Breast Cancer Analysis and Prediction

A Project Report

Submitted by

Sara Dharadhar

Rhea Gupta

Vanshika Gupta

Anya Jain

Under the Guidance of

Prof. Ameyaa Biwalkar

in partial fulfillment for the award of the degree of

B.TECH

COMPUTER ENGINEERING

At

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT ENGINEERING,NMIMS, MUMBAI

MARCH , 2020

DECLARATION

We, Sara Dharadhar, Rhea Gupta, Vanshika Gupta, Anya Jain, Roll No. B024,B031,B032,B036 B Tech (Computer Engineering), VI semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.

2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)

3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)

4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Students: _____, _____, _____, _____

Names: Sara Dharadhar , Rhea Gupta , Vanshika Gupta , Anya Jain

Roll Nos. : B024, B031, B032, B036

Place: Mumbai

Date:

CERTIFICATE

This is to certify that the project entitled "__Breast Cancer Analysis and Prediction__" is the bonafide work carried out by _Sara Dharadhar,Rhea Gupta,Vanshika Gupta, Anya Jain_ of B Tech, MPSTME (NMIMS), Mumbai, during the VI semester of the academic year__2019-2020_, in partial fulfillment of the requirements for the Course Programming Language.

_____

Prof. Ameyaa Biwalkar

Internal Mentor

_____ _____

Examiner 1 Examiner 2

Table of contents

1.INTRODUCTION

- Vector Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women , and it is the second leading cause of cancer death among women.
- Breast Cancer occurs as a results of abnormal growth of cells in the breast tissue, commonly referred to as a Tumor.
- A tumor does not mean cancer - tumors can be benign (not cancerous), pre-malignant (pre-cancerous), or malignant (cancerous).
- Tests such as MRI, mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer.
- This is an analysis of the Breast Cancer Wisconsin (Diagnostic) DataSet, obtained from Kaggle.
- This data set was created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin,USA.
- To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. T
- The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, than it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated

1.1 Main Objective

This analysis aims to observe which features are most helpful in predicting malignant or benign cancer cells. To observe general trends in the data set and applying a machine learning model to the dataset.

1.2 Attribute Information

1)ID number
2) Diagnosis (M = malignant, B = benign) 3-32)
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 mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

2. TECHNICAL CONTENT

We used Jupyter Notebook for our Data Analysis Project. Some important libraries that we used for our project are as follows:

Pandas
For Indexing, manipulating, renaming, sorting, merging data frames. To Update, Add, Delete columns from a data frame, To Impute missing files, handle missing data or NANs and to Plot data with histogram or box plot

NumPy
For basic and advanced array operations, To work with DateTime or Linear Algebraand for basic Slicing and Advanced Indexing in NumPy Python

SciPy
SciPy library for modules for efficient mathematical routines as linear algebra, interpolation, optimization, integration, and statistics.

Matplotlib
For visualizations such as: Line plots, Scatter plots, Area plots, Bar charts and Histograms and Pie charts. Matplotlib also facilitates labels, grids, legends, and some more formatting entities.

Seaborn
For correlation, and regression models.

Scikit Learn
For Machine learning. Regression and pre processing of data.

Plotly
For Graphs such as: Basic Charts: Line, Pie, Scatter, Bubble, Dot, Gantt, Sunburst, Treemap, Sankey, Filled Area Charts AND Statistical and Seaborn Styles: Error, Box, Histograms, Facet and Trellis Plots, Tree plots, Violin Plots, Trend Lines.

Itertools
For functions that create iterators for efficient looping. This module implements a number of iterator building blocks inspired by constructs from APL, Haskell, and SML. Each has been recast in a form suitable for Python.

Warnings
Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn’t warrant raising an exception and terminating the program. This module was used for Error Handling.

3. METHODS IMPLEMENTED

1) Preparing dataset and dropping missing value columns

- Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import itertools
from itertools import chain
from sklearn.feature_selection import RFE
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score, learning_curve,
train_test_split
from sklearn.metrics import precision_score, recall_score, confusion_matrix, roc_cu
rve, precision_recall_curve, accuracy_score
import warnings
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff
warnings.filterwarnings('ignore')
```

- Reading Dataset

```
data = pd.read_csv('data.csv')
```

- Plotting Graph

```
empty = pd.DataFrame(len(data['id']) - data.isnull().sum(), columns = ['Count'])
trace = go.Bar(x = empty.index, y = empty['Count'])
layout = dict(title =  "Missing Values")
fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```

Fig1 shows the result.

2) Exploratory data analysis

- Plotting bar chart for count of diagnosis

```
M = data[(data['diagnosis'] != 0)]
B = data[(data['diagnosis'] == 0)]
In [26]:
trace = go.Bar(x = (len(M), len(B)), y = ['malignant', 'benign'], orientation = 'h', ma
rker=dict(
        color=[ 'gold', 'lightskyblue'],
        line=dict(color='black',width=1.5)))

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

Fig2.a. shows the result.

- Pie chart for count percentage

```
fig = dict(data = [trace], layout=layout)
py.iplot(fig)
trace = go.Pie(labels = ['benign','malignant'], values = data['diagnosis'].value_co
unts(),
               textfont=dict(size=15),
               marker=dict(colors=['lightskyblue', 'gold'],
                           line=dict(color='black', width=1.5)))


layout = dict(title =  'Distribution of diagnosis variable')
```

Fig2.b. shows the result.

- In the following histogram we have taken features at random and in the following analysis we will re cognise the important features.

```
fig = dict(data = [trace], layout=layout)
py.iplot(fig)
def plot_distribution(data_select, size_bin) :
    tmp1 = M[data_select]
    tmp2 = B[data_select]
    hist_data = [tmp1, tmp2]

    group_labels = ['malignant', 'benign']
    colors = ['gold', 'skyblue']

    fig = ff.create_distplot(hist_data, group_labels, colors = colors, show_hist = True
, bin_size = size_bin, curve_type='kde')

    fig['layout'].update(title = data_select)

    py.iplot(fig, filename = 'Density plot')
In [32]:
plot_distribution('radius_mean', .5)
plot_distribution('texture_mean', .5)
plot_distribution('perimeter_mean', 5)
plot_distribution('area_mean', 10)
```

Fig2.c,2.d,2.e, show the results.

- Heatmap

```
correlation = data.corr()
matrix_cols = correlation.columns.tolist()
corr_array  = np.array(correlation)
In [31]:
trace = go.Heatmap(z = corr_array,
                   x = matrix_cols,
                   y = matrix_cols,
                   xgap = 2,
                   ygap = 2,
                   colorscale='Viridis',
                   colorbar   = dict() ,
```

```
                    )
layout = go.Layout(dict(title = 'Correlation Matrix for variables',
                        autosize = False,
                        height  = 720,
                        width   = 800,
                        margin  = dict(r = 0 ,l = 210,
                                       t = 25,b = 210,
                                    ),
                        yaxis   = dict(tickfont = dict(size = 9)),
                        xaxis   = dict(tickfont = dict(size = 9)),
                       )
                  )
fig = go.Figure(data = [trace],layout = layout)
py.iplot(fig)
```

Fig2.f. shows the result.

3) Correlation

- Positively correlated features

```
palette ={0 : 'lightblue', 1 : 'gold'}
edgecolor = 'grey'


fig = plt.figure(figsize=(12,12))

plt.subplot(221)
ax1 = sns.scatterplot(x = data['perimeter_mean'], y = data['radius_worst'], hue = "diag
nosis",
                      data = data, palette = palette, edgecolor=edgecolor)
plt.title('perimeter mean vs radius worst')
plt.subplot(222)
ax2 = sns.scatterplot(x = data['area_mean'], y = data['radius_worst'], hue = "diagnosis
",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('area mean vs radius worst')
plt.subplot(223)
ax3 = sns.scatterplot(x = data['texture_mean'], y = data['texture_worst'], hue = "diagn
osis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('texture mean vs texture worst')
plt.subplot(224)
ax4 = sns.scatterplot(x = data['area_worst'], y = data['radius_worst'], hue = "diagnosi
s",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('area mean vs radius worst')

fig.suptitle('Positive correlated features', fontsize = 20)
plt.savefig('1')
plt.show()
```

Fig3.a. shows the result.

- Uncorrelated features

```
fig = plt.figure(figsize=(12,12))

plt.subplot(221)
ax1 = sns.scatterplot(x = data['smoothness_mean'], y = data['texture_mean'], hue = "dia
gnosis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('smoothness mean vs texture mean')
plt.subplot(222)
ax2 = sns.scatterplot(x = data['radius_mean'], y = data['fractal_dimension_worst'], hue
= "diagnosis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('radius mean vs fractal dimension_worst')
plt.subplot(223)
ax3 = sns.scatterplot(x = data['texture_mean'], y = data['symmetry_mean'], hue = "diagn
osis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('texture mean vs symmetry mean')
plt.subplot(224)
ax4 = sns.scatterplot(x = data['texture_mean'], y = data['symmetry_se'], hue = "diagnos
is",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('texture mean vs symmetry se')

fig.suptitle('Uncorrelated features', fontsize = 20)
plt.savefig('2')
plt.show()
```

Fig3.b. shows the result.

- Negatively correlated features

```
fig = plt.figure(figsize=(12,12))

plt.subplot(221)
ax1 = sns.scatterplot(x = data['area_mean'], y = data['fractal_dimension_mean'], hu
e = "diagnosis",
                       data = data, palette =palette, edgecolor=edgecolor)
plt.title('smoothness mean vs fractal dimension mean')
plt.subplot(222)
ax2 = sns.scatterplot(x = data['radius_mean'], y = data['fractal_dimension_mean'],
hue = "diagnosis",
                       data = data, palette =palette, edgecolor=edgecolor)
plt.title('radius mean vs fractal dimension mean')
plt.subplot(223)
ax2 = sns.scatterplot(x = data['area_mean'], y = data['smoothness_se'], hue = "diag
nosis",
                       data = data, palette =palette, edgecolor=edgecolor)
plt.title('area mean vs fractal smoothness se')
plt.subplot(224)
```

```
ax2 = sns.scatterplot(x = data['smoothness_se'], y = data['perimeter_mean'], hue =
"diagnosis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('smoothness se vs perimeter mean')

fig.suptitle('Negative correlated features', fontsize = 20)
plt.savefig('3')
plt.show()
```

Fig3.c. shows the result.

- Correlation between important features

```
palette ={0 : 'lightblue', 1 : 'gold'}
edgecolor = 'grey'
fig = plt.figure(figsize=(12,12))

plt.subplot(221)
ax1 = sns.scatterplot(x = data['perimeter_worst'], y = data['radius_worst'], hue =
"diagnosis",
                      data = data, palette = palette, edgecolor=edgecolor)
plt.title('perimeter worst vs radius worst')
plt.subplot(222)
ax2 = sns.scatterplot(x = data['concave points_worst'], y = data['radius_worst'], h
ue = "diagnosis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('concave points worst vs radius worst')
plt.subplot(223)
ax3 = sns.scatterplot(x = data['area_worst'], y = data['perimeter_worst'], hue = "d
iagnosis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('area worst vs perimeter worst')
plt.subplot(224)
ax4 = sns.scatterplot(x = data['area_worst'], y = data['radius_worst'], hue = "diag
nosis",
                      data = data, palette =palette, edgecolor=edgecolor)
plt.title('area worst vs radius worst')

fig.suptitle('Correlation', fontsize = 20)
plt.savefig('5')
plt.show()
```

Fig3.d. shows the result.

- Heatmap for radius_worst,perimeter_worst,area_worst,concave points_worst

```
features = ['radius_worst','perimeter_worst','area_worst','concave points_worst']
plt.figure(figsize=(20,20))
heat = sns.heatmap(data[features].corr(), vmax=1, square=True, annot=True)
```

Fig3.e. shows the result.

4) Logistic regression model

- Defining functions

```
def plot_confusion_matrix(cm, classes,
                          normalize = False,
                          title = 'Confusion matrix"',
                          cmap = plt.cm.Blues) :
    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation = 0)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])) :
        plt.text(j, i, cm[i, j],
                 horizontalalignment = 'center',
                 color = 'white' if cm[i, j] > thresh else 'black')

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Show metrics
def show_metrics():
    tp = cm[1,1]
    fn = cm[1,0]
    fp = cm[0,1]
    tn = cm[0,0]
    print('Accuracy  =     {:.3f}'.format((tp+tn)/(tp+tn+fp+fn)))
    print('Precision =     {:.3f}'.format(tp/(tp+fp)))
    print('Recall    =     {:.3f}'.format(tp/(tp+fn)))
    print('F1_score  =     {:.3f}'.format(2*(((tp/(tp+fp))*(tp/(tp+fn)))/
                                           ((tp/(tp+fp))+(tp/(tp+fn))))))
```

The precision-recall curve shows the tradeoff between precision and recall for different threshold

```
In [62]:
def plot_precision_recall():
    plt.step(recall, precision, color = 'b', alpha = 0.2,
             where = 'post')
    plt.fill_between(recall, precision, step ='post', alpha = 0.2,
                     color = 'b')

    plt.plot(recall, precision, linewidth=2)
    plt.xlim([0.0,1])
    plt.ylim([0.0,1.05])
    plt.xlabel('Recall')
    plt.ylabel('Precision')
```

```
    plt.title('Precision Recall Curve')
    plt.show();
```

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

In [63]:

```
def plot_roc():
    plt.plot(fpr, tpr, label = 'ROC curve', linewidth = 2)
    plt.plot([0,1],[0,1], 'k--', linewidth = 2)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.show();
```

The Learning curve determines cross-validated training and test scores.

In [64]:

```
def plot_learning_curve(estimator, title, X, y, ylim = None, cv = None,
                        n_jobs = 1, train_sizes = np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel('Training examples')
    plt.ylabel('Score')
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv = cv, n_jobs = n_jobs, train_sizes = train_sizes)
    train_scores_mean = np.mean(train_scores, axis = 1)
    train_scores_std = np.std(train_scores, axis = 1)
    test_scores_mean = np.mean(test_scores, axis = 1)
    test_scores_std = np.std(test_scores, axis = 1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha = 0.1, color = "g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color = "r",
             label = "Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color = "g",
             label = "Cross-validation score")
    plt.legend(loc = "best")
    return plt
```

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In [65]:

```
def cross_val_metrics(model) :
    scores = ['accuracy', 'precision', 'recall']
    for sc in scores:
        scores = cross_val_score(model, X, y, cv = 5, scoring = sc)
        print('[%s] : %0.5f (+/- %0.5f)'%(sc, scores.mean(), scores.std()))
```

In [67]:

```
y = np.array(data.diagnosis.tolist())
data = data.drop('diagnosis', 1)
X = np.array(data.as_matrix())
```

```
In [68]:
scaler = StandardScaler()
X = scaler.fit_transform(X)
In [69]:
random_state = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.12, random_stat
e = random_state)
```

- Logistic regression model

  Shows confusion matrix and ROC curve for all features.

```
log_clf = LogisticRegression(random_state = random_state)
param_grid = {
             'penalty' : ['l2','l1'],
             'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
             }

CV_log_clf = GridSearchCV(estimator = log_clf, param_grid = param_grid , scoring =
'accuracy', verbose = 1, n_jobs = -1)
CV_log_clf.fit(X_train, y_train)

best_parameters = CV_log_clf.best_params_
print('The best parameters for using this model is', best_parameters)
#Log with best hyperparameters
CV_log_clf = LogisticRegression(C = best_parameters['C'],
                                penalty = best_parameters['penalty'],
                                random_state = random_state)

CV_log_clf.fit(X_train, y_train)
y_pred = CV_log_clf.predict(X_test)
y_score = CV_log_clf.decision_function(X_test)

# Confusion maxtrix & metrics
cm = confusion_matrix(y_test, y_pred)
class_names = [0,1]
plt.figure()
plot_confusion_matrix(cm,
                      classes=class_names,
                      title='Logistic Confusion matrix')
plt.savefig('6')
plt.show()

show_metrics()

# ROC curve
fpr, tpr, t = roc_curve(y_test, y_score)
plot_roc()
```

Fig4.a. shows the result.

Shows confusion matrix and ROC curve with recursive feature elimation.

```
#Logistic regression with RFE
log_clf = LogisticRegression(C = best_parameters['C'],
                                penalty = best_parameters['penalty'],
```

```
                                    random_state = random_state)

selector = RFE(log_clf)
selector = selector.fit(X_train, y_train)

y_pred = selector.predict(X_test)
y_score = selector.predict_proba(X_test)[:,1]


# Confusion maxtrix & metrics
cm = confusion_matrix(y_test, y_pred)
class_names = [0,1]
plt.figure()
plot_confusion_matrix(cm,
                      classes=class_names,
                      title='Logistic Confusion matrix')
plt.show()

show_metrics()

# ROC curve
fpr, tpr, t = roc_curve(y_test, y_score)
plot_roc()
```

Fig4.b. shows the result.

Learning curve for all features.

```
plot_learning_curve(CV_log_clf, 'Learning Curve For Logistic Model', X, y, (0.85,1.
05), 10)
plt.savefig('7')
plt.show()
```

Fig4.c. shows the result.

Learning curve with recursive feature elimination

```
plot_learning_curve(selector, 'Learning Curve For Logistic Model with RFE', X, y, (
0.85,1.05), 10)
plt.show()
```

Fig4.d. shows the result.

Confusion matrix for recall 100% at different threshold values

```
# Threshold
thresholds_adj = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]

plt.figure(figsize = (15,15))

j = 1
for i in thresholds_adj:
    y_score = CV_log_clf.predict_proba(X_test)[:,1] > i


    plt.subplot(3,3,j)
    j += 1

    cm = confusion_matrix(y_test, y_score)

    tp = cm[1,1]
```

```
fn = cm[1,0]
fp = cm[0,1]
tn = cm[0,0]

print('Recall w/ threshold = %s :'%i, (tp/(tp+fn)))

class_names = [0,1]
plot_confusion_matrix(cm,
                      classes=class_names,
                      title='Threshold = %s'%i)
```

Fig4.e. shows the result.

4. SCREENSHOTS

1.Preparing dataset and dropping missing value columns

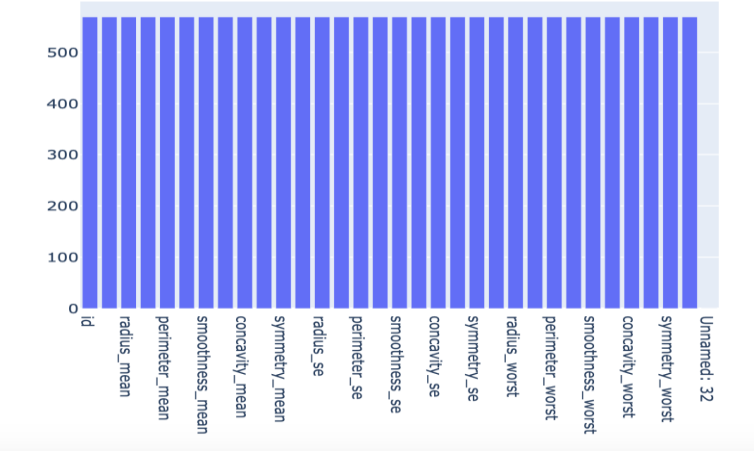


Fig 1. Shows dropped column

In the above figure the unnamed column with no values has been dropped .

2.Exploratory data analysis

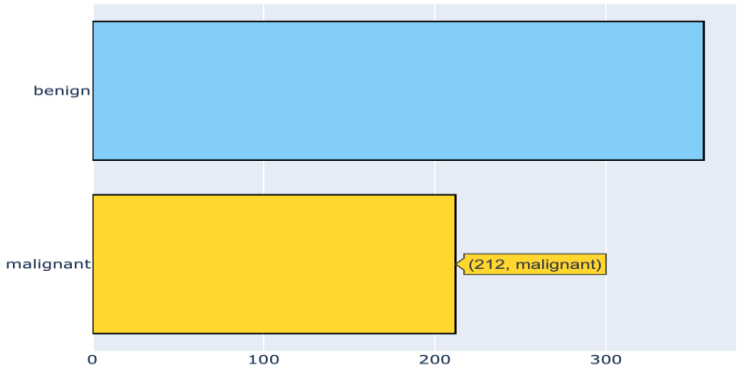


Fig 2.a. Shows the count of the diagnosis column via a bar chart

The above figure shows the count of total malignant and benign cells in our dataset , we can conclude that our dataset has more benign examples.

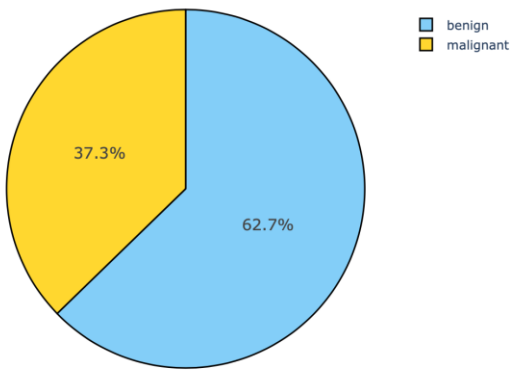


Fig 2.b. Shows the count percentage of the diagnosis column via a pie chart

The above figure shows the count percentage of total malignant and benign cells in our dataset , we can conclude that our dataset has more benign examples

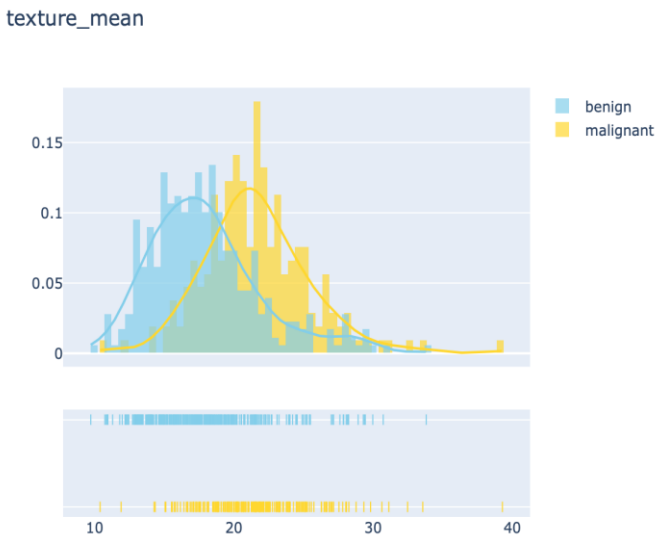


Fig 2.c. Shows the frequency of texture_mean as it increases and how increase in texture_mean indicates more malignant cells. Shows KDE curve.

We can conclude from the above data that as texture mean increases the cells become more malignant.The Histogram also shows us the frequency of cells at different texture mean values.

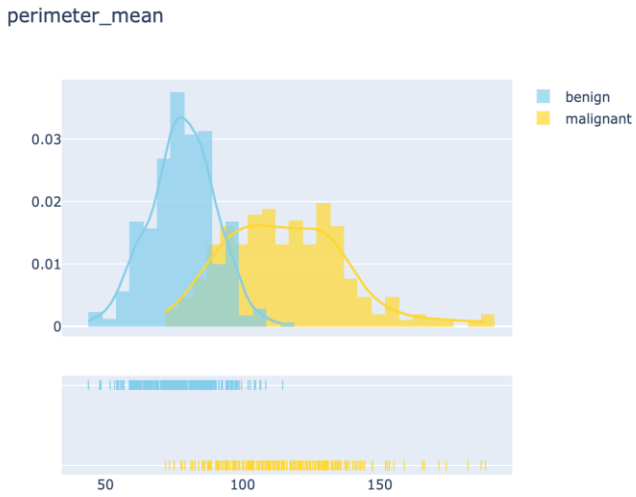


Fig 2.d. Shows the frequency of perimeter_mean as it increases and how increase in perimeter_mean indicates more malignant cells. Shows KDE curve.

We can conclude from the above data that as perimeter mean increases the cells become more malignant.The Histogram also shows us the frequency of cells at different perimeter mean values.

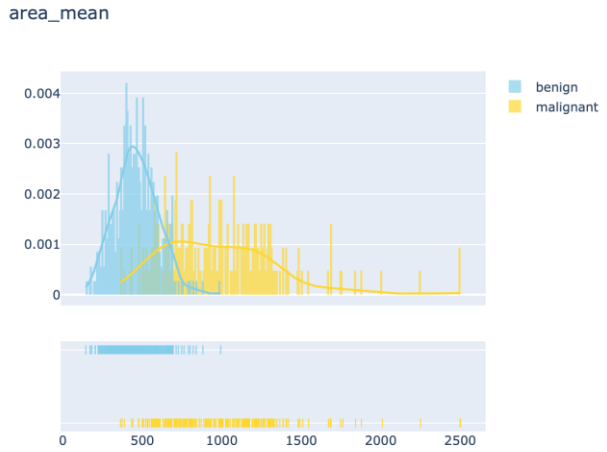


Fig 2.e. Shows the frequency of area_mean as it increases and how increase in area_mean indicates more malignant cells. Shows KDE curve.

We can conclude from the above data that as area mean increases the cells become more malignant.The Histogram also shows us the frequency of cells at different area mean values.

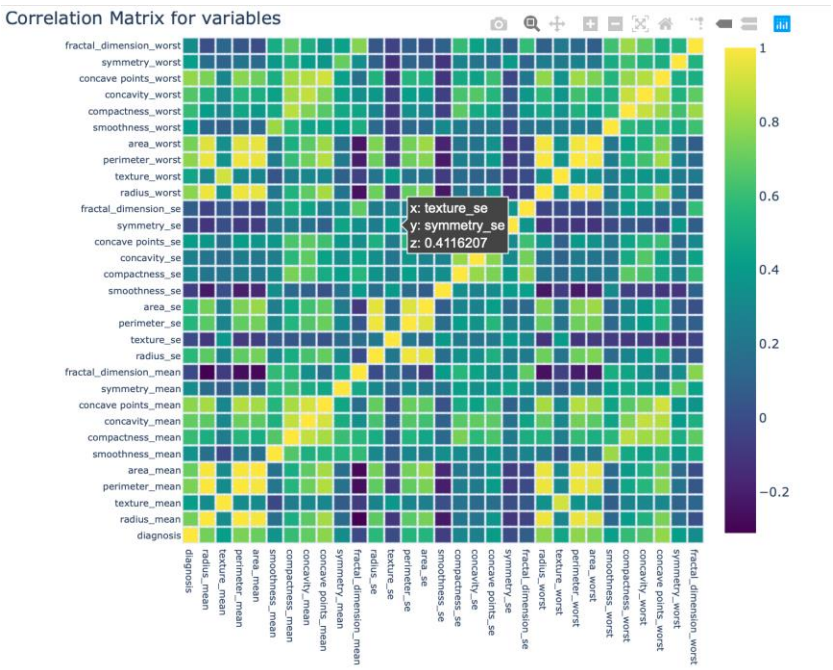


Fig 2.f. Shows the correlation matrix(heat map) for all the features given in our data set.

The heatmap varies from different colours , where yellow indicates positive correlation between the features and it decreases to blue which

indicates negative correlation between the features. .

3.Correlation

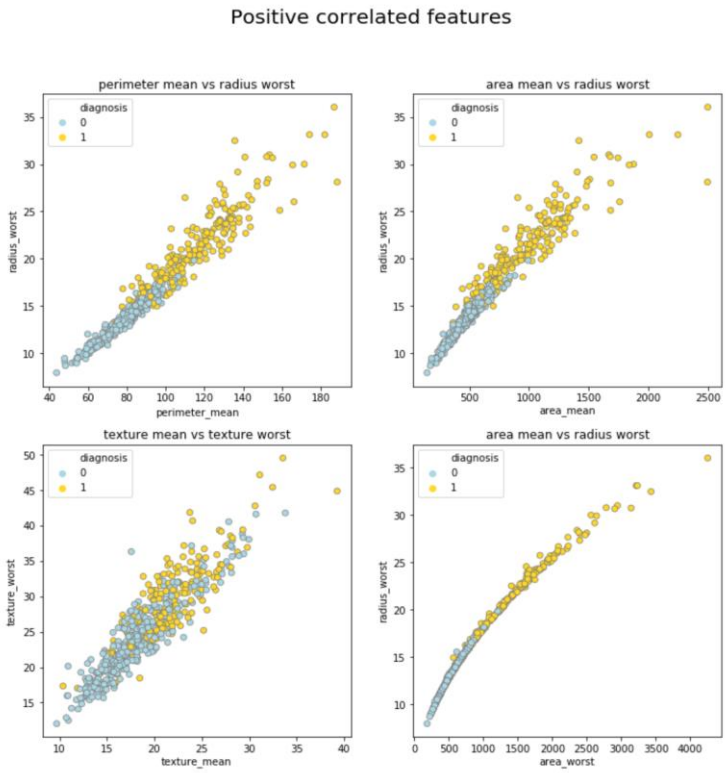


Fig3.a.Shows a scatter plot for the positively correlated features .

The above figure shows how perimeter mean vs radius worst, area mean vs radius worst ,texture mean vs texture worst, area mean vs area worst are positive correlated in the scatter plot as they have a positive slope.

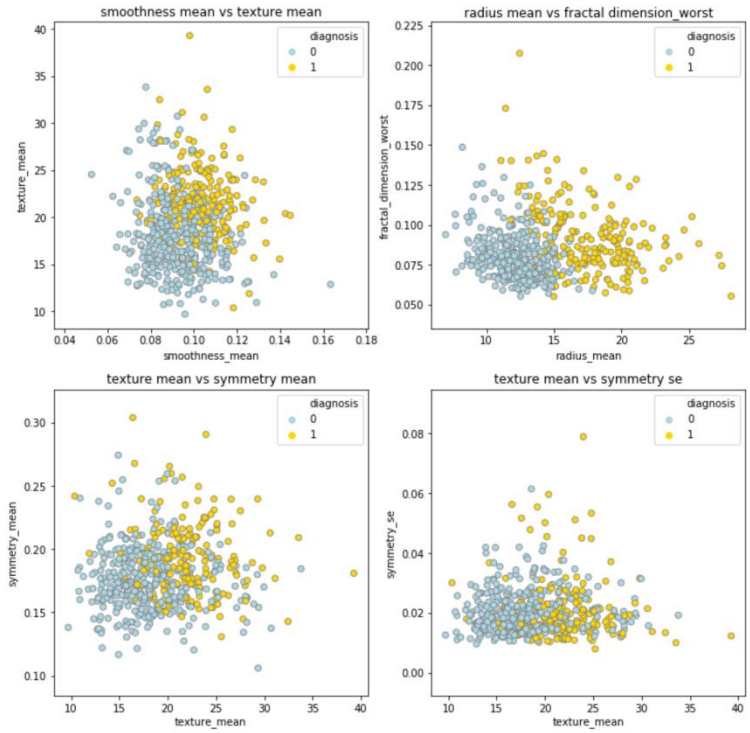


Fig3.b.Shows a scatter plot for the uncorrelated features .

The above figure shows how smoothness mean vs texture mean ,radius mean vs dimension worst, texture mean vs symmetry mean, texture. mean vs symmetry se are uncorrelated in the scatter plot as they have a no slope.

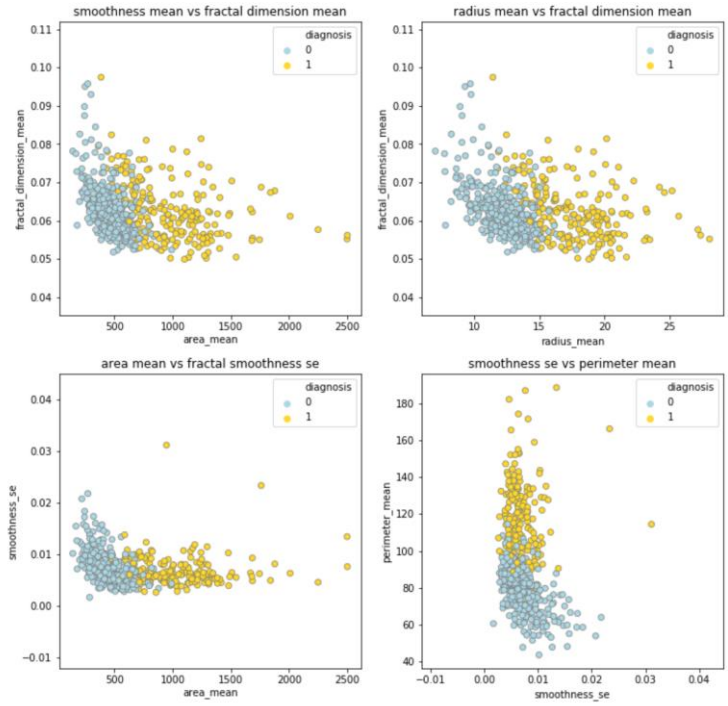


Fig3.c.Shows the scatter plot negative correlated features .

The above figure shows how smoothness mean vs fractional dim mean, area mean vs fractional smoothness se,radius mean vs fractional dim mean, smoothness se vs perimeter mean are positive correlated in the scatter plot as they have a positive slope.

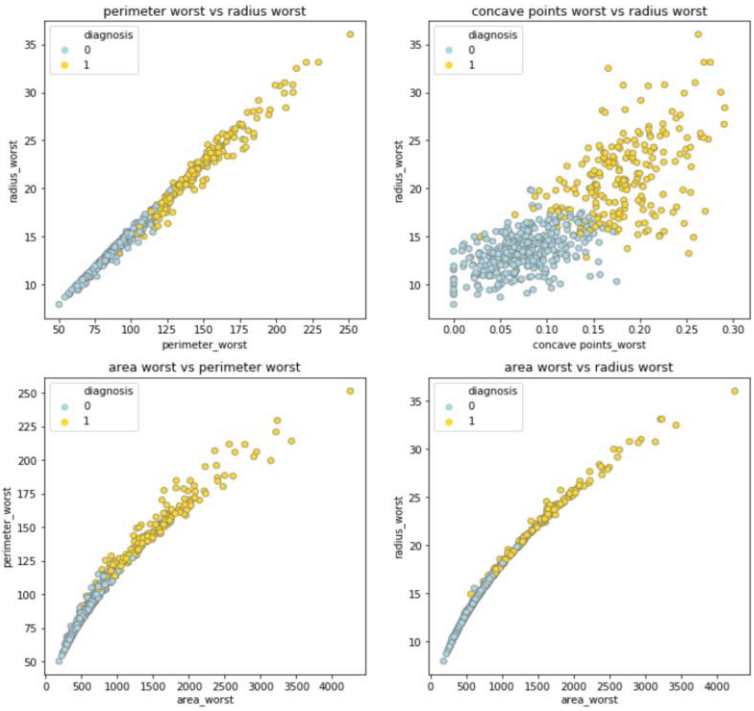


Fig3.d.Shows the scatter plot for the important feature correlation .

On further analysis we found the important features from the dataset and plotted scatter plots to see their correlation.

The above figure shows how perimeter worst vs radius worst, concave point worst vs radius worst ,area worst vs perimeter worst, area worst vs radius worst are positive correlated in the scatter plot as they have a positive slope.

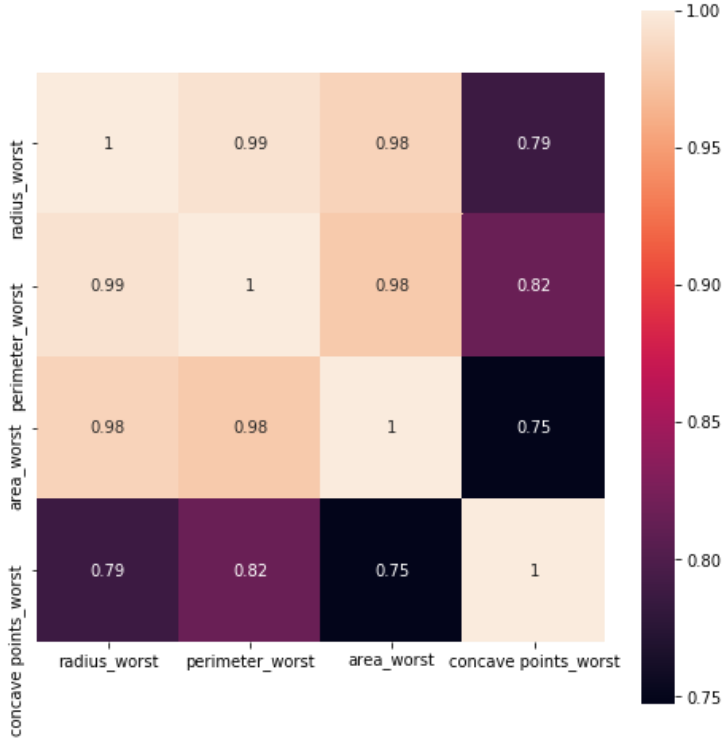


Fig3.e.Shows heatmap for radius_worst,perimeter_worst,area_worst,concave points_worst.

From the above heatmap which is for the important feature correlation we can see that most of them have correlation as its all>0.75 , However maximum correlation is shown by the skin colour which decreases towards purple which shows minimum correlation.

4.Logistic regression model

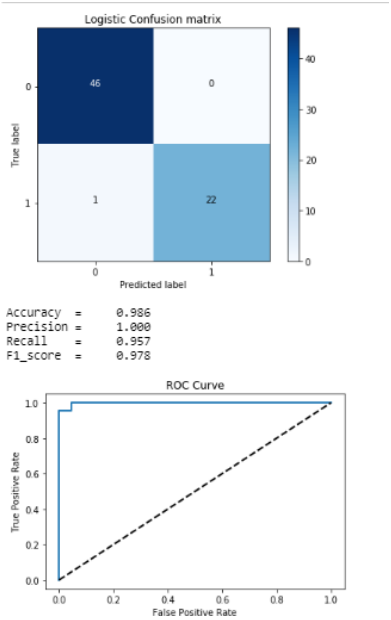


Fig 4.a. Shows confusion matrix and ROC curve for all features.

The above diagram shows the confusion matrix for all features along with the ROC curve .We can see. That the accuracy, precision and recall is close to 1 and hence we can conclude this is a good result.

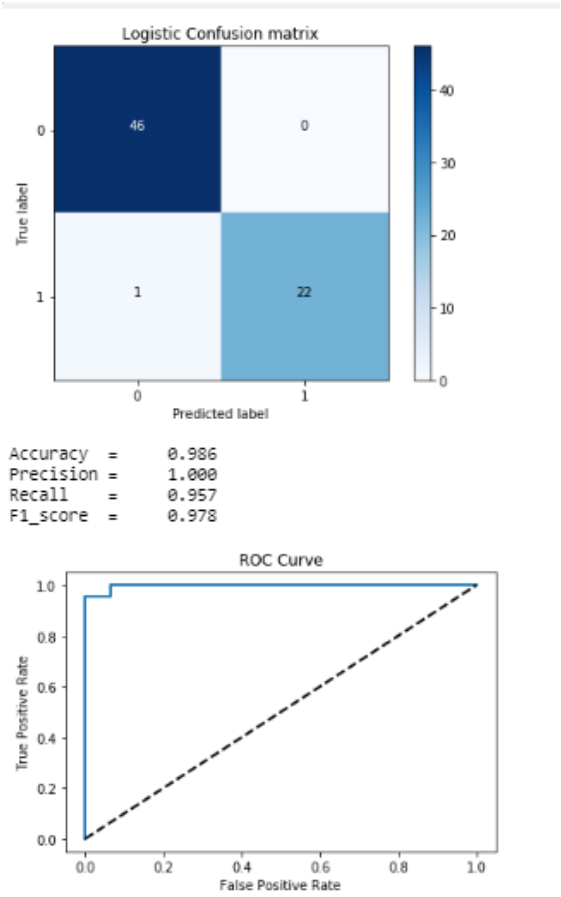


Fig 4.b. Shows confusion matrix and ROC curve for recursive feature elimination.

The above diagram shows the confusion matrix for features after RFE along with the ROC curve .We can see. That the accuracy, precision and recall is close to 1 and hence we can conclude this is a good result.

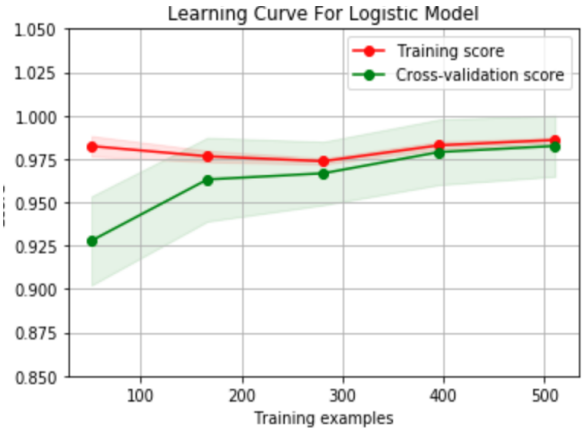


Fig 4.c. Shows Learning curve without RFE

The learning curve in the diagram is showing the training score and cross validation score for different sizes of training set . We can conclude from the above curve that variance is low(bias is high) and our accuracy , precision and recall is also close to 1 which shows that this is a good result .

```
[accuracy] : 0.98242 (+/- 0.00560)
[precision] : 0.99036 (+/- 0.01181)
[recall] : 0.96235 (+/- 0.01131)
[accuracy] : 0.97367 (+/- 0.00778)
[precision] : 0.98094 (+/- 0.01754)
[recall] : 0.94817 (+/- 0.01753)
```

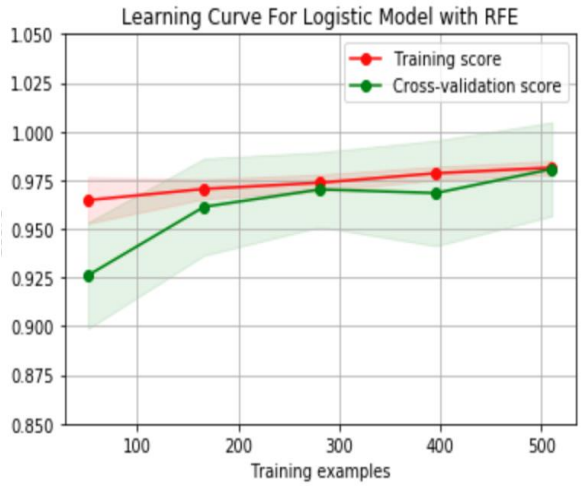
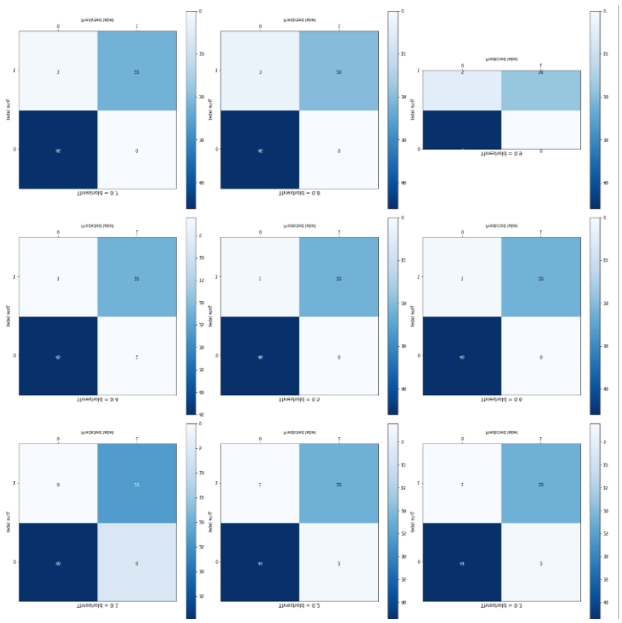


Fig 4.d. Shows Learning curve with RFE

The learning curve in the diagram is showing the training score and cross validation score for different sizes of training set . We can conclude from the above curve that variance is higher comparatively(bias is low) and our accuracy , precision and recall is lesser than the previous learning curve . Hence the learning curve without recursive feature elimination showed us better results .



```
Accuracy  =     0.913
Precision =     0.793
Recall    =     1.000
F1_score  =     0.885
```

Fig 4.e. Shows confusion matrix for 100% recall at different threshold values .

The above figure shows the different results we get when we take recall(sensitivity) as 100% and at different threshold values .

On doing this we see that the precision is decreasing and so we can conclude that using 100% recall and with logistic regression does not give a good result .

5.CONCLUSION & FUTURE SCOPE

- We did an exploratory data analysis which included bar charts ,pie charts, histograms and heat maps to explore the various features visually and to recognise the important features .
- We used Confusion matrix, ROC Curve and learning curves to describe the applied Machine Learning Model.
- We compared the accuracy, precision and recall with all features and also with recursive feature elimination using the logistic regression model.
- We got our values close to 1 which can conclude that Logistic regression worked well.
- We also plotted a Learning Curve which showed that the accuracy, precision and recall without RFE was better as compared to the one with RFE.
- We noticed that if we made recall for the current model 100% the precision fell drastically. For future scope we would go into deeper analysis of the data set and choose to test another model that would give better results with 100% recall i.e a better precision.

6.SOCIETAL APPLICATION

- Data analysis and machine learning can help us by:
  1. Population health management: Predictive algorithms can be applied to identify high-risk cancer patients with a higher chance of readmission after surgery or chemotherapy. Such data can prompt crucial preventive care while reducing costs and strain on a patient
  2. Radiomics: The field of computer-assisted texture analysis uses quantitative data from scans to study tumor characteristics. The patterns could help predict which future patients might benefit.
  3. Pathology: Inaccurate biopsy reads can lead to excessive or inappropriate treatment, the authors note. Artificial intelligence algorithms are offering deep insight on biopsy reads — Google claims its AI tool has 99 percent accuracy in metastatic breast cancer detection — and give oncologists more time to focus on other aspects of care.

- About 1 in 8 U.S. women (about 12%) will develop invasive breast cancer over the course of her lifetime. In 2020, an estimated 276,480 new cases of invasive breast cancer are expected to be diagnosed in women in the U.S., along with 48,530 new cases of non-invasive (in situ) breast cancer. About 85% of breast cancers occur in women who have no family history of breast cancer. These occur due to genetic mutations that happen as a result of the aging process and life in general, rather than inherited mutations.
- Because of all the mentioned above reasons data analysis of cells in women could help in detecting the cancer at early stages or detecting similar trends in breast cancer cells. By doing so the ratio of women being cured from breast cancer or maybe even preventing it could increase considerably.
- Our analysis can be improved by combining various models to keep the recall at 100% and increasing precision which will help researchers find common trends and detect all malignant tumors correctly based on cell data such as the ones given in the dataset.