due to factors such as earlier detection.

<u>Causes</u>

Breast cancer develops as a result of genetic mutations or damage to DNA. These can be associated with Trusted Source exposure to estrogen, inherited genetic defects, or inherited genes that can cause cancer, such as the *BRCA1* and *BRCA2* genes.

When a person is healthy, their immune system attacks any abnormal DNA or growths. When a person has cancer, this does not happen.

As a result, cells within breast tissue begin to multiply uncontrollably, and they do not die as usual. This excessive cell growth forms a tumor that deprives surrounding cells of nutrients and energy.

Breast cancer usually starts in the inner lining of the milk ducts or the lobules that supply them with milk. From there, it can spread to other parts of the body.

<u>Symptoms</u>

In its early stages, breast cancer may not cause any symptoms. In many cases, a tumor may be too small to be felt, but an abnormality can still be seen on a mammogram.

If a tumor can be felt, the first sign is usually a new lump in the breast that was not there before. However, not all lumps are cancer.

Each type of breast cancer can cause a variety of symptoms. Many of these symptoms are similar, but some can be different. Symptoms for the most common breast cancers include:

- a breast lump or tissue thickening that feels different from surrounding tissue and is new
- breast pain
- red or discoloured, pitted skin on the breast
- swelling in all or part of your breast
- a nipple discharge other than breast milk
- bloody discharge from the nipple
- peeling, scaling, or flaking of skin on your nipple or breast
- a sudden, unexplained change in the shape or size of the breast
- inverted nipple
- changes to the appearance of the skin on the breasts
- a lump or swelling under the arm

Stages

To stage breast cancer, doctors need to know:

- if the cancer is invasive or noninvasive
- how large the tumor is
- whether the lymph nodes are involved
- if the cancer has spread to nearby tissue or organs

Breast cancer has five main stages: stages 0 to 4.

Types of Breast Cancer

Breast cancer can be categorized in several ways. Most often it is classified by where it originates and whether it moves from that spot.

An easily identifiable characteristic of breast cancer is the type of cell it is formed in.

- Ductal carcinoma is formed in the cells lining the milk ducts.
- Lobular carcinoma is formed in the milk-producing lobules.

Another important characteristic of breast cancer is whether it invades surrounding tissue or stays where it originally formed.

- Noninvasive (in situ) breast cancer has not spread into surrounding tissue.
- Invasive (infiltrating) breast cancer has moved into the tissue surrounding it.

Diagnosis

Tests that can help the doctor diagnose breast cancer include:

- Mammogram. The most common way to see below the surface of your breast is with an imaging test called a mammogram.
- Ultrasound. A breast ultrasound uses sound waves to create a picture of the tissues deep in your breast. An ultrasound can help the doctor distinguish between a solid mass, such as a tumor, and a benign cyst.

The doctor may also suggest tests such as an MRI or a breast biopsy.

Treatment

- Surgery. Including mastectomy and lumpectomy.
- Radiation therapy. Radiation therapy is often used to destroy any cancer cells remaining in the breast or surrounding tissue after surgical removal of the cancer.
- Chemotherapy. Chemotherapy medication is often used to destroy cancer cells that have spread to distance parts of the body.
- Hormone therapy. Anti-estrogen and anti-progesterone therapy can be used to slow the growth of hormone receptor-positive tumors.
- Immunotherapy. Immunotherapy is a way to stimulate your immune system so that it is able to recognize and attack cancer cells. This is a growing area of research that is continuing to find new ways to treat cancer.
- Biological treatment. Targeted drugs to destroy specific types of breast cancer
- Other targeted therapy. For HER2-positive breast cancer, some types of targeted therapy can detect and disrupt the growth-promoting proteins on the surface of the cancer cells. This may help slow the growth of HER2-positive tumors.

Prevention

The steps one can take to lower the risk of getting breast cancer include the following:

- Limit alcohol to no more than one drink a day.
- Choose a healthy diet
- Maintain a moderate weight throughout your life.
- Stay physically active.
- Breastfeed if you can.
- Avoid hormone therapy for postmenopausal symptoms.
- Avoid breast implants.
- Talk to your doctor about medication or surgical treatments to reduce the chances you will get breast cancer if you are at high risk.

HARDWARE AND SOFTWARE USED

Hardware used was simply my laptop. I used Jupyter notebook for Python Coding.

PROBLEM STATEMENT

In its later stages, breast cancer is horrendous and causes death in most sufferers. However, a lot of neglect is required for the cancer to progress to such stages. When detected early and correctly, the death rate significantly reduces among patients and numerous treatments exist which can provide relief. This is the key: early detection. One of the most important aspects of the detection process is to classify a growth in the breast as cancerous or non-cancerous. This requires the expertise of a doctor. But, if the parameters which determine the type of tumor are fed to a machine learning model, this process can be entirely This project aims to observe which features are most helpful in predicting malignant or benign tumors and to see general trends that may aid us in model selection and hyper parameter selection. The goal is to classify whether the tumor is benign or malignant. To achieve this, I have used machine learning classification methods to fit a function that can predict the category a tumor belongs to.

The dataset that I will use has been sourced from the UCI Machine Learning repository.

Name: Breast Cancer Wisconsin (Diagnostic) Data Set

Characteristic: Multivariate

No missing values.

569 rows

32 columns (ID, diagnosis, 30 real-valued input features)

Diagnosis (M = malignant, B = benign)

Class distribution: 357 benign, 212 malignant

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

The dataset is in the .DATA format.

METHODOLOGY

The steps undertaken:

1. Making the data possible to work with: This dataset was sourced from the UCI Machine Learning Repository. The data present was in the .DATA format. I downloaded it, then I converted the .DATA file into an Excel workbook.
2. Importing the libraries and dataset to Python: pandas and numpy
3. Inspecting the dataset: The columns didn't have names so I gave them names which I found in the data dictionary that was available with the dataset.
4. Train-Test Split: Importing train_test_split from sklearn.
5. Feature Scaling: Using the StandardScaler from sklearn
6. Visualising correlations: Using the matplotlib library.
7. Removing highly correlated columns.
8. Model Building
9. Feature Selection
10. Evaluation metrics evaluation: Like accuracy, sensitivity, specificity, FPR.
11. Model Testing

The full code can be found in Annexure 1.

RESULTS

A successful logistic regression classifier was developed during this project. It can correctly predict whether a tumour is benign or malignant in the vast majority of patients. Classifiers such as this one are very useful and speed up the pipeline of medical diagnosis and treatment. Computer Assisted Diagnosis Systems or CAD which contain such classifiers for different diseases are proposed in the medical industry. With extremely high accuracy, these classifiers are valuable and are already in use in quite a few domains. They play a supporting role in diagnosis. Although CAD has been used in clinical environments for over 40 years, CAD usually does not substitute the doctor or other professional, but rather plays a supporting role. The professional (generally a radiologist) is generally responsible for the final interpretation of a medical image.

Evaluation Metrics observed:

- Accuracy: 95.32%
- Sensitivity: 91.30%
- Specificity: 98.03%
- False Positive Rate: 1.97%
- The ROC curve does not pass through the 45 degree diagonal

SUMMARY

This project began with a discussion on breast cancer. I understood its causes, its diagnosis, it’s treatment and its prevention. I understood how tumours may occur in the breast but the presence of tumours is not necessarily indicative of disease. This is because tumours are of two kinds: cancerous and non-cancerous. In oncological jargon, these are known as malignant and benign respectively. All tumours have certain attributes which can be related to them being classified as benign or malignant. Traditionally, doctors who have knowledge of these differentiating attributes classified tumours as benign or malignant after going through the mammogram of a patient. Efforts were made later to automate this process to reduce the doctor’s effort and diagnosis time, therefore speeding up detection and treatment. The attempts were fruitful in the form of CAD or Computer Assisted Diagnosis Systems which acted as a support system to the doctor. The critical component of a CAD is, of course, the classifier. In this project, I built a classifier akin to what may be used in an actual CAD. To build the classifier, I utilized machine learning. Since the model was built on the basis of labelled columns, it was supervised. Logistic regression, which is the basic classification algorithm was used to build this model. The model wound up well since it had high accuracy, sensitivity, and specificity. It has a low False Positive Rate as well.

Overall, the project fulfilled its objective and gave me a great understanding of logistic regression since I applied it practically. The confluence between healthcare and machine learning is something that I am passionate about. This topic also has a great scope as affordable, efficacious and efficient healthcare has been the quest of mankind since its inception and has become very important in the last century due to the increasing prevalence of diseases and disorders. I hope that I can take my interest forward by doing more projects on how valuable technology is in bolstering healthcare and its outreach.

BIBLIOGRAPHY

wikipedia.org

cancer.org

healthline.com

medicalnewstoday.com

cdc.gov

who.int

researchgate.net

pubmed.ncbi.nlm.nih.gov

ANNEXURE

Annexure 1 begins on the following page.

Breast Cancer Prevalence

With 32 predictor variables I will predict whether a particular tumour present in a woman's breast is cancerous or not. In oncological terminology, this is referred to as malignant and benign.

Importing and Merging Data

In [1]:

```
# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Importing Pandas and NumPy
import pandas as pd, numpy as np
```

In [3]:

```
# Importing dataset
data = pd.read_excel(r'C:\Users\white\Downloads\Book1.xlsx')
```

Inspecting the Dataframe

In [4]:

```
# Let's see the head of the dataset
data.head()
```

Out[4]:

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Colu
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0

5 rows × 32 columns

We observe that there are no names for the columns. I took a look at the data dictionary provided along with the dataset to understand how to name the columns. There are ten values measured for each nucleus. It appears that there are three nucleui in each cell. Therefore, I will perform a numerica naming of columns.

In [5]:

```
data.columns = ['id', 'result', 'radius1', 'texture1', 'perimeter1', 'area1', 'smoot
                'radius2', 'texture2', 'perimeter2', 'area2', 'smooth2', 'compact2',
                'radius3', 'texture3', 'perimeter3', 'area3', 'smooth3', 'compact3',
```

In [6]:

```
data.head()
```

Out[6]:

	id	result	radius1	texture1	perimeter1	area1	smooth1	compact1	concave1	num_con
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0

5 rows × 32 columns

In [7]:

```
# Let's check the dimensions of the dataframe
data.shape
```

Out[7]: (569, 32)

In [8]:

```
# let's look at the statistical aspects of the dataframe
data.describe()
```

Out[8]:

	id	radius1	texture1	perimeter1	area1	smooth1	compact1	con
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.0
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.1
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.4

8 rows × 31 columns

In [9]:

```
# Let's see the datatype of each column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             569 non-null    int64
 1   result         569 non-null    object
 2   radius1        569 non-null    float64
 3   texture1       569 non-null    float64
 4   perimeter1     569 non-null    float64
 5   area1          569 non-null    float64
 6   smooth1        569 non-null    float64
 7   compact1       569 non-null    float64
 8   concave1       569 non-null    float64
 9   num_concave_1  569 non-null    float64
```

```
 10  symmetry1      569 non-null    float64
 11  fractal1       569 non-null    float64
 12  radius2        569 non-null    float64
 13  texture2       569 non-null    float64
 14  perimeter2     569 non-null    float64
 15  area2          569 non-null    float64
 16  smooth2        569 non-null    float64
 17  compact2       569 non-null    float64
 18  concave2       569 non-null    float64
 19  num_concave_2  569 non-null    float64
 20  symmetry2      569 non-null    float64
 21  fractal2       569 non-null    float64
 22  radius3        569 non-null    float64
 23  texture3       569 non-null    float64
 24  perimeter3     569 non-null    float64
 25  area3          569 non-null    float64
 26  smooth3        569 non-null    float64
 27  compact3       569 non-null    float64
 28  concave3       569 non-null    float64
 29  num_concave_3  569 non-null    float64
 30  symmetry3      569 non-null    float64
 31  fractal3       569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

Data Preparation

Converting the binary target variable (M/B) to 0/1

In [10]:

```
var =  ['result']

# Defining the map function
def binary_map(x):
    return x.map({'M': 1, "B": 0})

# Applying the function to the housing list
data[var] = data[var].apply(binary_map)
```

In [11]:

```
data.head()
```

Out[11]:

	id	result	radius1	texture1	perimeter1	area1	smooth1	compact1	concave1	num_con
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	C
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	C
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	C
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	C
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	C

5 rows × 32 columns

All variables as numeric.

Discussing Outliers

Outliers in medical data can be indicative of abnormal values of a certain metric which can result in disease. It may happen, that if outliers are removed, most of the columns indicating disease can be removed. This makes the dataset unreliable for training a model. Therefore, neither with outlier analysis be performed nor will any outliers, if they exist, will be removed.

Checking for Missing Values

In [12]:

```
# Adding up the missing values (column-wise)
data.isnull().sum()
```

Out[12]:

```
id              0
result          0
radius1         0
texture1        0
perimeter1      0
area1           0
smooth1         0
compact1        0
concave1        0
num_concave_1   0
symmetry1       0
fractal1        0
radius2         0
texture2        0
perimeter2      0
area2           0
smooth2         0
compact2        0
concave2        0
num_concave_2   0
symmetry2       0
fractal2        0
radius3         0
texture3        0
perimeter3      0
area3           0
smooth3         0
compact3        0
concave3        0
num_concave_3   0
symmetry3       0
fractal3        0
dtype: int64
```

There are no missing values in this dataset.

Test-Train Split

In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

```
# Putting feature variable to X
X = data.drop(['result', 'id'], axis=1)

X.head()
```

Out[14]:

	radius1	texture1	perimeter1	area1	smooth1	compact1	concave1	num_concave_1	symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259

	radius1	texture1	perimeter1	area1	smooth1	compact1	concave1	num_concave_1	symmetry
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 30 columns

In [15]:

```
# Putting response variable to y
y = data['result']

y.head()
```

Out[15]:

```
0    1
1    1
2    1
3    1
4    1
Name: result, dtype: int64
```

In [16]:

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=
```

Feature Scaling

In [17]:

```
from sklearn.preprocessing import StandardScaler
```

In [18]:

```
scaler = StandardScaler()

X_train[['radius1', 'texture1', 'perimeter1', 'area1', 'smooth1', 'compact1', 'conca

X_train.head()
```

Out[18]:

	radius1	texture1	perimeter1	area1	smooth1	compact1	concave1	num_concave_1	s
18	1.637784	0.690583	1.596428	1.760808	0.159736	-0.008936	0.818487	1.223840	
213	0.959747	1.489559	0.956704	0.865550	0.325707	0.224893	1.087267	0.465394	
532	-0.101282	-0.673064	-0.146924	-0.203307	-0.241784	-0.601366	-0.907797	-0.767831	
191	-0.359447	0.517199	-0.383828	-0.398715	-0.624460	-0.729873	-0.727727	-0.510572	
235	-0.001988	0.479710	-0.063141	-0.123251	-0.391810	-0.662279	-0.946194	-0.763648	

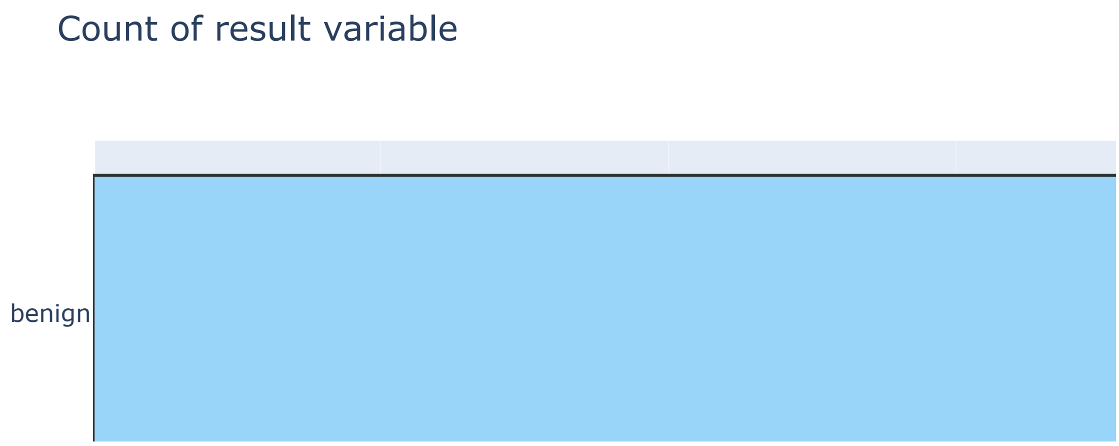
5 rows × 30 columns

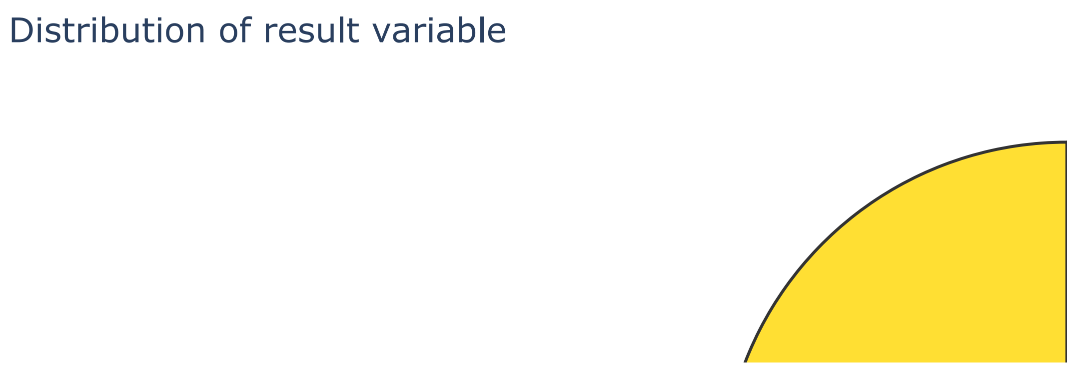
Checking the Cancer Rate

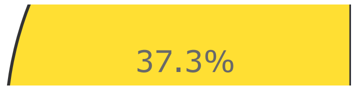
In [19]:

```
import plotly.graph_objs as go
import plotly.offline as py
M = data[(data['result'] != 0)]
B = data[(data['result'] == 0)]
#COUNT
trace = go.Bar(x = (len(M), len(B)), y = ['malignant', 'benign'], orientation = 'h',
        color=[ 'gold', 'lightskyblue'],
        line=dict(color='#000000',width=1.5)))
```

```
layout = dict(title =  'Count of result variable')

fig = dict(data = [trace], layout=layout)
py.iplot(fig)

#PERCENTAGE
trace = go.Pie(labels = ['benign','malignant'], values = data['result'].value_counts
               textfont=dict(size=15), opacity = 0.8,
               marker=dict(colors=['lightskyblue', 'gold'],
                           line=dict(color='#000000', width=1.5)))


layout = dict(title =  'Distribution of result variable')

fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```








```
In [20]: cancer = (sum(data['result'])/len(data['result'].index))*100
         cancer
```

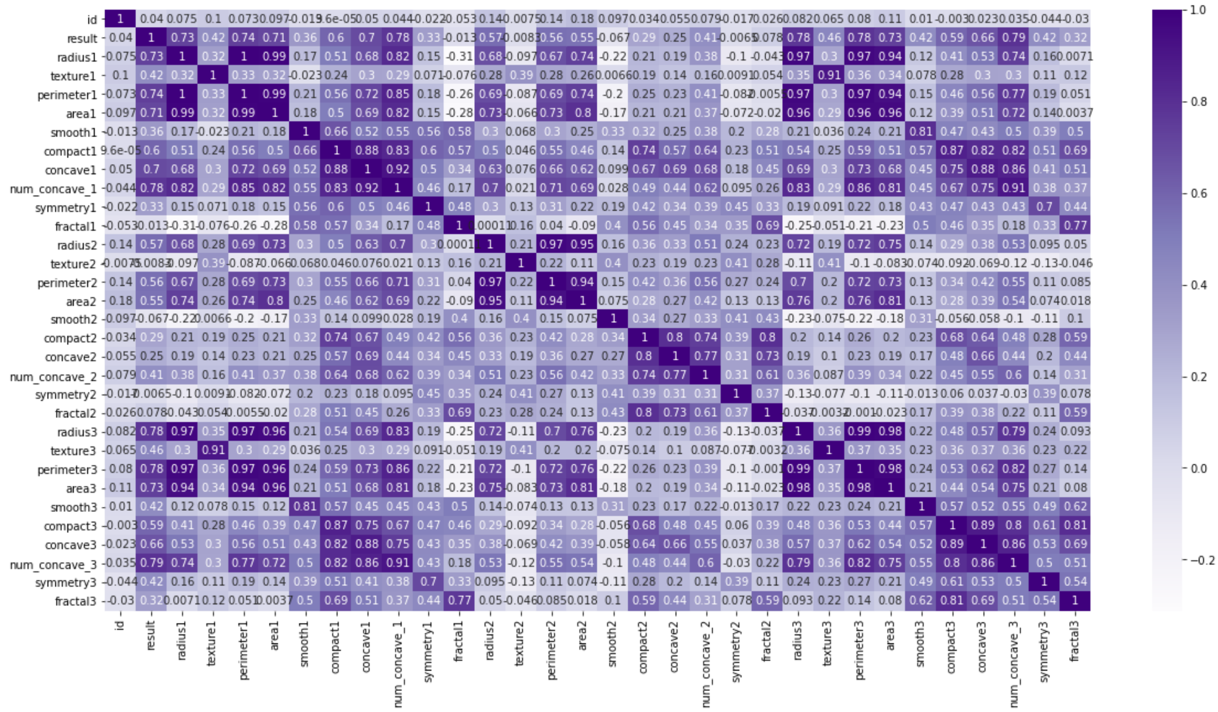
```
Out[20]: 37.258347978910365
```

About 37% of tumours are maligant.

Looking at Correlations

```
In [21]: # Importing matplotlib and seaborn
         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
```

```
In [22]: # Let's see the correlation matrix
         plt.figure(figsize = (20,10)) # Size of the figure
         sns.heatmap(data.corr(),annot = True, cmap = "Purples")
         plt.show()
```



Dropping highly correlated variables

In [23]:

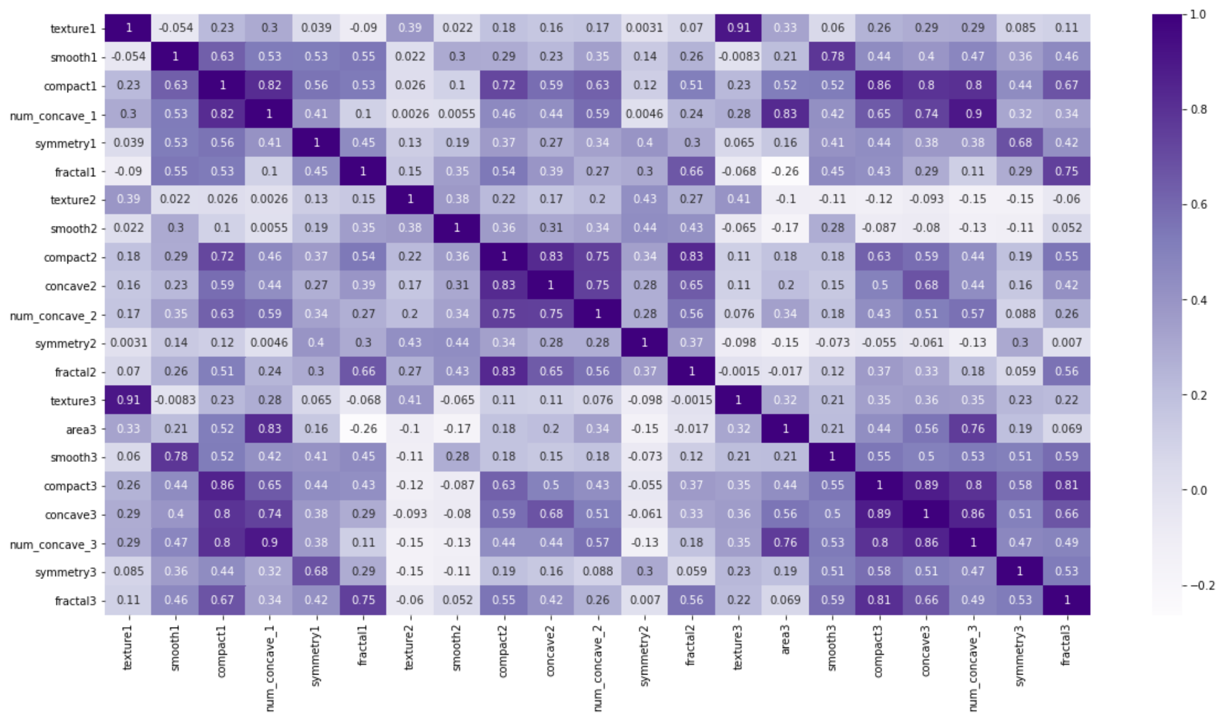
```
X_test = X_test.drop(['radius1','perimeter1','area1','concave1','radius2','perimeter
X_train = X_train.drop(['radius1','perimeter1','area1','concave1','radius2','perimet
```

Checking the Correlation Matrix

After dropping highly correlated variables now let's check the correlation matrix again.

In [24]:

```
plt.figure(figsize = (20,10))
sns.heatmap(X_train.corr(),annot = True, cmap = "Purples")
plt.show()
```



Model Building

Running The First Training Model

In [25]:
```
import statsmodels.api as sm
```

In [26]:
```
# Logistic regression model
logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Binomial())
logm1.fit().summary()
```

Out[26]:

Generalized Linear Model Regression Results

Dep. Variable:	result	**No. Observations:**	398
Model:	GLM	**Df Residuals:**	376
Model Family:	Binomial	**Df Model:**	21
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-16.433
Date:	Tue, 25 Apr 2023	**Deviance:**	32.866
Time:	18:33:36	**Pearson chi2:**	1.15e+03
No. Iterations:	11		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.7721	0.948	-0.815	0.415	-2.630	1.086
texture1	1.2470	2.115	0.590	0.555	-2.898	5.393
smooth1	1.1497	2.452	0.469	0.639	-3.656	5.955
compact1	-3.8941	3.686	-1.056	0.291	-11.119	3.331
num_concave_1	5.2556	4.997	1.052	0.293	-4.538	15.049
symmetry1	0.6702	1.461	0.459	0.646	-2.192	3.533
fractal1	0.1628	2.187	0.074	0.941	-4.124	4.450
texture2	-0.9352	1.577	-0.593	0.553	-4.027	2.156
smooth2	1.7629	1.419	1.242	0.214	-1.018	4.544
compact2	-3.3574	5.038	-0.666	0.505	-13.232	6.518
concave2	-1.9897	2.788	-0.714	0.475	-7.454	3.474
num_concave_2	6.6346	3.035	2.186	0.029	0.687	12.583
symmetry2	1.5761	1.705	0.924	0.355	-1.766	4.919
fractal2	-7.5251	4.686	-1.606	0.108	-16.709	1.659
texture3	1.9740	2.654	0.744	0.457	-3.228	7.177
area3	13.1890	4.226	3.121	0.002	4.906	21.472
smooth3	0.2684	2.153	0.125	0.901	-3.952	4.489
compact3	-0.2651	5.915	-0.045	0.964	-11.858	11.327
concave3	5.5042	4.760	1.156	0.248	-3.826	14.834
num_concave_3	-4.7294	4.434	-1.067	0.286	-13.419	3.961

symmetry3	-1.4026	1.869	-0.751	0.453	-5.066	2.260
fractal3	7.0952	4.057	1.749	0.080	-0.856	15.046

Feature Selection Using RFE

In [27]:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [28]:

```
from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15) # running RFE with 15 variables as output
rfe = rfe.fit(X_train, y_train)
```

In [29]:

```
rfe.support_
```

Out[29]:

```
array([ True,  True, False,  True, False,  True,  True, False,  True,
       False,  True, False,  True,  True,  True,  True, False,  True,
        True,  True,  True])
```

In [30]:

```
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

Out[30]:

```
[('texture1', True, 1),
 ('smooth1', True, 1),
 ('compact1', False, 2),
 ('num_concave_1', True, 1),
 ('symmetry1', False, 6),
 ('fractal1', True, 1),
 ('texture2', True, 1),
 ('smooth2', False, 4),
 ('compact2', True, 1),
 ('concave2', False, 3),
 ('num_concave_2', True, 1),
 ('symmetry2', False, 7),
 ('fractal2', True, 1),
 ('texture3', True, 1),
 ('area3', True, 1),
 ('smooth3', True, 1),
 ('compact3', False, 5),
 ('concave3', True, 1),
 ('num_concave_3', True, 1),
 ('symmetry3', True, 1),
 ('fractal3', True, 1)]
```

In [31]:

```
col = X_train.columns[rfe.support_]
```

In [32]:

```
X_train.columns[~rfe.support_]
```

Out[32]:

```
Index(['compact1', 'symmetry1', 'smooth2', 'concave2', 'symmetry2',
       'compact3'],
      dtype='object')
```

Assessing the model with StatsModels

In [33]:

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[33]: Generalized Linear Model Regression Results

Dep. Variable:	result	**No. Observations:**	398
Model:	GLM	**Df Residuals:**	382
Model Family:	Binomial	**Df Model:**	15
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-18.790
Date:	Tue, 25 Apr 2023	**Deviance:**	37.580
Time:	18:33:46	**Pearson chi2:**	1.25e+03
No. Iterations:	11		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-0.8242	0.749	-1.101	0.271	-2.292	0.643
texture1	2.5183	1.839	1.369	0.171	-1.086	6.123
smooth1	-0.1784	1.865	-0.096	0.924	-3.833	3.476
num_concave_1	3.6463	3.012	1.211	0.226	-2.258	9.550
fractal1	-0.1142	1.718	-0.066	0.947	-3.482	3.254
texture2	-0.0706	1.180	-0.060	0.952	-2.383	2.242
compact2	-6.2141	2.760	-2.252	0.024	-11.623	-0.805
num_concave_2	5.7435	2.179	2.636	0.008	1.473	10.015
fractal2	-2.1568	2.103	-1.026	0.305	-6.279	1.965
texture3	0.7313	2.214	0.330	0.741	-3.607	5.070
area3	12.3712	3.998	3.094	0.002	4.535	20.208
smooth3	2.0162	1.318	1.530	0.126	-0.567	4.599
concave3	2.5319	1.374	1.843	0.065	-0.161	5.225
num_concave_3	-3.3955	3.352	-1.013	0.311	-9.966	3.175
symmetry3	0.4739	0.508	0.933	0.351	-0.522	1.470
fractal3	3.6212	2.050	1.767	0.077	-0.397	7.639

In [34]:

```
# Getting the predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

Out[34]:

```
18     1.000000e+00
213    8.003465e-01
532    3.153827e-04
191    3.545547e-04
235    7.258893e-03
471    2.071169e-03
485    4.307244e-06
153    9.872773e-08
357    2.162015e-05
412    1.465136e-09
dtype: float64
```

In [35]:
```
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

Out[35]:
```
array([1.00000000e+00, 8.00346500e-01, 3.15382745e-04, 3.54554665e-04,
       7.25889335e-03, 2.07116917e-03, 4.30724397e-06, 9.87277283e-08,
       2.16201453e-05, 1.46513613e-09])
```

Creating a dataframe with the actual cancer flag and the predicted probabilities

In [36]:
```
y_train_pred_final = pd.DataFrame({'Cancer':y_train.values, 'Cancer_Prob':y_train_pr
y_train_pred_final['id'] = y_train.index
y_train_pred_final.head()
```

Out[36]:

	Cancer	Cancer_Prob	id
0	1	1.000000	18
1	1	0.800346	213
2	0	0.000315	532
3	0	0.000355	191
4	0	0.007259	235

Creating new column 'predicted' with 1 if Cancer_Prob > 0.5 else 0

In [37]:
```
y_train_pred_final['predicted'] = y_train_pred_final.Cancer_Prob.map(lambda x: 1 if

# Let's see the head
y_train_pred_final.head()
```

Out[37]:

	Cancer	Cancer_Prob	id	predicted
0	1	1.000000	18	1
1	1	0.800346	213	1
2	0	0.000315	532	0
3	0	0.000355	191	0
4	0	0.007259	235	0

In [38]:
```
from sklearn import metrics
```

In [39]:
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.Cancer, y_train_pred_final.p
print(confusion)
```

```
[[254   1]
 [  4 139]]
```

In [40]:
```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.Cancer, y_train_pred_final.predicted
```

```
0.9874371859296482
```

Checking VIFs

In [41]:

```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [42]:

```
# Create a dataframe that will contain the names of all the feature variables and th
vif = pd.DataFrame()
vif['Features'] = X_train[col].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_tra
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[42]:

	Features	VIF
12	num_concave_3	21.63
2	num_concave_1	16.86
8	texture3	13.03
0	texture1	9.10
14	fractal3	8.77
5	compact2	7.63
11	concave3	7.10
3	fractal1	6.56
9	area3	6.42
7	fractal2	6.17
1	smooth1	5.94
10	smooth3	5.06
6	num_concave_2	4.96
4	texture2	2.64
13	symmetry3	1.80

There are a few variables with high VIF. It's best to drop these variables as they aren't helping much with prediction and unnecessarily making the model complex. The variable 'num_concave_3' has the highest VIF. So let's start by dropping that.

In [43]:

```
col = col.drop('num_concave_3', 1)
col
```

Out[43]:

```
Index(['texture1', 'smooth1', 'num_concave_1', 'fractal1', 'texture2',
       'compact2', 'num_concave_2', 'fractal2', 'texture3', 'area3', 'smooth3',
       'concave3', 'symmetry3', 'fractal3'],
      dtype='object')
```

In [44]:

```
# Let's re-run the model using the selected variables
X_train_sm = sm.add_constant(X_train[col])
logm3 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm3.fit()
res.summary()
```

Out[44]:

Generalized Linear Model Regression Results

Dep. Variable:	result	**No. Observations:**	398
Model:	GLM	**Df Residuals:**	383
Model Family:	Binomial	**Df Model:**	14
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-19.339
Date:	Tue, 25 Apr 2023	**Deviance:**	38.679
Time:	18:33:53	**Pearson chi2:**	541.
No. Iterations:	11		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-0.7099	0.727	-0.977	0.328	-2.134	0.714
texture1	3.4577	1.613	2.144	0.032	0.297	6.618
smooth1	-0.2612	1.842	-0.142	0.887	-3.871	3.349
num_concave_1	1.6806	2.113	0.795	0.426	-2.460	5.822
fractal1	0.6025	1.531	0.393	0.694	-2.399	3.604
texture2	0.8002	0.857	0.933	0.351	-0.880	2.481
compact2	-5.7923	2.660	-2.178	0.029	-11.005	-0.579
num_concave_2	4.4123	1.531	2.882	0.004	1.411	7.413
fractal2	-1.2898	1.945	-0.663	0.507	-5.103	2.523
texture3	-0.7149	1.714	-0.417	0.677	-4.074	2.644
area3	12.2684	3.936	3.117	0.002	4.553	19.983
smooth3	1.6469	1.272	1.295	0.195	-0.846	4.140
concave3	2.1867	1.356	1.613	0.107	-0.470	4.844
symmetry3	0.3696	0.504	0.733	0.463	-0.618	1.357
fractal3	2.3489	1.473	1.595	0.111	-0.538	5.236

In [45]:
```
y_train_pred = res.predict(X_train_sm).values.reshape(-1)
```

In [46]:
```
y_train_pred[:10]
```

Out[46]:
```
array([1.00000000e+00, 7.68146122e-01, 6.78769390e-04, 1.00377150e-03,
       7.49152262e-03, 1.55095773e-02, 1.04061793e-05, 8.31563680e-08,
       1.45862069e-05, 1.52242702e-09])
```

In [47]:
```
y_train_pred_final['Cancer_Prob'] = y_train_pred
```

In [48]:
```
# Creating new column 'predicted' with 1 if Cancer_Prob > 0.5 else 0
y_train_pred_final['predicted'] = y_train_pred_final.Cancer_Prob.map(lambda x: 1 if
y_train_pred_final.head()
```

Out[48]:

	Cancer	Cancer_Prob	id	predicted
0	1	1.000000	18	1
1	1	0.768146	213	1
2	0	0.000679	532	0
3	0	0.001004	191	0
4	0	0.007492	235	0

In [49]:

```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.Cancer, y_train_pred_final.predicted
```

```
0.9849246231155779
```

So overall the accuracy hasn't dropped much.

Let's check the VIFs again

In [50]:

```
vif = pd.DataFrame()
vif['Features'] = X_train[col].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_tra
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[50]:

	Features	VIF
8	texture3	12.18
2	num_concave_1	10.73
0	texture1	8.69
13	fractal3	7.83
5	compact2	7.56
11	concave3	6.48
3	fractal1	6.36
9	area3	6.35
7	fractal2	5.88
1	smooth1	5.83
10	smooth3	4.85
6	num_concave_2	3.42
4	texture2	2.32
12	symmetry3	1.76

In [51]:

```
# Let's drop TotalCharges since it has a high VIF
col = col.drop('texture3')
col
```

Out[51]:

```
Index(['texture1', 'smooth1', 'num_concave_1', 'fractal1', 'texture2',
       'compact2', 'num_concave_2', 'fractal2', 'area3', 'smooth3', 'concave3',
```

```
       'symmetry3', 'fractal3'],
      dtype='object')
```

In [52]:

```
# Let's re-run the model using the selected variables
X_train_sm = sm.add_constant(X_train[col])
logm4 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm4.fit()
res.summary()
```

Out[52]:

Generalized Linear Model Regression Results

Dep. Variable:	result	**No. Observations:**	398
Model:	GLM	**Df Residuals:**	384
Model Family:	Binomial	**Df Model:**	13
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-19.428
Date:	Tue, 25 Apr 2023	**Deviance:**	38.856
Time:	18:33:58	**Pearson chi2:**	535.
No. Iterations:	11		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-0.8510	0.659	-1.290	0.197	-2.144	0.442
texture1	2.9298	0.928	3.158	0.002	1.112	4.748
smooth1	0.1848	1.480	0.125	0.901	-2.716	3.086
num_concave_1	1.5068	2.015	0.748	0.455	-2.442	5.456
fractal1	0.5031	1.525	0.330	0.742	-2.487	3.493
texture2	0.5561	0.619	0.898	0.369	-0.658	1.770
compact2	-5.7000	2.568	-2.219	0.026	-10.734	-0.666
num_concave_2	4.4944	1.497	3.003	0.003	1.561	7.428
fractal2	-1.2837	1.880	-0.683	0.495	-4.968	2.400
area3	12.1154	3.916	3.094	0.002	4.440	19.791
smooth3	1.3598	1.050	1.295	0.195	-0.699	3.418
concave3	2.0797	1.320	1.575	0.115	-0.508	4.667
symmetry3	0.3050	0.481	0.635	0.526	-0.637	1.247
fractal3	2.3557	1.489	1.582	0.114	-0.562	5.274

In [53]:

```
y_train_pred = res.predict(X_train_sm).values.reshape(-1)
```

In [54]:

```
y_train_pred[:10]
```

Out[54]:

```
array([1.00000000e+00, 8.03254206e-01, 6.30537191e-04, 7.70102736e-04,
       9.64772842e-03, 9.98123172e-03, 1.13446287e-05, 7.98630370e-08,
       2.03777625e-05, 1.44696103e-09])
```

In [55]:
```
y_train_pred_final['Cancer_Prob'] = y_train_pred
```

In [56]:
```
# Creating new column 'predicted' with 1 if Cancer_Prob > 0.5 else 0
y_train_pred_final['predicted'] = y_train_pred_final.Cancer_Prob.map(lambda x: 1 if
y_train_pred_final.head()
```

Out[56]:

	Cancer	Cancer_Prob	id	predicted
0	1	1.000000	18	1
1	1	0.803254	213	1
2	0	0.000631	532	0
3	0	0.000770	191	0
4	0	0.009648	235	0

In [57]:
```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.Cancer, y_train_pred_final.predicted
```

```
0.9849246231155779
```

The accuracy is still practically the same.

Let's now check the VIFs again

In [58]:
```
vif = pd.DataFrame()
vif['Features'] = X_train[col].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_tra
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[58]:

	Features	VIF
2	num_concave_1	10.70
5	compact2	7.52
12	fractal3	7.21
10	concave3	6.37
8	area3	6.34
3	fractal1	6.03
7	fractal2	5.83
1	smooth1	5.64
9	smooth3	4.37
6	num_concave_2	3.39
11	symmetry3	1.72
0	texture1	1.66
4	texture2	1.64

All variables have a good value of VIF. So we need not drop any more variables and we can proceed with making predictions using this model only

In [59]:
```
# Let's take a look at the confusion matrix again
confusion = metrics.confusion_matrix(y_train_pred_final.Cancer, y_train_pred_final.p
confusion
```

Out[59]:
```
array([[253,   2],
       [  4, 139]], dtype=int64)
```

In [60]:
```
# Let's check the overall accuracy.
metrics.accuracy_score(y_train_pred_final.Cancer, y_train_pred_final.predicted)
```

Out[60]:
```
0.9849246231155779
```

Metrics beyond simply accuracy

In [61]:
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [62]:
```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

Out[62]:
```
0.972027972027972
```

In [63]:
```
# Let us calculate specificity
TN / float(TN+FP)
```

Out[63]:
```
0.9921568627450981
```

In [64]:
```
# Calculate false postive rate
print(FP/ float(TN+FP))
```

```
0.00784313725490196
```

In [65]:
```
# positive predictive value
print (TP / float(TP+FP))
```

```
0.9858156028368794
```

In [66]:
```
# Negative predictive value
print (TN / float(TN+ FN))
```

```
0.9844357976653697
```

Plotting the ROC Curve

An ROC curve demonstrates several things:

- It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).

- The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
- The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

In [67]:

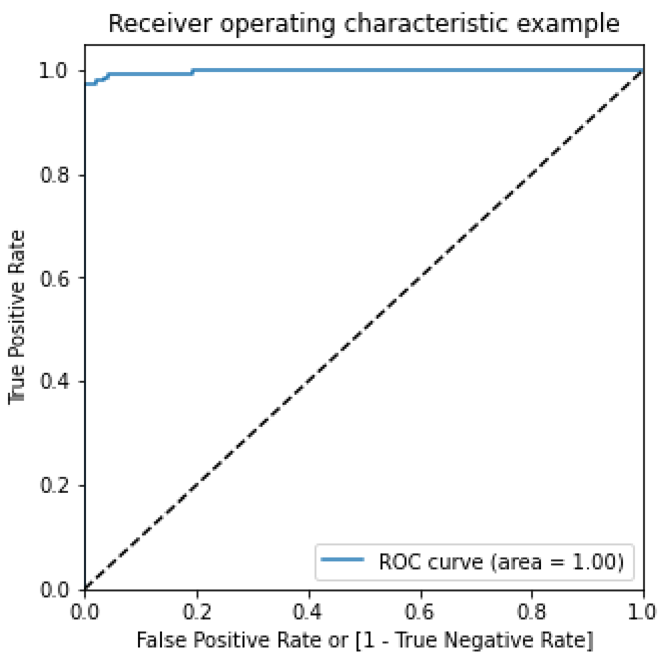
```
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None
```

In [68]:

```
fpr, tpr, thresholds = metrics.roc_curve( y_train_pred_final.Cancer, y_train_pred_fi
```

In [69]:

```
draw_roc(y_train_pred_final.Cancer, y_train_pred_final.Cancer_Prob)
```



Finding Optimal Cutoff Point

Optimal cutoff probability is that probability where we get balanced sensitivity and specificity.

In [70]:

```
# Let's create columns with different probability cutoffs
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    y_train_pred_final[i]= y_train_pred_final.Cancer_Prob.map(lambda x: 1 if x > i e
y_train_pred_final.head()
```

Out[70]:

	Cancer	Cancer_Prob	id	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	1	1.000000	18	1	1	1	1	1	1	1	1	1	1	1
1	1	0.803254	213	1	1	1	1	1	1	1	1	1	1	0
2	0	0.000631	532	0	1	0	0	0	0	0	0	0	0	0
3	0	0.000770	191	0	1	0	0	0	0	0	0	0	0	0
4	0	0.009648	235	0	1	0	0	0	0	0	0	0	0	0

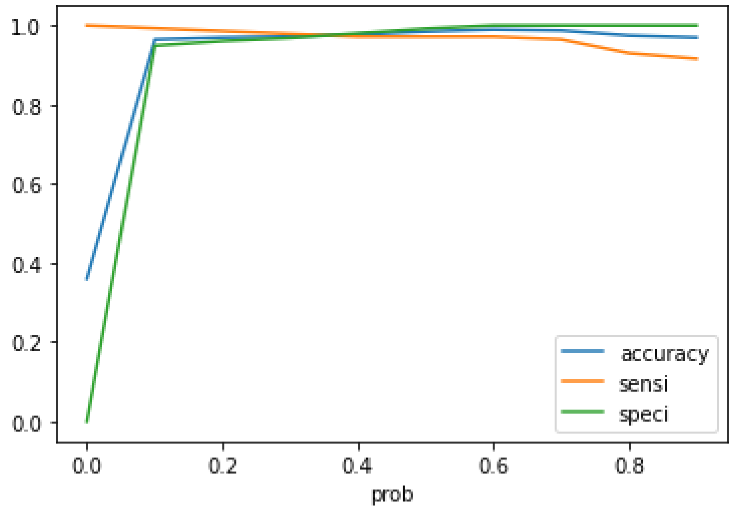
In [71]:

```
# Now let's calculate accuracy sensitivity and specificity for various probability c
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
from sklearn.metrics import confusion_matrix

# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = metrics.confusion_matrix(y_train_pred_final.Cancer, y_train_pred_final[i]
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

```
     prob  accuracy     sensi     speci
0.0   0.0  0.359296  1.000000  0.000000
0.1   0.1  0.964824  0.993007  0.949020
0.2   0.2  0.969849  0.986014  0.960784
0.3   0.3  0.972362  0.979021  0.968627
0.4   0.4  0.977387  0.972028  0.980392
0.5   0.5  0.984925  0.972028  0.992157
0.6   0.6  0.989950  0.972028  1.000000
0.7   0.7  0.987437  0.965035  1.000000
0.8   0.8  0.974874  0.930070  1.000000
0.9   0.9  0.969849  0.916084  1.000000
```

In [72]:

```
# Let's plot accuracy sensitivity and specificity for various probabilities.
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



From the curve above, 0.3 is the optimum point to take it as a cutoff probability.

In [73]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final.Cancer_Prob.map( lambda x

y_train_pred_final.head()
```

Out[73]:

	Cancer	Cancer_Prob	id	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	final_predic
0	1	1.000000	18	1	1	1	1	1	1	1	1	1	1	1	
1	1	0.803254	213	1	1	1	1	1	1	1	1	1	1	0	
2	0	0.000631	532	0	1	0	0	0	0	0	0	0	0	0	
3	0	0.000770	191	0	1	0	0	0	0	0	0	0	0	0	
4	0	0.009648	235	0	1	0	0	0	0	0	0	0	0	0	

In [74]:
```
# Let's check the overall accuracy.
metrics.accuracy_score(y_train_pred_final.Cancer, y_train_pred_final.final_predicted
```

Out[74]: 0.9723618090452262

In [75]:
```
confusion2 = metrics.confusion_matrix(y_train_pred_final.Cancer, y_train_pred_final.
confusion2
```

Out[75]:
```
array([[247,   8],
       [  3, 140]], dtype=int64)
```

In [76]:
```
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

In [77]:
```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

Out[77]: 0.9790209790209791

In [78]:
```
# Let us calculate specificity
TN / float(TN+FP)
```
Out[78]:
```
0.9686274509803922
```
In [79]:
```
# Calculate false postive rate
print(FP/ float(TN+FP))
```
```
0.03137254901960784
```
In [80]:
```
# Positive predictive value
print (TP / float(TP+FP))
```
```
0.9459459459459459
```
In [81]:
```
# Negative predictive value
print (TN / float(TN+ FN))
```
```
0.988
```
Precision and Recall

In [82]:
```
#Looking at the confusion matrix again
```
In [83]:
```
confusion = metrics.confusion_matrix(y_train_pred_final.Cancer, y_train_pred_final.p
confusion
```
Out[83]:
```
array([[253,   2],
       [  4, 139]], dtype=int64)
```
Precision

TP / TP + FP

In [84]:
```
confusion[1,1]/(confusion[0,1]+confusion[1,1])
```
Out[84]:
```
0.9858156028368794
```
Recall

TP / TP + FN

In [85]:
```
confusion[1,1]/(confusion[1,0]+confusion[1,1])
```
Out[85]:
```
0.972027972027972
```
In [86]:
```
from sklearn.metrics import precision_score, recall_score
```
In [87]:
```
precision_score(y_train_pred_final.Cancer, y_train_pred_final.predicted)
```
Out[87]:
```
0.9858156028368794
```

In [88]:
```
recall_score(y_train_pred_final.Cancer, y_train_pred_final.predicted)
```
Out[88]:
```
0.972027972027972
```

Precision and recall tradeoff

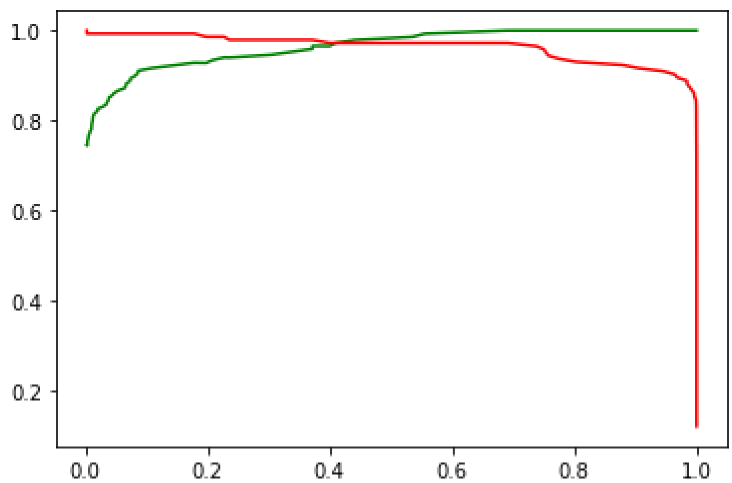
In [89]:
```
from sklearn.metrics import precision_recall_curve
```

In [90]:
```
y_train_pred_final.Cancer, y_train_pred_final.predicted
```

Out[90]:
```
(0      1
 1      1
 2      0
 3      0
 4      0
       ..
 393    0
 394    1
 395    0
 396    0
 397    0
 Name: Cancer, Length: 398, dtype: int64,
 0      1
 1      1
 2      0
 3      0
 4      0
       ..
 393    0
 394    1
 395    0
 396    0
 397    0
 Name: predicted, Length: 398, dtype: int64)
```

In [91]:
```
p, r, thresholds = precision_recall_curve(y_train_pred_final.Cancer, y_train_pred_fi
```

In [92]:
```
plt.plot(thresholds, p[:-1], "g-")
plt.plot(thresholds, r[:-1], "r-")
plt.show()
```



Making predictions on the test set

In [93]:
```
X_test.head()
```

Out[93]:

	texture1	smooth1	compact1	num_concave_1	symmetry1	fractal1	texture2	smooth2	comp
400	21.02	0.12300	0.25760	0.11980	0.2113	0.07115	0.7747	0.007159	0.0:
225	13.47	0.09906	0.07624	0.04603	0.2075	0.05448	0.8121	0.007089	0.0'
321	19.66	0.08020	0.08564	0.07726	0.1928	0.05096	0.6863	0.004536	0.0'
173	14.71	0.10060	0.05743	0.02583	0.1566	0.06669	1.8050	0.014960	0.0:
506	20.04	0.10960	0.11520	0.02166	0.2124	0.06894	0.7959	0.006272	0.0:

5 rows × 21 columns

In [94]:
```
X_test[['texture1', 'smooth1', 'compact1', 'num_concave_1', 'symmetry1', 'fractal1',
```

In [95]:
```
X_test = X_test[col]
X_test.head()
```

Out[95]:

	texture1	smooth1	num_concave_1	fractal1	texture2	compact2	num_concave_2	fractal
400	0.350574	1.782986	1.732833	1.093800	-0.835957	0.627078	-0.171565	0.49316
225	-1.381898	0.144290	-0.116177	-1.102576	-0.761750	-0.588809	0.194369	-0.79220
321	0.038499	-1.146680	0.666588	-1.566358	-1.011356	-0.616419	0.133639	-0.74766
173	-1.097360	0.249703	-0.622480	0.506167	1.208310	-0.220857	0.656847	0.38216
506	0.125697	0.865754	-0.726999	0.802619	-0.793893	-0.179974	-0.267486	0.03854

In [96]:
```
X_test_sm = sm.add_constant(X_test)
```

Making predictions on the test set.

In [97]:
```
y_test_pred = res.predict(X_test_sm)
```

In [98]:
```
y_test_pred[:10]
```

Out[98]:
```
400    1.000000e+00
225    1.933357e-03
321    9.999974e-01
173    1.681099e-07
506    1.883079e-04
380    1.036066e-04
197    1.378656e-03
260    1.000000e+00
40     2.223837e-03
160    9.907566e-05
dtype: float64
```

In [99]:

```
# Converting y_pred to a dataframe which is an array
y_pred_1 = pd.DataFrame(y_test_pred)
```

In [100…

```
# Let's see the head
y_pred_1.head()
```

Out[100…

	0
400	1.000000e+00
225	1.933357e-03
321	9.999974e-01
173	1.681099e-07
506	1.883079e-04

In [101…

```
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
```

In [102…

```
# Putting CustID to index
y_test_df['id'] = y_test_df.index
```

In [103…

```
# Removing index for both dataframes to append them side by side
y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

In [104…

```
# Appending y_test_df and y_pred_1
y_pred_final = pd.concat([y_test_df, y_pred_1],axis=1)
```

In [105…

```
y_pred_final.head()
```

Out[105…

	result	id	0
0	1	400	1.000000e+00
1	0	225	1.933357e-03
2	1	321	9.999974e-01
3	0	173	1.681099e-07
4	0	506	1.883079e-04

In [106…

```
# Renaming the column
y_pred_final= y_pred_final.rename(columns={ 0 : 'Cancer_Prob'})
```

In [107…

```
# Rearranging the columns
y_pred_final = y_pred_final.reindex(['id','result','Cancer_Prob'], axis=1)
```

In [108…

```python
# Let's see the head of y_pred_final
y_pred_final.head()
```

Out[108…

	id	result	Cancer_Prob
0	400	1	1.000000e+00
1	225	0	1.933357e-03
2	321	1	9.999974e-01
3	173	0	1.681099e-07
4	506	0	1.883079e-04

In [109…

```python
y_pred_final['final_predicted'] = y_pred_final.Cancer_Prob.map(lambda x: 1 if x > 0.
```

In [110…

```python
y_pred_final.head()
```

Out[110…

	id	result	Cancer_Prob	final_predicted
0	400	1	1.000000e+00	1
1	225	0	1.933357e-03	0
2	321	1	9.999974e-01	1
3	173	0	1.681099e-07	0
4	506	0	1.883079e-04	0

In [111…

```python
# Let's check the overall accuracy.
metrics.accuracy_score(y_pred_final.result, y_pred_final.final_predicted)
```

Out[111…

```
0.9532163742690059
```

In [112…

```python
confusion2 = metrics.confusion_matrix(y_pred_final.result, y_pred_final.final_predic
confusion2
```

Out[112…

```
array([[100,   2],
       [  6,  63]], dtype=int64)
```

In [113…

```python
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

In [114…

```python
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

Out[114…

```
0.9130434782608695
```

In [115…

```python
# Let us calculate specificity
TN / float(TN+FP)
```

```
Out[115…   0.9803921568627451
```