INDUSTRY INTERNSHIP PROJECT

ON

PREDICTION OF BREAST CANCER CAUSED BY GENETIC MUTATION USING MACHINE LEARNING

INDUSTRY NAME: AT&T

INSTITUTE:

M.K.S.S.S CUMMINS COLLEGE OF ENGINEERING FOR WOMEN, PUNE

INDUSTRY GUIDE:

MR. PETER BAKER

SUBMITTED BY:

PRERITA JAISWAL

SYNOPSIS:

With increasing dependence on technology in all the domains, professionals are aiming for even higher pros of Machine Learning in particular by using it for the prediction and prognosis of Breast Cancer and not just its diagnosis. Breast Cancer prediction is different from its diagnosis and detection because in cancer prediction we work on three aspects: The risk of developing cancer, The reiteration of cancer, Surviving probability after cancer. It is a very difficult and risk-prone task to perform as further diagnosis and treatment depends on this prognosis and prediction. After the cleaning, processing and splitting of the dataset for training, testing and cross-validation, a random model will be built with maximum possible Multiclass so as to have a standard to compare ML prediction models with and decide which algorithm would be best to build the final Machine learning model for maximum precision. A categorical analysis on the columns given in the dataset will be done so as to know the impact and significance of each attribute on the final prediction. Data preprocessing is done to get it in the proper format for the application of the machine learning algorithms. The Machine Learning Classification techniques which have been used in the Project are K-nearest neighbours (K-NN), Support Vector Machine (SVM), Logistic Regression, Naïve Bayes and Random Forest.

TECHNOLOGY USED:

Machine Learning is an application of **Artificial Intelligence** (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide.

Machine learning is widely used in **bioinformatics** and particularly in **Cancer Diagnosis**. Cancer diagnosis is one of the most studied problems in the medical domain. Early detection of cancer is essential for a rapid response and better chances of cure. Unfortunately, early detection of cancer is often difficult because the symptoms of the disease at the beginning are absent. Thus, it is necessary to discover and interpret new knowledge to prevent and minimize the risk adverse consequences. The interpretation of genetic mutation is usually done manually through clinical pathologists which makes the process very slow, so to pace up the process we can apply Machine Learning and Artificial Intelligence which will make the predictions faster and more accurate.

DATASET USED:

Wisconsin Diagnostic Breast Cancer (WDBC) dataset obtained by the University of Wisconsin Hospital is used to classify tumours as Benign or Malignant.

Dataset Link: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

MACHINE LEARNING CLASSIFICATION TECHNIQUES USED:

K Nearest Neighbour:- It is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.

Support Vector Machine:- They are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

Logistic Regression:- Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Naive Bayes:- These are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.
Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Random Forest:- These are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

PROJECT CODE AND OUTPUT:-

```
#import libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#Load data

from google.colab import files
uploaded=files.upload()
df=pd.read_csv('data.csv')
#count no. of rows and cols in dataset
df.shape
```

OUTPUT->

(569, 32)

```
#count number of empty values in each col (NaN, NAN, na)
df.isna().sum()
#Get info of the Dataset
df.info()
#Get count of number of Malignant(M: Cancerous) or Belign(B) cells
df['diagnosis'].value_counts()
```

OUTPUT->

```
B    357
M    212
```

```
#Visualize
sns.countplot(df['diagnosis'],label='count')
#Look at the data types to see which col need to be encoded
df.dtypes
#Encode the categorical data values
from sklearn.preprocessing import LabelEncoder
labelencoder_Y =LabelEncoder()
df.iloc[:,1]=labelencoder_Y.fit_transform(df.iloc[:,1].values)
#Pair Plot
sns.pairplot(df.iloc[:,1:6], hue='diagnosis') #to get diagnosis points
#correlation
df.iloc[:,1:12].corr()
```

```
# counter plot of feature mean radius
plt.figure(figsize = (18,8))
sns.countplot(df['radius_mean'])
#Visualize the correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.iloc[:,1:12].corr(),annot=True, fmt=".0%")
#split the dataset into independent(X) and dependent(Y) data sets
x=df.iloc[:,2:31].values
y=df.iloc[:,1].values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
x_train,y_train=shuffle(x_train,y_train)
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)

#MACHINE LEARNING CLASSIFICATION TECHNIQUES
#K Nearest Neighbor
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5, metric= 'minkowski')
knn.fit(x_train,y_train)
print("KNN", knn.score(x_train,y_train))
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('Model KNN')
cm=confusion_matrix(y_test, knn.predict(x_test))
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print()
print(cm)
print()
print('Testing Accuracy=', (TP+TN)/(TP+TN+FN+FP))
print()
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
```

```
sns.heatmap(cm, annot = True)
plt.show()
print()
print("Classification Report of KNN")
print()
print(classification_report(y_test,knn.predict(x_test)))
print("Accuracy Score: ",accuracy_score(y_test,knn.predict(x_test)))
print()

#Support Vector Machine
from sklearn.svm import SVC
svc=SVC()
svc.fit(x_train,y_train)
print("SVC", svc.score(x_train,y_train))
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('Model Support Vector Machine')
cm=confusion_matrix(y_test, svc.predict(x_test))
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print()
print(cm)
print()
print('Testing Accuracy=', (TP+TN)/(TP+TN+FN+FP))
print()
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
print()
print("Classification Report of Support Vector Machine")
print()
print(classification_report(y_test,svc.predict(x_test)))
print("Accuracy Score: ",accuracy_score(y_test,svc.predict(x_test)))
print()
```

```
#Logictic Regression
from sklearn.linear_model import LogisticRegression
log=LogisticRegression(random_state=0)
log.fit(x_train,y_train)
print("Logictic Regression", svc.score(x_train,y_train))
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('Model: Logistic Regression')
cm=confusion_matrix(y_test, log.predict(x_test))
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print()
print(cm)
print()
print('Testing Accuracy=', (TP+TN)/(TP+TN+FN+FP))
print()
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
print()
print("Classification Report of Logistic Regression")
print()
print(classification_report(y_test,log.predict(x_test)))
print("Accuracy Score: ",accuracy_score(y_test,log.predict(x_test)))
print()
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(x_train,y_train)

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('Model: Naive Bayes')
cm=confusion_matrix(y_test, nb.predict(x_test))
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
```

```
FP=cm[0][1]
print()
print(cm)
print()
print('Testing Accuracy=', (TP+TN)/(TP+TN+FN+FP))
print()
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
print()
print("Classification Report of Naive Bayes")
print()
print(classification_report(y_test,nb.predict(x_test)))
print("Accuracy Score: ",accuracy_score(y_test,nb.predict(x_test)))
print()
#Random Forest
from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier(n_estimators=10, criterion='entropy')
forest.fit(x_train,y_train)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('Model: Random Forest')
cm=confusion_matrix(y_test, forest.predict(x_test))
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print()
print(cm)
print()
print('Testing Accuracy=', (TP+TN)/(TP+TN+FN+FP))
print()
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()
print()
print("Classification Report of Random Forest")
print()
print(classification_report(y_test,forest.predict(x_test)))
```

```
print("Accuracy Score: ",accuracy_score(y_test,forest.predict(x_test)))

#Comparing the confusion matrices
from sklearn.metrics import confusion_matrix
print("Confusion Matrix of KNN: \n Training Set")
cm=confusion_matrix(y_train, knn.predict(x_train))
print(cm,"\n\nTesting State")
cm=confusion_matrix(y_test, knn.predict(x_test))
print(cm)
print("\nConfusion Matrix of Support Vector Machine: \n Training Set")
cm=confusion_matrix(y_train, svc.predict(x_train))
print(cm)
print("\nTesting State")
cm=confusion_matrix(y_test, svc.predict(x_test))
print(cm)
print("\nConfusion Matrix of Support Vector Machine: \n Training Set")
cm=confusion_matrix(y_train, log.predict(x_train))
print(cm)
print("\nTesting State")
cm=confusion_matrix(y_test, log.predict(x_test))
print(cm)
print("\nConfusion Matrix of Support Vector Machine: \n Training Set")
cm=confusion_matrix(y_train, nb.predict(x_train))
print(cm)
print("\nTesting State")
cm=confusion_matrix(y_test, nb.predict(x_test))
print(cm)
print("\nConfusion Matrix of Support Vector Machine: \n Training Set")
cm=confusion_matrix(y_train, forest.predict(x_train))
print(cm)
print("\nTesting State")
cm=confusion_matrix(y_test, forest.predict(x_test))
print(cm)
```

OUTPUT AND DIAGRAMS

Dataset Used:

1	id	diagnosis	radius_mean	texture_mea	perimeter_m	area_mean	smoothness_	compactness	concavity_m	concave poir	symmetry_m	fractal_dime	radius_se	texture_se	perimeter_se	area_se	smoothness_	compactness	concavity_se	concave poir	symmetry_se	fractal_
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.0
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	0.00
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	0.00
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	0.0
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	0.00
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	0.00
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	0.00
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	0.00
19	84862001	M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	0.5692	1.073	3.854	54.18	0.007026	0.02501	0.03188	0.01297	0.01689	0.00
20	849014	M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	0.7582	1.017	5.865	112.4	0.006494	0.01893	0.03391	0.01521	0.01356	0.00
21	8510426	B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	0.2699	0.7886	2.058	23.56	0.008462	0.0146	0.02387	0.01315	0.0198	0
22	8510653	B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	0.1852	0.7477	1.383	14.67	0.004097	0.01898	0.01698	0.00649	0.01678	0.00
23	8510824	B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	0.2773	0.9768	1.909	15.7	0.009606	0.01432	0.01985	0.01421	0.02027	0.00
24	8511133	M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032	0.4388	0.7096	3.384	44.91	0.006789	0.05328	0.06446	0.02252	0.03672	0.00
25	851509	M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278	0.6917	1.127	4.303	93.99	0.004728	0.01259	0.01715	0.01038	0.01083	0.00
26	852552	M	16.65	21.38	110	904.6	0.1121	0.1457	0.1525	0.0917	0.1995	0.0633	0.8068	0.9017	5.455	102.6	0.006048	0.01882	0.02741	0.0113	0.01468	0.00
27	852631	M	17.14	16.4	116	912.7	0.1186	0.2276	0.2229	0.1401	0.304	0.07413	1.046	0.976	7.276	111.4	0.008029	0.03799	0.03732	0.02397	0.02308	0.00
28	852763	M	14.58	21.53	97.41	644.8	0.1054	0.1868	0.1425	0.08783	0.2252	0.06924	0.2545	0.9832	2.11	21.05	0.004452	0.03055	0.02681	0.01352	0.01454	0.00
29	852781	M	18.61	20.25	122.1	1094	0.0944	0.1066	0.149	0.07731	0.1697	0.05699	0.8529	1.849	5.632	93.54	0.01075	0.02722	0.05081	0.01911	0.02293	0.00
30	852973	M	15.3	25.27	102.4	732.4	0.1082	0.1697	0.1683	0.08751	0.1926	0.0654	0.439	1.012	3.498	43.5	0.005233	0.03057	0.03576	0.01083	0.01768	0.00
31	853201	M	17.57	15.05	115	955.1	0.09847	0.1157	0.09875	0.07953	0.1739	0.06149	0.6003	0.8225	4.655	61.1	0.005627	0.03033	0.03407	0.01354	0.01925	0.00
32	853401	M	18.63	25.11	124.8	1088	0.1064	0.1887	0.2319	0.1244	0.2183	0.06197	0.8307	1.466	5.574	105	0.006248	0.03374	0.05196	0.01158	0.02007	0.0
33	853612	M	11.84	18.7	77.93	440.6	0.1109	0.1516	0.1218	0.05182	0.2301	0.07799	0.4825	1.03	3.475	41	0.005551	0.03414	0.04205	0.01044	0.02273	0.00
34	85382601	M	17.02	23.98	112.8	899.3	0.1197	0.1496	0.2417	0.1203	0.2248	0.06382	0.6009	1.398	3.999	67.78	0.008268	0.03082	0.05042	0.01112	0.02102	0.00
35	854002	M	19.27	26.47	127.9	1162	0.09401	0.1719	0.1657	0.07593	0.1853	0.06261	0.5558	0.6062	3.528	68.17	0.005015	0.03318	0.03497	0.009643	0.01543	0.00
36	854039	M	16.13	17.88	107	807.2	0.104	0.1559	0.1354	0.07752	0.1998	0.06515	0.334	0.6857	2.183	35.03	0.004185	0.02868	0.02664	0.009067	0.01703	0.00

No. of nan entries in all the columns:

id 0

diagnosis 0

radius_mean 0

texture_mean 0

perimeter_mean 0

area_mean 0

smoothness_mean 0

compactness_mean 0

concavity_mean 0

concave points_mean 0

symmetry_mean 0

fractal_dimension_mean 0

radius_se 0

texture_se 0

perimeter_se 0

area_se 0

smoothness_se 0

compactness_se 0

concavity_se 0

concave points_se 0

symmetry_se 0
fractal_dimension_se 0
radius_worst 0
texture_worst 0
perimeter_worst 0
area_worst 0
smoothness_worst 0
compactness_worst 0
concavity_worst 0
concave points_worst 0
symmetry_worst 0
fractal_dimension_worst 0

Information of Dataset:

OUTPUT->
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):

#	Column	Non-Null Count	Dtype
---	------	--------------	-----
0	id	569 non-null	int64
1	diagnosis	569 non-null	object
2	radius_mean	569 non-null	float64
3	texture_mean	569 non-null	float64
4	perimeter_mean	569 non-null	float64
5	area_mean	569 non-null	float64
6	smoothness_mean	569 non-null	float64
7	compactness_mean	569 non-null	float64
8	concavity_mean	569 non-null	float64
9	concave points_mean	569 non-null	float64
10	symmetry_mean	569 non-null	float64
11	fractal_dimension_mean	569 non-null	float64
12	radius_se	569 non-null	float64
13	texture_se	569 non-null	float64
14	perimeter_se	569 non-null	float64
15	area_se	569 non-null	float64
16	smoothness_se	569 non-null	float64
17	compactness_se	569 non-null	float64
18	concavity_se	569 non-null	float64

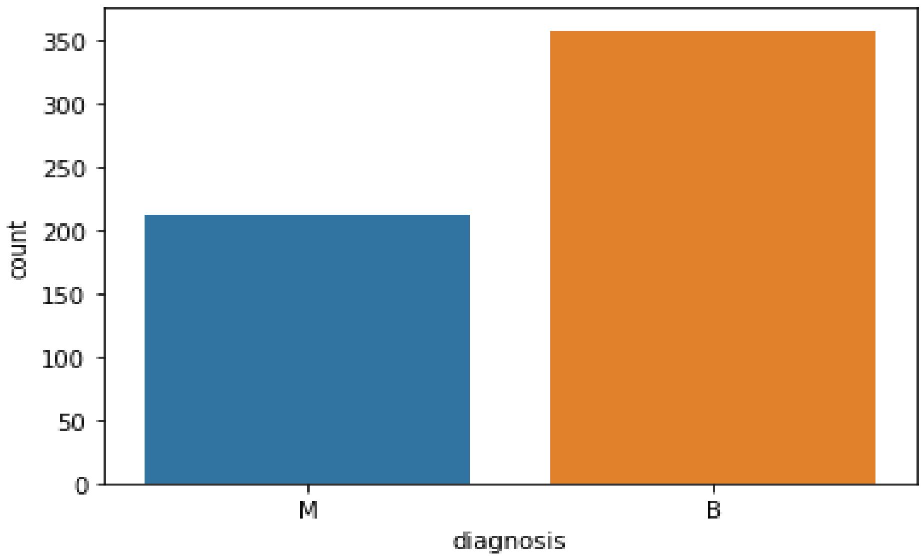
19	concave points_se	569 non-null	float64
20	symmetry_se	569 non-null	float64
21	fractal_dimension_se	569 non-null	float64
22	radius_worst	569 non-null	float64
23	texture_worst	569 non-null	float64
24	perimeter_worst	569 non-null	float64
25	area_worst	569 non-null	float64
26	smoothness_worst	569 non-null	float64
27	compactness_worst	569 non-null	float64
28	concavity_worst	569 non-null	float64
29	concave points_worst	569 non-null	float64
30	symmetry_worst	569 non-null	float64
31	fractal_dimension_worst	569 non-null	float64

Count of Malignant and Belign cells and visualize it

OUTPUT->

B 357

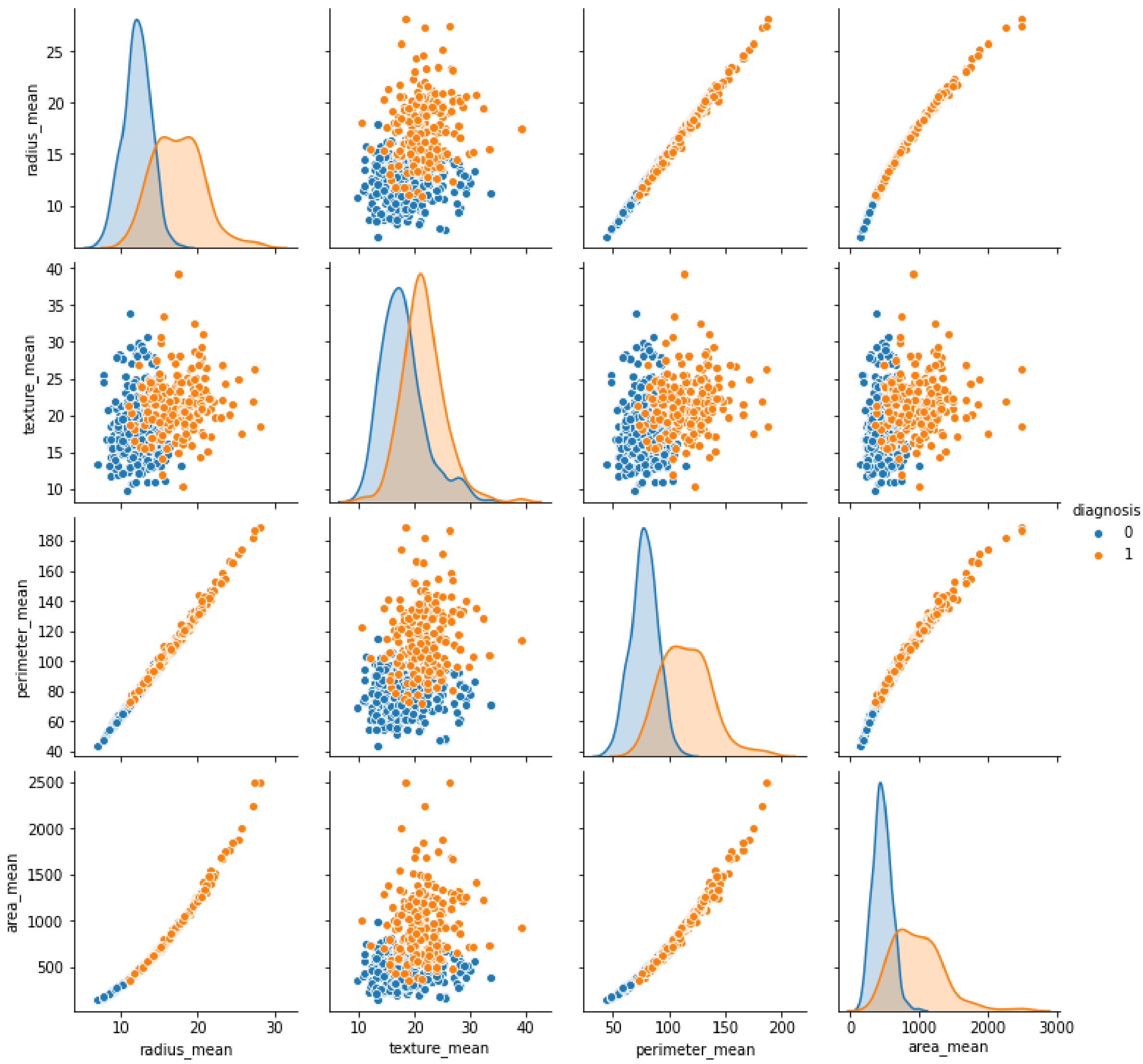
M 212



Datatypes of the Columns

id	int64
diagnosis	object
radius_mean	float64
texture_mean	float64
perimeter_mean	float64
area_mean	float64
smoothness_mean	float64
compactness_mean	float64
concavity_mean	float64
concave points_mean	float64
symmetry_mean	float64
fractal_dimension_mean	float64
radius_se	float64
texture_se	float64
perimeter_se	float64
area_se	float64
smoothness_se	float64
compactness_se	float64
concavity_se	float64
concave points_se	float64
symmetry_se	float64
fractal_dimension_se	float64
radius_worst	float64
texture_worst	float64
perimeter_worst	float64
area_worst	float64
smoothness_worst	float64
compactness_worst	float64
concavity_worst	float64
concave points_worst	float64
symmetry_worst	float64
fractal_dimension_worst	float64

Pair Plot to get Diagnosis Points

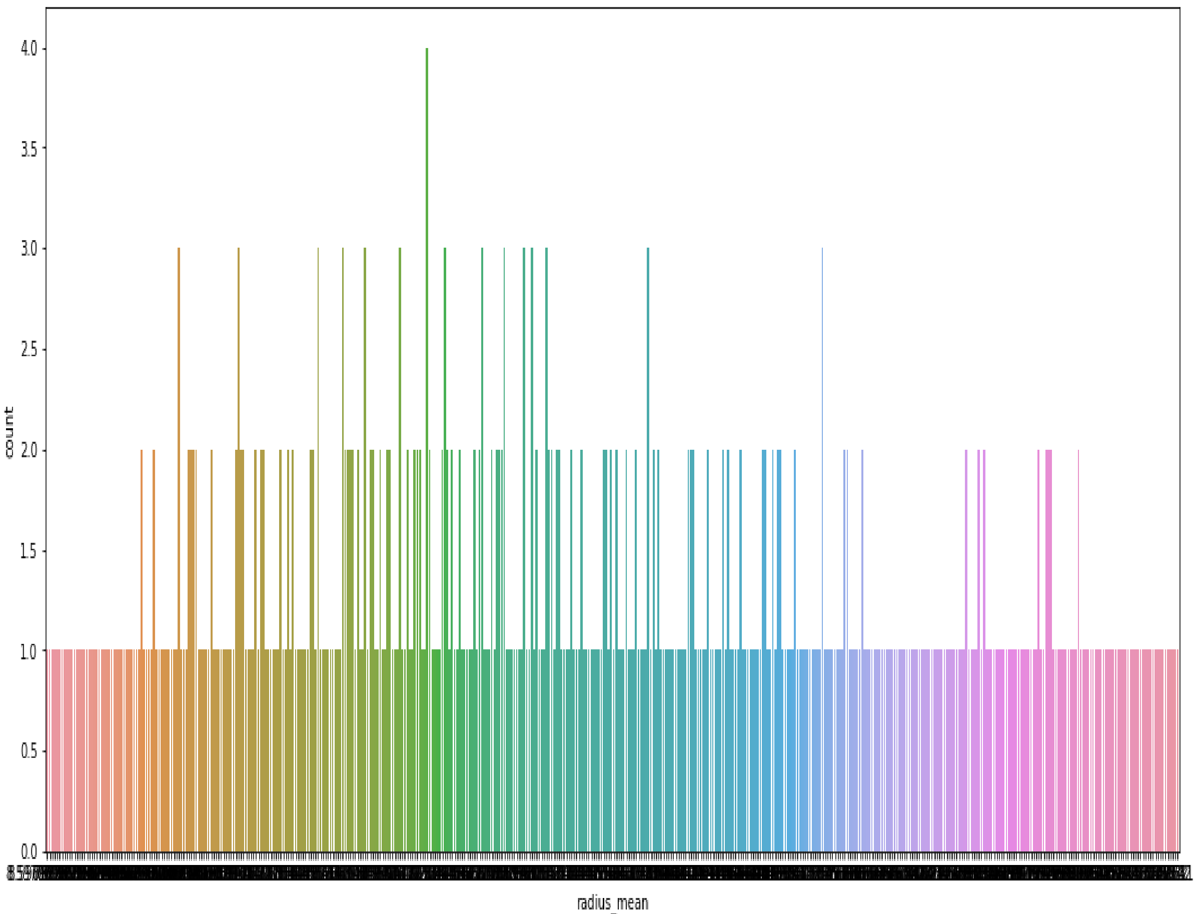


Correlation of Columns

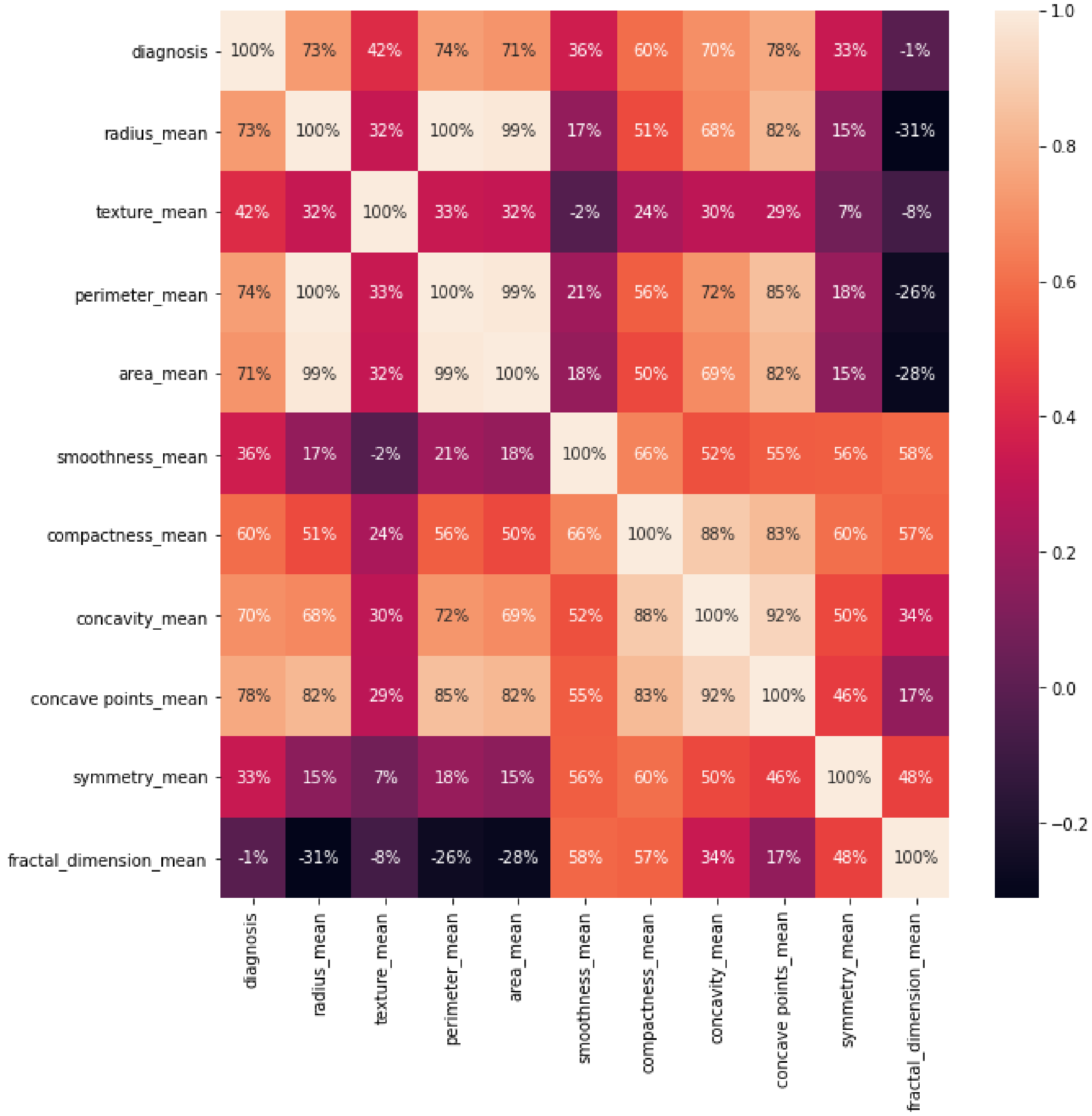
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360	0.776614	0.330499	-0.012838
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	-0.311631
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293464	0.071401	-0.076437
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0.183027	-0.261477
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	-0.283110

smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	0.584792
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.565369
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.336783
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.166917
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.479921
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.000000

Counter Plot of Feature “Mean Radius”



Visualize Correlation using Heatmap



MODEL 1: KNN

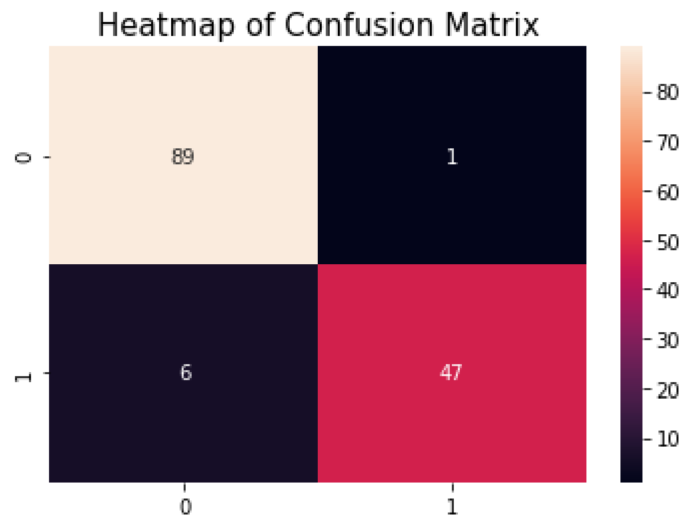
KNN- 0.9765258215962441

Model KNN:

[[89 1]

 [6 47]]

Testing Accuracy= 0.951048951048951



Classification Report of KNN

	precision	recall	f1-score	support
0	0.94	0.99	0.96	90
1	0.98	0.89	0.93	53
accuracy			0.95	143
macro avg	0.96	0.94	0.95	143
weighted avg	0.95	0.95	0.95	143

Accuracy Score: 0.951048951048951

MODEL 2: SVM

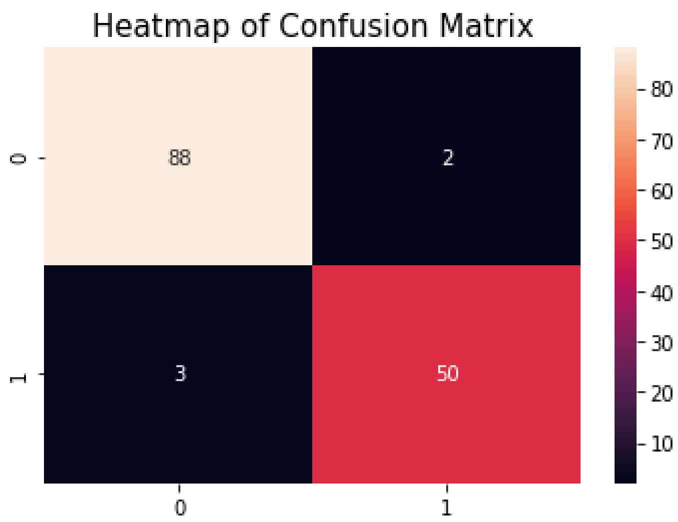
SVM 0.9835680751173709

Model Support Vector Machine

[[88 2]

 [3 50]]

Testing Accuracy= 0.9790209790209791



Classification Report of Support Vector Machine

	precision	recall	f1-score	support
0	0.97	0.98	0.97	90
1	0.96	0.94	0.95	53
accuracy			0.97	143
macro avg	0.96	0.96	0.96	143
weighted avg	0.96	0.97	0.96	143

Accuracy Score: 0.9790209790209791

MODEL 3: LOGISTIC REGRESSION

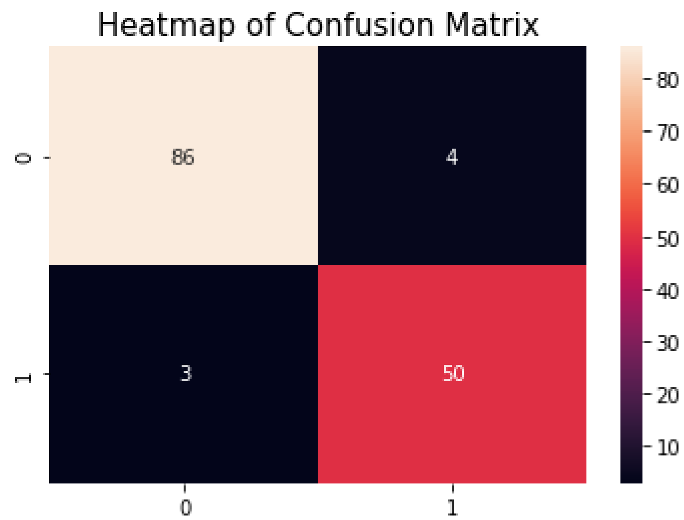
Logictic Regression 0.9835680751173709

Model: Logistic Regression

[[86 4]

 [3 50]]

Testing Accuracy= 0.951048951048951



Classification Report of Logistic Regression

	precision	recall	f1-score	support
0	0.97	0.96	0.96	90
1	0.93	0.94	0.93	53
accuracy			0.95	143
macro avg	0.95	0.95	0.95	143
weighted avg	0.95	0.95	0.95	143

Accuracy Score: 0.951048951048951

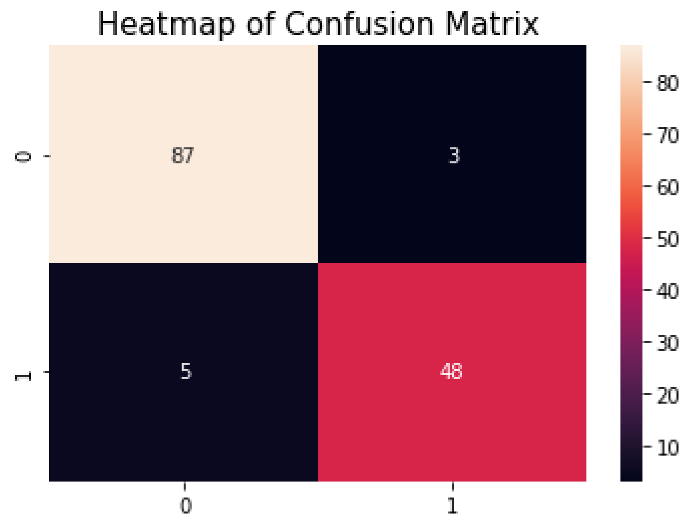
MODEL 4: NAIVE BAYES

Model: Naive Bayes

[[87 3]

 [5 48]]

Testing Accuracy= 0.9440559440559441



Classification Report of Naive Bayes

	precision	recall	f1-score	support
0	0.95	0.97	0.96	90
1	0.94	0.91	0.92	53
accuracy			0.94	143
macro avg	0.94	0.94	0.94	143
weighted avg	0.94	0.94	0.94	143

Accuracy Score: 0.9440559440559441

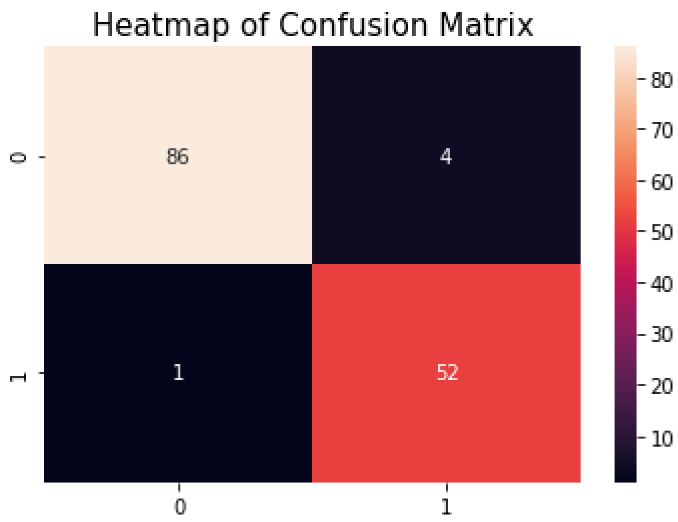
MODEL 5: RANDOM FOREST

Model: Random Forest

[[86 4]

 [1 52]]

Testing Accuracy= 0.9440559440559441



Classification Report of Random Forest

	precision	recall	f1-score	support
0	0.99	0.96	0.97	90
1	0.93	0.98	0.95	53
accuracy			0.97	143
macro avg	0.96	0.97	0.96	143
weighted avg	0.97	0.97	0.97	143

Accuracy Score: 0.9440559440559441

CONFUSION MATRICES

Confusion Matrix of KNN:

Training Set Testing State

[[265 2] [[89 1]

[8 151]] [6 47]]

Confusion Matrix of Support Vector Machine:

Training Set Testing State

[[265 2] [[88 2]

[5 154]] [3 50]]

Confusion Matrix of Logistic Regression:

Training Set Testing State

[[267 0] [[86 4]

[4 155]] [3 50]]

Confusion Matrix of Naive Bayes:

Training Set Testing State

[[262 5] [[87 3]

[16 143]] [5 48]]

Confusion Matrix of Random Forest:

Training Set Testing State

[[267 0] [[84 6]

[0 159]] [1 52]]

CONCLUSION

Breast cancer if found at an early stage will help save the lives of thousands of women or even men. These projects help the real world patients and doctors to gather as much information as they can to predict and detect Breast Cancer early. On applying the above Machine Learning Classification techniques, we found that the technique with highest Training Set Accuracy is **Random Forest with Accuracy= 0.9953051643192489**. However, the maximum Testing Set Accuracy is of **Support Vector Machine= 0.9790209790209791**. Therefore, the most suitable Machine Learning Classification Model used for prediction of Breast Cancer is Support Vector Machine.