COEN 240 Final Project

BREAST CANCER PREDICTION

Pooja Gona - W1650024

Abstract:

The aim of this study is to discover the key attributes that are effective in forecasting whether cancer is benign or malignant, as well as identifying general patterns that may assist in selecting appropriate models and hyperparameters. The goal is to classify and ascertain whether breast cancer is benign or malignant. To achieve this, a machine learning classification algorithm was used to train a function that can accurately predict the categorical class of new input data.

Introduction to Problem:

Breast cancer (BC) is a critical public health issue globally, with majority of the latest cancer cases and deaths due to cancer among women. Early detection of BC is vital for improving patients' chances of survival and prognosis by enabling timely clinical intervention. Accurate classification of benign tumors can prevent patients from unnecessary procedures, making the proper diagnosis of BC and patient classification into benign or malignant groups a subject of extensive research. Machine learning (ML) is widely preferred for BC pattern classification and forecast modeling due to its ability to uncover important features from complex datasets. Data classification techniques, such as data mining, are effective in various fields, including the medical industry, where they are frequently used for diagnosis and analysis to reach conclusions.

Data Set Description:

The data set has the following attributes in the dataset.

MEAN_PERIMETER

MEAN_RADIUS

DIAGNOSIS

MEAN_AREA

MEAN_SMOOTHNESS

MEAN_TEXTURE

Snapshot of Dataset Used:

mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
17.99	10.38	122.8	1001	0.1184	0
20.57	17.77	132.9	1326	0.08474	0
19.69	21.25	130	1203	0.1096	0
11.42	20.38	77.58	386.1	0.1425	0
20.29	14.34	135.1	1297	0.1003	0
12.45	15.7	82.57	477.1	0.1278	0
18.25	19.98	119.6	1040	0.09463	0
13.71	20.83	90.2	577.9	0.1189	0
13	21.82	87.5	519.8	0.1273	0
12.46	24.04	83.97	475.9	0.1186	0
16.02	23.24	102.7	797.8	0.08206	0
15.78	17.89	103.6	781	0.0971	0
19.17	24.8	132.4	1123	0.0974	0
15.85	23.95	103.7	782.7	0.08401	0
13.73	22.61	93.6	578.3	0.1131	0
14.54	27.54	96.73	658.8	0.1139	0
14.68	20.13	94.74	684.5	0.09867	0
16.13	20.68	108.1	798.8	0.117	0
19.81	22.15	130	1260	0.09831	0

Data Pre-Processing Techniques:

Data preprocessing is a critical phase in the data mining process that involves modifying or eliminating data prior to usage to ensure or enhance performance. This is especially important in projects related to data mining and machine learning, as the quality of the data can significantly affect the accuracy of the results. The methods used to collect data are often not strictly controlled, leading to missing values, values that fall outside of the expected range, and unrealistic combinations of data. Failure to thoroughly check for these issues in the data analysis process can result in erroneous results. As a result, before any analysis is conducted, it is essential to prioritize the quality and representation of the data. In computational biology, data preprocessing is frequently the most crucial stage of a machine learning project. The goal of data preprocessing is to modify and standardize the dataset using various methods so that it can be effectively used by the machine learning model.

- **Dropping unwanted columns**: Our dataset contains a column with no values, which, if used, could result in errors. This column is therefore removed from the dataset.
- **Encoding**: The categorical value (diagnosis) from the sklearn pre-processing module is encoded in this pre-processing step using Label Encoder.
- **Separation of test and train data**: Test data are required to validate our model. Therefore, using the train test split module, the data set is split into train and test data.
- **Standard scalarization**: It is used to transform and fit the train data.

Classification that can be used:

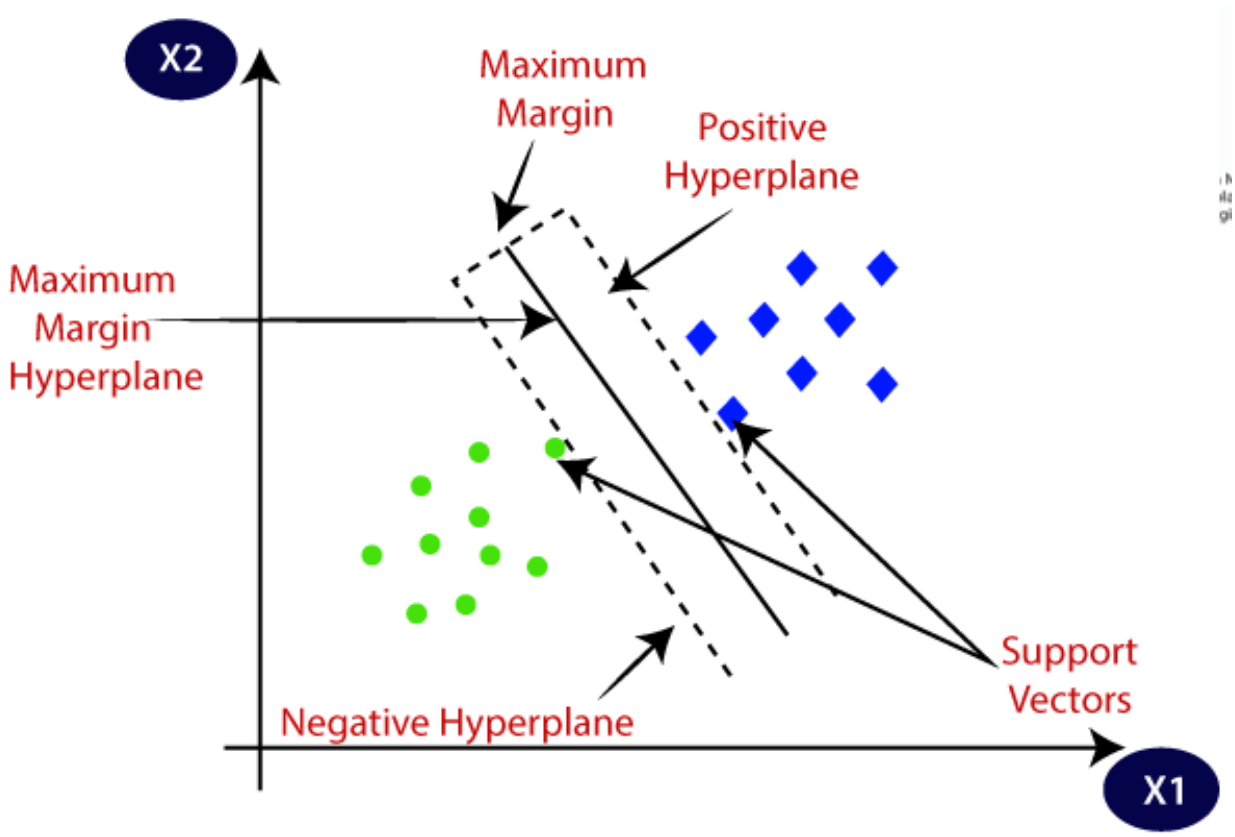
1. Support Vector Machines
2. Random Forests
3. Logistic Regression
4. K-Nearest Neighbour

Algorithm Used - SVM :

A supervised machine learning approach called the Support Vector Machine (SVM) can be used for classification, regression, and outlier detection. The effective SVM algorithm locates the hyperplane that best distinguishes between various classes of data points.

The decision boundary dividing data points with various class labels is the hyperplane. The hyperplane is a line in two-dimensional space and changes to a hyperplane in higher dimensions.

By increasing the margin, or the separation between the hyperplane and the nearest data points for each class, the SVM method determines the ideal hyperplane. Support vectors are the closest data points, hence the name Support Vector Machine.



Using a kernel function, SVM can separate data that is linearly separable and non-linearly separable. The original feature space is converted into a higher-dimensional feature space using the kernel function, increasing the likelihood that the data points can be separated linearly.

The flexibility with which SVM can handle high-dimensional data and its resistance to overfitting are only two of its many benefits. It does have certain drawbacks, though, namely the requirement for a wise choice of kernel function and the challenge of interpreting the outcomes.

In terms of machine learning and data science, SVM is a potent and well-known algorithm.

Advantages of SVM:

- Effective in high-dimensional areas: SVMs are a suitable option for applications requiring a lot of features since they are effective in high-dimensional spaces.

- SVMs have a strong ability to generalize, which enables them to generate precise predictions on previously unexplored data.

- SVMs can employ the kernel method to shift the data into a higher-dimensional space, which makes it simpler to separate classes that aren't separable linearly.

- SVMs are less prone to overfitting than other machine learning algorithms, especially when the kernel method is utilized, making them more robust to overfitting.

- Versatility: SVMs can be applied to a variety of tasks, such as text classification, picture recognition, and bioinformatics.

- SVMs are a sparse solution

Disadvantages of SVM :

- Noise sensitivity: SVM is susceptible to noisy data, which can result in overfitting.

- Selection of kernel: The SVM algorithm's performance can be significantly impacted by the kernel function that is selected. Unfortunately, it is not always easy and requires knowledge to choose the best kernel function.

- Restricted scalability: SVM's scalability may be restricted by the cost of computation, particularly when dealing with huge datasets.

- SVM is a discriminative method that produces class labels based on decision boundaries, but it does not offer probabilistic estimates of class membership.

- Selection of numerous parameters, including the cost function, regularization parameter, and kernel function, is crucial for SVM performance but can be challenging.

Code Snapshots:

In [5]:
```
dataset.drop(dataset.columns[dataset.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)
```

In [6]:
```
dataset.head()
```

Out[6]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

In [7]:
```
x=dataset.drop(['diagnosis'],axis=1)
y=dataset['diagnosis']
```

In [8]:
```
y.head()
```

Out[8]:
```
0    0
1    0
2    0
3    0
4    0
Name: diagnosis, dtype: int64
```

In [9]:
```
x.head()
```

Out[9]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness
0	17.99	10.38	122.80	1001.0	0.11840
1	20.57	17.77	132.90	1326.0	0.08474
2	19.69	21.25	130.00	1203.0	0.10960
3	11.42	20.38	77.58	386.1	0.14250
4	20.29	14.34	135.10	1297.0	0.10030

In [10]:
```
dataset.nunique()
```

Out[10]:
```
mean_radius         456
mean_texture        479
mean_perimeter      522
mean_area           539
mean_smoothness     474
diagnosis             2
dtype: int64
```

In [11]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
y = labelencoder_Y.fit_transform(y)
```

In [12]:
```
y.shape
print(y)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1]
```

In [13]:
```
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)
```

In [14]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
```

```
[[-0.65079907 -0.43057322 -0.68024847 -0.62698309 -0.91381897]
 [-0.82835341  0.15226547 -0.82773762 -0.75309358  0.65281216]
 [ 1.68277234  2.18977235  1.60009756  1.67383892  0.10362413]
 ...
 [-1.33114223 -0.22172269 -1.3242844  -1.05503654  0.32763504]
 [-1.25110186 -0.24600763 -1.28700242 -1.02864778 -1.94137868]
 [-0.74662205  1.14066273 -0.72203706 -0.7080938  -0.27141349]]
```

In [15]:
```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, Y_train)
```

Out[15]:
```
SVC(kernel='linear', random_state=0)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [16]:
```
Y_pred = classifier.predict(X_test)
```

In [17]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
```

OUTPUT:

In [22]:
```
datapredicted=pd.DataFrame(cm,index=['Cancer','Healthy'],columns=['Predicted_Cancer','Predicted_Healthy'])
datapredicted
```

Out[22]:

	Predicted_Cancer	Predicted_Healthy
Cancer	47	6
Healthy	5	85

```
In [20]: from sklearn.metrics import classification_report
         matrix=classification_report(Y_test,Y_pred)

In [23]: print(matrix)
```

	precision	recall	f1-score	support
0	0.90	0.89	0.90	53
1	0.93	0.94	0.94	90
accuracy			0.92	143
macro avg	0.92	0.92	0.92	143
weighted avg	0.92	0.92	0.92	143

Conclusion :

- The suggested machine-learning algorithms could detect breast cancer since effective therapeutic interventions could slow down the disease progress and reduce death rates by making an early diagnosis of this condition.
- Modeling performance may be enhanced by using various machine learning techniques, having access to larger datasets from several institutions (multi-center study), and taking into account important characteristics from a number of pertinent data sources.

References :

1.https://www.indianjournals.com/ijor.aspx?target=ijor:jicse&volume=9&issue=1&article=006

2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6610104/

3. https://www.semanticscholar.org/paper/Comparison-of-Classification-Models-for-Early-of-Ghani-Alam/fafe8be39d18668bbeeac2c9b4279f0bd3833801

4. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4764253/

5. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7308891/

6. https://aapm.onlinelibrary.wiley.com/doi/full/10.1002/mp.13886

7. https://europepmc.org/article/pmc/pmc7609380

8. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5217832/