Spears School of Business, Management Science and Information System

Predicting Breast Cancer Diagnosis Outcomes

MSIS 5223: Programming for Data Science

Andrew Langford – A11672973

Greg McGillin – A20172829

Akanksha Narwani – A20144086

Hieu Nghiem – A20171926

EXECUTIVE SUMMARY

Breast cancer is among the most common forms of cancer and is one of the leading causes of death for women globally. Each year approximately 124 out of 100,000 women are diagnosed with breast cancer, and on average 23 out of the 124 women will die of this disease.

The data set we are analyzing was created by Dr. William H. Wolberg, Nick Street and Olvi L. Mangasarian at the University of Wisconsin. The dataset contains 569 rows and 32 columns and has been frequently used for research, as it is already well suited for analysis without additional modification.

We chose this particular dataset because it presents us with an interesting opportunity to apply data science to a real life problem. All of the data has been taken from real clinical evaluations, and also lends itself well to predictive modeling, with many variables that can be potentially used as predictors for our target variable, the patient's diagnosis.

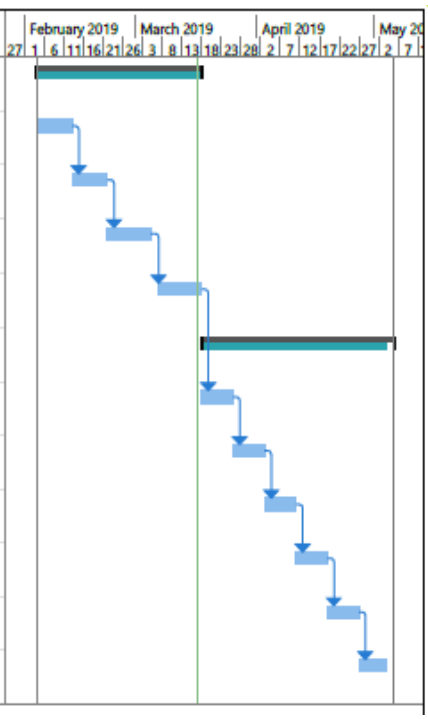
Data science is increasingly being used throughout the medical field, where data driven decision making is helping to improve patient outcomes. Predictive models are one of the many ways data can be leveraged to allow us to make more insightful decisions, and gain a better understanding of the risks patients face.

Through our analysis of the Breast Cancer Diagnosis in Wisconsin data set, we hope our insights from our predictive model can help medical professionals to achieve the following:

- Assist in creating more accurate diagnoses
- Increase awareness of the risks patients face relating to their diagnosis.
- Encourage patient involvement.

PROJECT SCHEDULE

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Deliverable 1**	**31 days**	**Mon 2/4/19**	**Sun 3/17/19**
2		Executive Summary	7 days	Mon 2/4/19	Tue 2/12/19
3		Project Schedule	7 days	Wed 2/13/19	Thu 2/21/19
4		Statement of Scope	7 days	Fri 2/22/19	Mon 3/4/19
5		Data Peparation	7 days	Thu 3/7/19	Sun 3/17/19
6		**Deliverable 2**	**37 days**	**Mon 3/18/19**	**Sun 5/5/19**
7		Adjust Deliverable 1 Requirements	6 days	Mon 3/18/19	Mon 3/25/19
8		Select Modeling Technique	6 days	Tue 3/26/19	Tue 4/2/19
9		Data Splitting & Sub-Samples	6 days	Wed 4/3/19	Wed 4/10/19
10		Build Model	6 days	Thu 4/11/19	Thu 4/18/19
11		Assess Model	6 days	Fri 4/19/19	Fri 4/26/19
12		Formatting, Style, Grammar, Spelling	6 days	Sat 4/27/19	Fri 5/3/19



We have pretty much followed our originally set schedule. There was no change in estimated durations.

STATEMENT OF SCOPE

The data set contains 32 total variables, many of which we will use to predict the eventual malignant or benign category of each tumor suspected of cancer. For our project, we will investigate this clinical data set to examine how each variable relates to a patient's diagnosis, and present our findings on how to better predict a patient's diagnosis based on these variables. Our target variable is 'Diagnosis', which can take on one of two possible values, 'Malignant' (M), or Benign (B). The Predictor variables in our dataset are various measures of the patient's tumor that is suspected of being cancerous.

Our goal for this project is to build different classification models using Python and choose the best model based on various model evaluation matrices. The data set we will be using contains a wide range of information relating to each patient's tumor that is being tested, which we will then use to predict the likelihood of a malignant vs. benign diagnosis.

DATA PREPARATION, DESCRIPTIVE STATISTICS AND DATA DICTIONARY

Data Access

We obtain the dataset from UC Irvine Machine Learning Repository, a reliable site for data sets collection which are used by machine learning community for the analysis of machine learning algorithms.

http://mlr.cs.umass.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data

This dataset was created by Dr. William, Nick Street and Olvi in University of Wisconsin with real input data of breast cancer diagnostic. The dataset contains 10 main characteristics of the cell nuclei present in the image of a fine needle aspirate (FNA):

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)

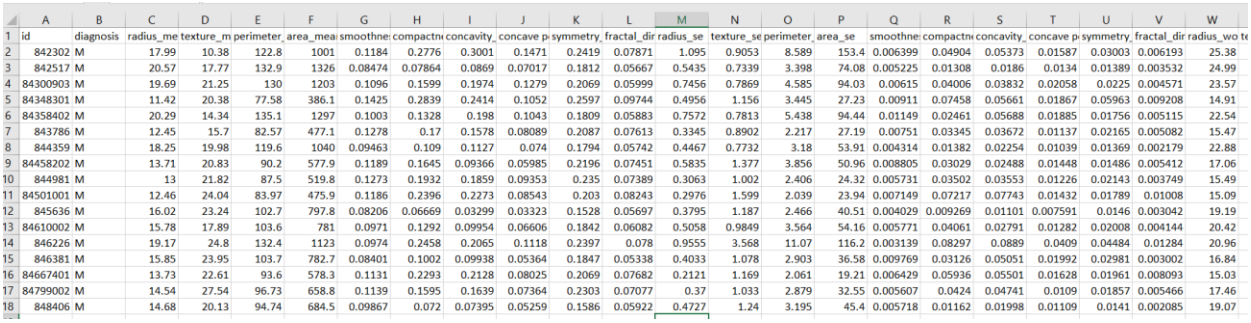
h) concave points (number of concave portions of the contour)

i) symmetry

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

The mean, standard error and "worst" or largest (mean of the three largest values) of these characteristics were computed for the nucleus in each image, resulting in 3*10 = 30 columns. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

The data set contains 569 rows and 32 columns including 30 columns above, ID of data and the model. The dataset is in the .data format so we open the file with Excel, and add headers using the attribute description, then save the file as wdbc.csv



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	id	diagnosis	radius_me	texture_m	perimeter_	area_mea	smoothne	compactn	concavity_	concave p	symmetry_	fractal_dir	radius_se	texture_se	perimeter_	area_se	smoothne	compactn	concavity_	concave p	symmetry_	fractal_dir	radius_wo
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053	8.589	153.4	0.006399	0.04904	0.05373	0.01587	0.03003	0.006193	25.38
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.0186	0.0134	0.01389	0.003532	24.99
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.00615	0.04006	0.03832	0.02058	0.0225	0.004571	23.57
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156	3.445	27.23	0.00911	0.07458	0.05661	0.01867	0.05963	0.009208	14.91
6	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.01149	0.02461	0.05688	0.01885	0.01756	0.005115	22.54
7	843786	M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902	2.217	27.19	0.00751	0.03345	0.03672	0.01137	0.02165	0.005082	15.47
8	844359	M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467	0.7732	3.18	53.91	0.004314	0.01382	0.02254	0.01039	0.01369	0.002179	22.88
9	84458202	M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835	1.377	3.856	50.96	0.008805	0.03029	0.02488	0.01448	0.01486	0.005412	17.06
10	844981	M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063	1.002	2.406	24.32	0.005731	0.03502	0.03553	0.01226	0.02143	0.003749	15.49
11	84501001	M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	0.2976	1.599	2.039	23.94	0.007149	0.07217	0.07743	0.01432	0.01789	0.01008	15.09
12	845636	M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	0.3795	1.187	2.466	40.51	0.004029	0.009269	0.01101	0.007591	0.0146	0.003042	19.19
13	84610002	M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	0.5058	0.9849	3.564	54.16	0.005771	0.04061	0.02791	0.01282	0.02008	0.004144	20.42
14	846226	M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	0.9555	3.568	11.07	116.2	0.003139	0.08297	0.0889	0.0409	0.04484	0.01284	20.96
15	846381	M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	0.4033	1.078	2.903	36.58	0.009769	0.03126	0.05051	0.01992	0.02981	0.003002	16.84
16	84667401	M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	0.2121	1.169	2.061	19.21	0.006429	0.05936	0.05501	0.01628	0.01961	0.008093	15.03
17	84799002	M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	0.37	1.033	2.879	32.55	0.005607	0.0424	0.04741	0.0109	0.01857	0.005466	17.46
18	848406	M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922	0.4727	1.24	3.195	45.4	0.005718	0.01162	0.01998	0.01109	0.0141	0.002085	19.07

We try to access the the files in R:

```
dataset = read.csv("C:\\Users\\hieut\\OneDrive - Oklahoma A and M System\\wdbc.csv", sep = ",")
```

Data consolidation

We just use one file, as the original data. We use some function to check if the data was correctly imported:

```
dim(dataset)
```

```
[1] 569  33 ##569 records and 32 rows
```

```
colnames(dataset)
```

```
 [1] "id"                "diagnosis"         "radius_mean"       "texture_mean"
 [5] "perimeter_mean"    "area_mean"         "smoothness_mean"   "compactness_mean"
 [9] "concavity_mean"    "concave.points_mean" "symmetry_mean"   "fractal_dimension_mean"
[13] "radius_se"         "texture_se"        "perimeter_se"      "area_se"
[17] "smoothness_se"     "compactness_se"    "concavity_se"      "concave.points_se"
[21] "symmetry_se"       "fractal_dimension_se" "radius_worst"   "texture_worst"
[25] "perimeter_worst"   "area_worst"        "smoothness_worst"  "compactness_worst"
[29] "concavity_worst"   "concave.points_worst" "symmetry_worst"  "fractal_dimension_worst"
```

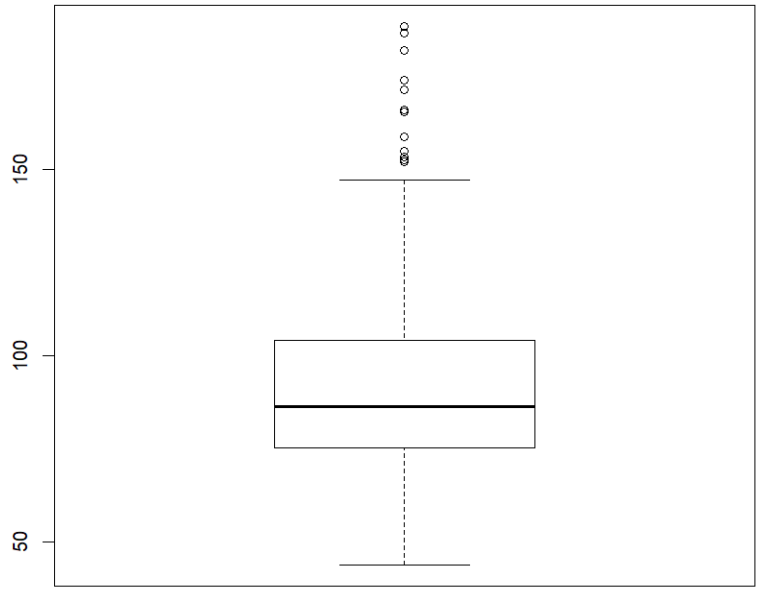
Data cleaning

Using function str(dataset), we see that all are numeric variables, except “diagnosis”, a categorical variable shows the status of tumor is “benign” or “malignant”. We will build a model to predict the tumor status which are categorical data, so we will convert “diagnosis” values as B (benign) = 0 and M (malignant) = 1 in the Data Transformation part.

We test if any null / NA values in the data set using is.null() and is.na() functions. Both return FALSE, so there’s no null values and no missing values.

For outliers, we use box plot for each variable, for example perimeter_mean variable:

```
boxplot(dataset$perimeter_mean)
```



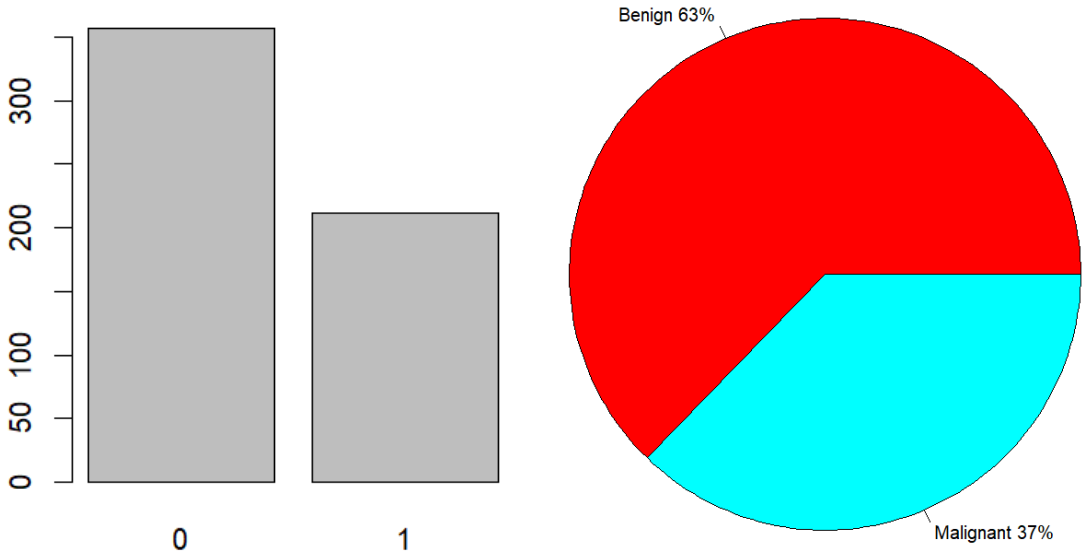
The outliers of all numerical variables have a range of 1-2% of data records (from 10-20 outliers each variable's values). However, we are going to keep those outliers in our analysis, because this is the real data input using machine to measure. And we cannot prove any wrong points in measurement, so there's no reason to remove the outliers.

Data transformation

The first step is to transform "diagnosis" values as B (benign) = 0 and M (malignant) = 1:

```
#Convert categorical data to numeric values
diag1 = gsub("M", 1, dataset$diagnosis)
dataset = data.frame(dataset, diag1)
diag2 = gsub("B", 0, dataset$diag1)
dataset = data.frame(dataset, diag2)
### Remove old diagnosis rows
dataset = subset(dataset, select = -c(diag1))
#Rename the column
names(dataset)[33] = "diag_status"
```

```
> plot(dataset$diag_status)
> summary(dataset$diag_status)
  0   1
357 212
```



There are 357 patients diagnosed with benign cancer status, and 212 patients diagnosed with malignant cancer status.

Data Dictionary

The table below shows the variables names with their descriptions and data types we use in the project:

Variable name	Variable description	Data Type
ID	ID of the patient	Int
diagnosis	The diagnosis of breast tumors (M = malignant, B = benign)	Factors
radius_mean	mean of distances from center to points on the perimeter	Num
texture_mean	mean of gray-scale values	Num
perimeter_mean	mean size of the core tumor	Num
area_mean	mean of area in radius lengths	Num

smoothness_mean	mean of smoothness in radius lengths	Num
compactness_mean	mean of perimeter^2 / area - 1.0	Num
concavity_mean	mean of severity of concave portions of the contour	Num
concave_points_mean	mean for number of concave portions of the contour	Num
symmetry_mean	mean of symmetry value	Num
fractal_dimension_mean	mean for ("coastline approximation" - 1)	Num
radius_se	standard error of distances from center to points on the perimeter	Num
texture_se	standard error of gray-scale values	Num
perimeter_se	standard error in size of the core tumor	Num
area_se	standard error of area in radius lengths	Num
smoothness_se	standard error of smoothness in radius lengths	Num
compactness_se	standard error of perimeter^2 / area - 1.0	Num
concavity_se	standard error of severity of concave portions of the contour	Num
concave_points_se	standard error for number of concave portions of the contour	Num
symmetry_se	standard error of symmetry value	Num
fractal_dimension_se	standard error for ("coastline approximation" – 1)	Num

radius_worst	standard error of distances from center to points on the perimeter	Num
texture_worst	mean of 3 largest values of gray-scale values	Num
perimeter_worst	mean of 3 largest values in size of the core tumor	Num
area_worst	mean of 3 largest values of area in radius lengths	Num
smoothness_worst	mean of 3 largest values of smoothness in radius lengths	Num
compactness_worst	mean of 3 largest values of perimeter^2 / area - 1.0	Num
concavity_worst	mean of 3 largest values of severity of concave portions of the contour	Num
concave_points_worst	mean of 3 largest values for number of concave portions of the contour	Num
symmetry_worst	mean of 3 largest values of symmetry value	Num
fractal_dimension_worst	mean of 3 largest values for ("coastline approximation" – 1)	Num

SELECTING MODELING TECHNIQUES

Our ultimate goal is to be able to categorize tumors as Malignant or Benign. Various medical studies on breast cancer have led to the conclusion that studying various features of the tumor can help in predicting the probability of the tumor converting into breast cancer. The following features of tumor have proven to be good predictors of diagnosis:

- Radius
- Texture
- Perimeter
- Smoothness
- Compactness
- Concavity
- Concave Points
- Symmetry
- Fractal Dimension

Analyzing the issue has led us to decide that building categorizing models for predictive analysis best suits the data set. We chose the following modeling techniques:

DECISION TREE GINI

GOAL: When it comes to designing algorithms that can classify the dataset into desired categories, decision tree is a top choice among many other machine learning algorithms due to its ease of understanding and interpreting. It gives us the power to choose among various attributes which we can use to design the model.

The first type of decision tree modeling technique uses Gini Impurity. We want to classify the tumor as Malignant or Benign and the Decision Tree model will take the factors (features of tumor) in account to decide if it's 0 or 1 (Benign or Malignant). Each node in the tree will act as a test case for one of the attributes and each edge descending from that node will correspond to one of the categories.

Since all predictors are continuous, Gini algorithm works very well. The decision tree will use Gini index to evaluate splits in the data set, for example, the data set may be split according to the radius of tumor or the smoothness of tumor etc.

ASSUMPTIONS: A decision tree is a non-parametric modeling technique, which means that there are no underlying assumptions about the distribution of errors or the data.

DECISION TREE ENTROPY

GOAL: Since our problem is binary classification (Malignant or Benign), we have built another tree which uses Iterative Dichotomiser 3 with Entropy Function and Information Gain as matrices. It uses various features of tumor to categorize it as one of the above stated categories. For example, it may say that, if radius is greater than a certain value and the area is between certain values etc., the patient is at a high risk of cancer.

ASSUMPTIONS: A decision tree is a non-parametric modeling technique, which means that there are no underlying assumptions about the distribution of errors or the data

LOGISTIC REGRESSION

GOAL: We have converted the variable "diagnosis" into binary, i.e. if the diagnosis is Malignant, it will take up the value 1 and if it is Benign, it will take up the value 0. Logistic regression will take in account all the predictors and will calculate the probability of the diagnosis to be Malignant or Benign. It will then categorize the tumor based on a threshold probability value.

Since we want to study how various features of tumor influences whether or not a woman will have cancer, Logistic Regression is a modeling technique which is very well suited to our issue.

ASSUMPTIONS: Following are the assumptions of Logistic Regression,

1) Binary logistic regression requires the dependent variable to be binary. We have converted the dependent variable into binary.

2) Logistic regression requires the observations to be independent of each other. The observations were recorded at different times on different cells. Hence they are independent of each other.

3) logistic regression requires there to be little or no multicollinearity among the independent variables. For this we have calculated the VIF scores of each variable. Some variables show high multicollinearity, to solve the issue, we will split the data set into 3 groups of 10 variables each.

```
>>> print(vif)
const                    1945.673866
diagnosis                   4.431144
radius_mean              3817.259795
texture_mean               11.891280
perimeter_mean           3792.697001
area_mean                 348.115385
smoothness_mean             8.194309
compactness_mean           51.445960
concavity_mean             71.002747
concave points_mean        60.172431
symmetry_mean               4.220806
fractal_dimension_mean     15.756978
radius_se                  75.737325
texture_se                  4.205686
perimeter_se               70.398925
area_se                    41.196467
smoothness_se               4.070801
compactness_se             15.366350
concavity_se               15.914022
concave points_se          11.601253
symmetry_se                 5.179151
fractal_dimension_se        9.724753
radius_worst              815.945630
texture_worst              18.606605
perimeter_worst           405.150023
area_worst                343.494355
smoothness_worst           10.925968
compactness_worst          36.984867
concavity_worst            32.090394
concave points_worst       36.781339
symmetry_worst              9.543023
```

4) Logistic regression typically requires a large sample size, at least 500 samples. Our sample size is above the minimum requirement.

SUPPORT VECTOR MACHINES

GOAL: SVM is one of the most robust and accurate algorithm among the other classification algorithms and since our goal is to accurately classify a tumor into Malignant or Benign, we have used this modeling technique. Since we have a good number of features but a limited amount of data, this technique is proven to work well. This algorithm will build a decision boundary and classify the tumor based on the features as prone to cancer or not.

ASSUMPTIONS: Support Vector Machine is quite tolerant to input data hence it does not have any underlying assumptions.

DATA SPLITTING AND SUBSAMPLING

JUSTIFICATION AND EXPLANATION FOR DATA SPLITTING

The dataset contains 10 main characteristics of the cell nuclei present in the image of a fine needle aspirate (FNA):

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 characteristics were computed for the nucleus in each image, resulting in 3*10 = 30 columns. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

The data set contains 569 rows and 32 columns including 30 columns above, ID of data and the model. To train, test and validate the model, we have split the data set initially into 3 groups of 10 variables i.e. Mean, SE(Standard Error) and Worst.

The data set is then split in a 75-25-0 ratio for training, testing and validation respectively. Since our data set has 569 rows, we decided to choose a number that is not too low for training our models. We did not choose 80-20-0 in order to get better results of model evaluation matrices. Since our dataset has limited number of observations, we did not split it any further for validation.

```
#Splitting Data
data
X = data.iloc[:,1:11]
y = data.iloc[:,0]
traindf, testdf = train_test_split(data, test_size = 0.25)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size = 0.25)
```

COMPARING MEAN, MEDIAN AND STANDARD DEVIATION

```
>>> std_dev
diagnosis                 0.488249
radius_mean               3.519726
texture_mean              4.176740
perimeter_mean           24.260087
area_mean               348.484300
smoothness_mean           0.014227
compactness_mean          0.052715
concavity_mean            0.077758
concave points_mean       0.038464
symmetry_mean             0.027273
fractal_dimension_mean    0.007035
radius_se                 0.295626
texture_se                0.558419
perimeter_se              2.168504
area_se                  49.416442
smoothness_se             0.003157
compactness_se            0.018234
concavity_se              0.027206
concave points_se         0.006151
symmetry_se               0.008193
fractal_dimension_se      0.002381
radius_worst              4.853900
texture_worst             6.011619
perimeter_worst          33.767209
area_worst              570.473690
smoothness_worst          0.022891
compactness_worst         0.153658
concavity_worst           0.201178
concave points_worst      0.065324
symmetry_worst            0.062408
fractal_dimension_worst   0.017567
```

```
... t2, p2 = sts.ttest_ind(traindf,testdf)
>>> print("t = " + str(t2))
t = [ 1.45524561 -1.17098474  0.70322831 -1.06756894 -0.98893634  0.6766192
  0.75628159 -0.12087097 -0.33487814  0.20692904  1.40617271  0.7302798
  0.81312329  1.02009514  0.75090533  1.11853171  1.06053281 -0.03175332
  0.45461108  0.96530189  0.49410951 -0.62603891  0.86532238 -0.46764621
 -0.3881406   1.07994943  0.62578294 -0.0144634   0.00340502  1.32857024
  1.33031333]
>>> print("p = " + str(p2))
p = [0.14615468 0.24209679 0.48220213 0.28616932 0.32311611 0.49892346
 0.44979448 0.90383606 0.73784086 0.83613957 0.16022063 0.46552079
 0.41648867 0.30811839 0.45302115 0.26381363 0.28935381 0.97467995
 0.64956306 0.33480515 0.62142021 0.53154135 0.38722784 0.64021729
 0.69805767 0.28062386 0.53170913 0.98846536 0.99728439 0.18452425
 0.18394987]
>>>
```

Null Hypothesis: There is no difference in the means of training and testing set.

Alternate Hypothesis: The means of training and testing set are different.

For α = 0.05, we fail to reject the null hypothesis.

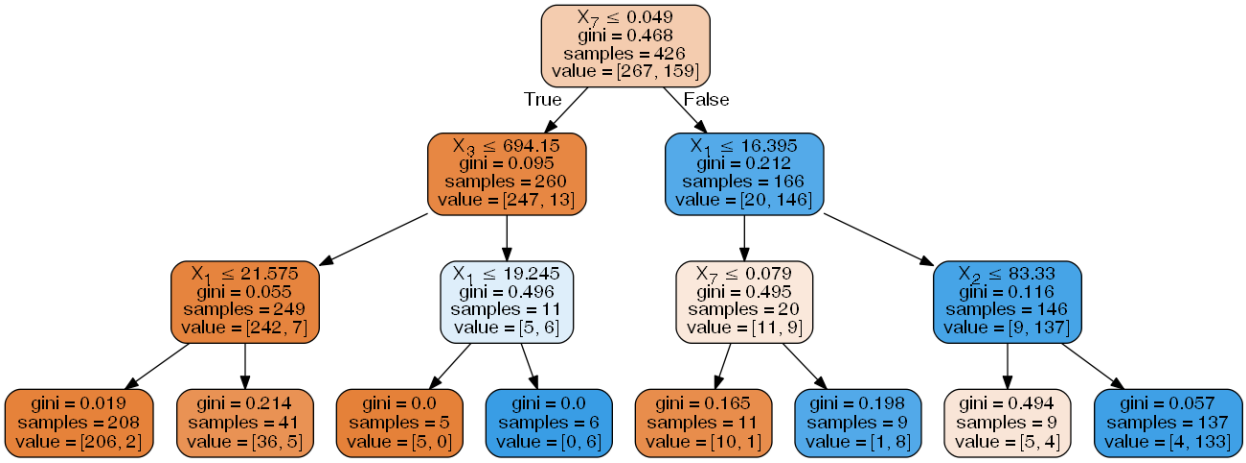
What does that mean?

The training and testing datasets are not significantly different from each other, which is desirable. It is because we don't want that our model performs well on the training data set but it fails to match up to the same performance for the test data.

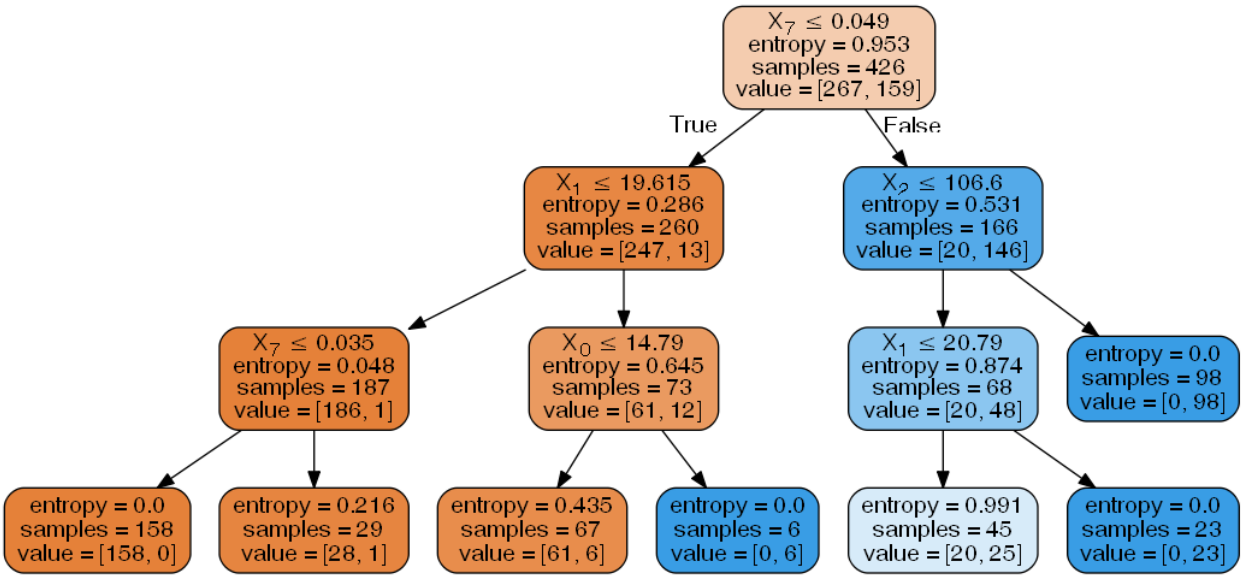
The values of standard deviation for most variables are low, which signifies the above stated point. The standard deviation of **area_mean**, **area_se** and **area_worst** are high. It can be removed by using subsampling methods like k-fold etc.

BUILDING THE MODELS

DECISION TREE GINI



DECISION TREE ENTROPY



LOGISTIC REGRESSION

```
>>> print(result.summary2())
                         Results: Logit
================================================================================
Model:                  Logit              Pseudo R-squared:    0.799
Dependent Variable:     diagnosis          AIC:                 132.9305
Date:                   2019-05-04 18:39   BIC:                 173.4749
No. Observations:       426                Log-Likelihood:      -56.465
Df Model:               9                  LL-Null:             -281.44
Df Residuals:           416                LLR p-value:         2.9397e-91
Converged:              1.0000             Scale:               1.0000
No. Iterations:         11.0000
--------------------------------------------------------------------------------
                         Coef.   Std.Err.    z    P>|z|    [0.025   0.975]
--------------------------------------------------------------------------------
radius_mean              -1.7318   3.8330 -0.4518 0.6514   -9.2442   5.7807
texture_mean              0.3810   0.0722  5.2762 0.0000    0.2395   0.5226
perimeter_mean           -0.2087   0.5700 -0.3660 0.7143   -1.3259   0.9086
area_mean                 0.0448   0.0114  3.9212 0.0001    0.0224   0.0672
smoothness_mean          78.5019  32.6175  2.4067 0.0161   14.5727 142.4311
compactness_mean         16.3337  22.1822  0.7363 0.4615  -27.1426  59.8100
concavity_mean            5.5162   8.7994  0.6269 0.5307  -11.7304  22.7628
concave points_mean      64.6593  29.5226  2.1902 0.0285    6.7961 122.5226
symmetry_mean            20.1190  11.2317  1.7913 0.0733   -1.8947  42.1327
fractal_dimension_mean -143.3650  85.1239 -1.6842 0.0921 -310.2048  23.4749
================================================================================
```

SUPPORT VECTOR MACHINES

ASSESSING THE MODELS

CHOSEN MODEL EVALUATION METRICS:

For each model, following matrices have been assessed:

1) **Accuracy Score:** Since our goal is to build classification models, we have to study the confusion matrix in order to understand how well our model is classifying the tumor as malignant or benign. Hence we are studying the accuracy score.

2) **Precision:** The score will tell us that out of the tumors classified as cancerous, how many are actually cancerous.

3) **Recall:** This score tells us that how many cancerous tumors could we catch. For our problem, this is a very important metric.

4) **F1 Score:** To seek balance between the precision and recall, we will study F11 score of each model.

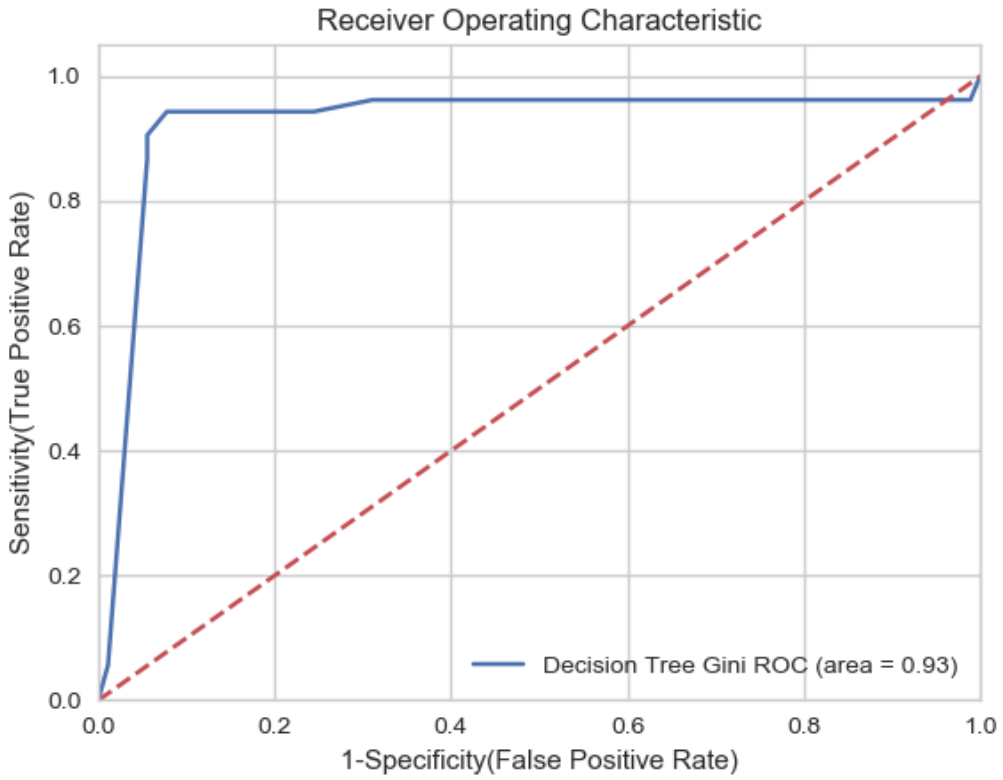
5) **Area under ROC curve:** AUC-ROC Curve is a very important matric to measure performance of our models. It will tell us how much our model is capable to distinguish between the two classes, Malignant and Benign.

DECISION TREE GINI

```
>>> gini = DecisionTreeClassifier(criterion = "gini", random_state = 100,max_depth=3, min_samples_leaf=5)
>>> gini.fit(X_train, y_train)
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=100,
            splitter='best')
>>> y_pred_gini = gini.predict(X_test)
>>> print("Predicted values:")
Predicted values:
>>> print(y_pred_gini)
[1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1
 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0
 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 1]
>>> print("Confusion Matrix: ", confusion_matrix(y_test, y_pred_gini))
Confusion Matrix:  [[85  5]
 [ 5 48]]
>>> print ("Accuracy : ", accuracy_score(y_test,y_pred_gini)*100)
Accuracy :  93.00699300699301
>>> print("Report : ", classification_report(y_test, y_pred_gini))
Report :              precision    recall  f1-score   support

          0       0.94      0.94      0.94        90
          1       0.91      0.91      0.91        53

avg / total       0.93      0.93      0.93       143
```



Accuracy= 93.0069

Precision= 0.93

Recall= 0.93

F1 Score= 0.93

AUC-ROC= 0.93

Strength= All the matrices have high values; hence the model is doing a good job in classification. Since the predictor variables are continuous, gini outperforms entropy.

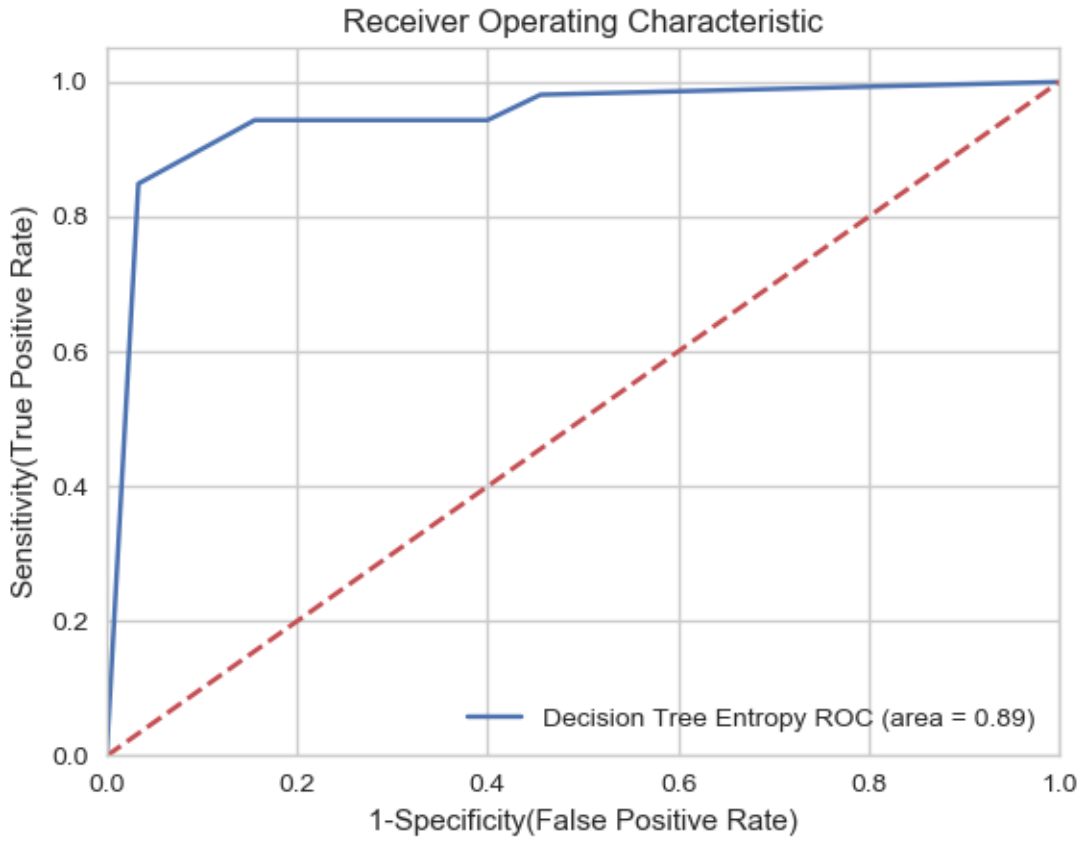
Weakness= While designing the model, we have to specify a lot of parameters to obtain the required results and scores.

DECISION TREE ENTROPY

```
>>> entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100,max_depth=3, min_samples_leaf=5)
>>> entropy.fit(X_train, y_train)
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=100,
            splitter='best')
>>> y_pred_entropy = entropy.predict(X_test)
>>> print("Predicted values:")
Predicted values:
>>> print(y_pred_entropy)
[1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0
 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1]
>>> print("Confusion Matrix: ", confusion_matrix(y_test, y_pred_entropy))
Confusion Matrix:  [[76 14]
 [ 3 50]]
>>> print ("Accuracy : ", accuracy_score(y_test,y_pred_entropy)*100)
Accuracy :  88.11188811188812
>>> print("Report : ", classification_report(y_test, y_pred_entropy))
Report :              precision    recall  f1-score   support

          0       0.96      0.84      0.90        90
          1       0.78      0.94      0.85        53

avg / total       0.90      0.88      0.88       143
```



Accuracy= 88.1118

Precision= 0.90

Recall= 0.88

F1 Score= 0.88

AUC-ROC= 0.89

Strength= The precision score is quite high. Which means the model has classified a good number of tumors correctly.

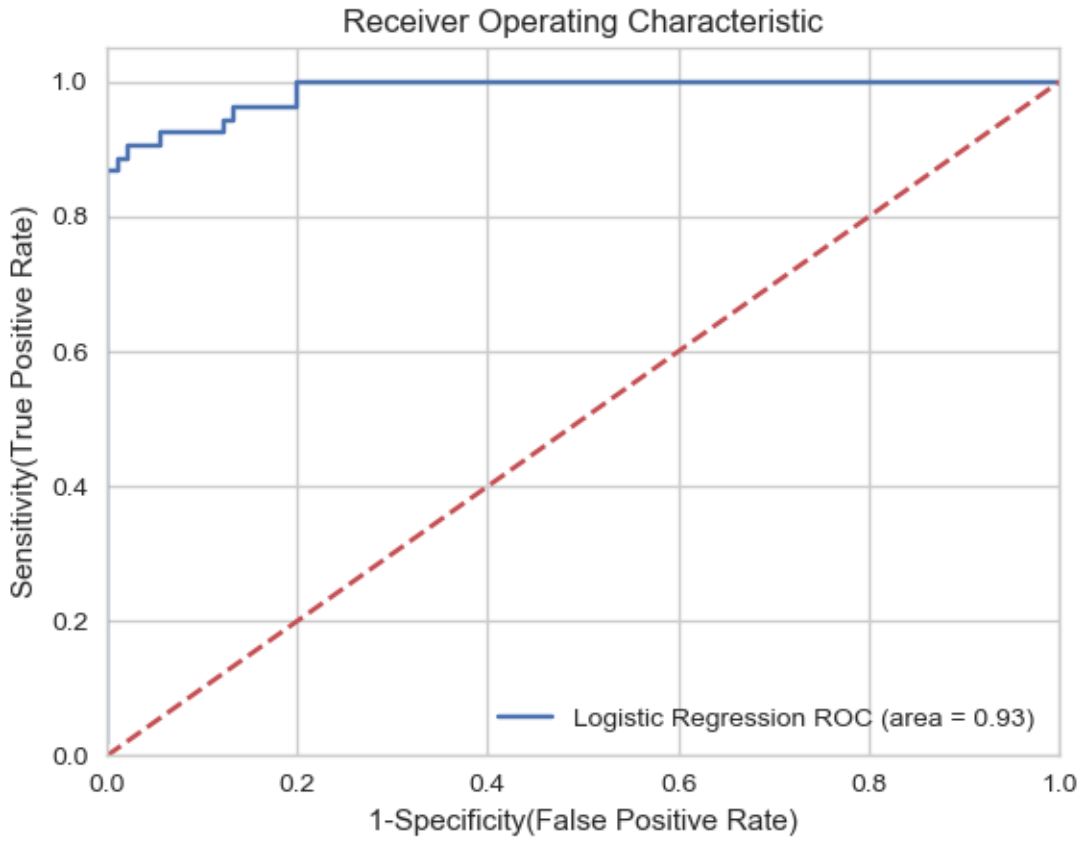
Weakness= The scores are not as high as the ones in gini, it is due to the fact that entropy is used for exploratory analysis whereas gini is used to reduce misclassifications. Entropy is mostly used when the predictors are categorical, but in our case, the variables are continuous.

LOGISTIC REGRESSION

```
>>> log_reg = LogisticRegression(random_state=0)
>>> log_reg.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
>>> y_pred_log_reg = log_reg.predict(X_test)
>>> print("Predicted values:")
Predicted values:
>>> print(y_pred_log_reg)
[1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1]
>>> print("Confusion Matrix: ", confusion_matrix(y_test, y_pred_log_reg))
Confusion Matrix:  [[86  4]
 [ 5 48]]
>>> print ("Accuracy : ", accuracy_score(y_test,y_pred_log_reg)*100)
Accuracy :  93.7062937062937
>>> print("Report : ", classification_report(y_test, y_pred_log_reg))
Report :              precision    recall  f1-score   support

          0       0.95      0.96      0.95        90
          1       0.92      0.91      0.91        53

avg / total       0.94      0.94      0.94       143
```



Accuracy= 93.7062

Precision= 0.94

Recall= 0.94

F1 Score= 0.94

AUC-ROC= 0.93

Strength= Since our goal is to catch as many cancerous tumors as possible, we want a high recall score. The model does a good job in classifying the tumors correctly.

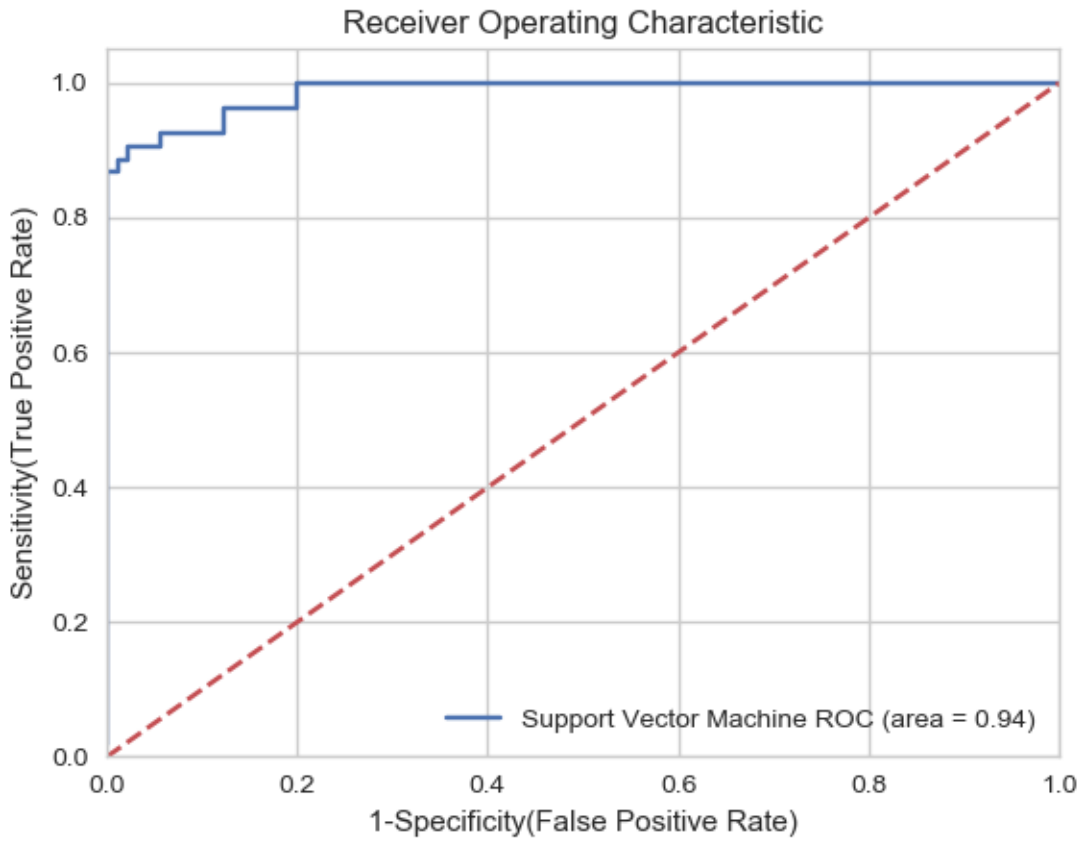
Weakness= There is no distinct weakness of the model. It is performing well.

SUPPORT VECTOR MACHINES

```
>>> svm_model = svm.LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
...      intercept_scaling=1, loss='squared_hinge', max_iter=1000,
...      multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
...      verbose=0)
>>> Support_Vector_Machine = CalibratedClassifierCV(svm_model)
>>> Support_Vector_Machine.fit(X_train, y_train)
CalibratedClassifierCV(base_estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0),
            cv=3, method='sigmoid')
>>> y_pred_svm = Support_Vector_Machine.predict(X_test)
>>> print("Predicted values:")
Predicted values:
>>> print(y_pred_svm)
[1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1]
>>> print("Confusion Matrix: ", confusion_matrix(y_test, y_pred_svm))
Confusion Matrix:  [[87  3]
 [ 5 48]]
>>> print ("Accuracy : ", accuracy_score(y_test,y_pred_svm)*100)
Accuracy :  94.4055944055944
>>> print("Report : ", classification_report(y_test, y_pred_svm))
Report :              precision    recall  f1-score   support

          0       0.95      0.97      0.96        90
          1       0.94      0.91      0.92        53

avg / total       0.94      0.94      0.94       143
```



Accuracy= 94.4055

Precision= 0.94

Recall= 0.94

F1 Score= 0.94

AUC-ROC= 0.94

Strength= The scores above are all high, which shows that the model is doing a wonderful job at correctly classifying the tumor as Malignant or Benign.

Weakness= SVM does complex data transformations, hence it is called 'Black Box'. It is quite difficult to understand the process going inside. Setting right parameters is also a problem.

FINAL MODEL

The final model that we have chosen for the issue in hand is "Support Vector Machines" for the following reasons:

- High Accuracy, Precision, Recall and AUC-ROC numbers. The statistics of the chosen model are better than all the other models that we have built. Since we want to catch as many cancerous tumors as possible, we require a high recall which is given by the model (0.94).
- Since SVM is used when we have a good number of features, it well suits our data set.
- Also, we have less number of observations. All the other models require good amount of data to train and test, whereas SVM performs well even when the number of observations are limited.