Breast Cancer Report

Inuwa Amir Usman

December 10, 2021

Abstract

Your abstract.

1 Introduction

Data is all around us. Data is information that has been translated into a form that is efficient for movement or processing. Relative to today's computers and transmission media, data is information converted into binary digital form. Any device or technology you use it most likely uses or has data to process. Most industries use data to make and predict their future decisions. Industries and companies use data to stay ahead of their competition, with Data comes trust and proof. In this report we will discuss the Important algorithms used for clustering, classification, modelling and pattern mining.

2 Data Set information

Given the data set is about Breast Cancer. The data set contains 9 columns + the class attribute which makes it 10 columns in total, and 286 rows, it contains Categorical attributes.

3 Cleaning and Preparing the Data

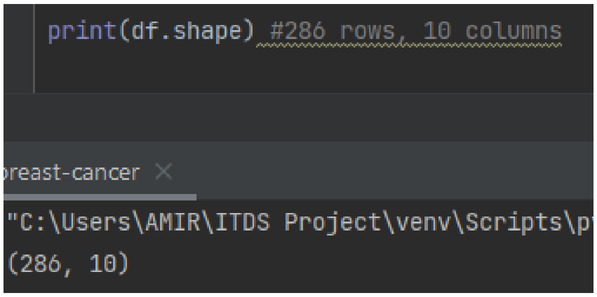
In this section well start about how we went through the steps and process of cleaning and preparing the given data set.

```
headers=["Class", "age", "menopause", "tumor-size", "inv-nodes",
         "node-caps", "deg-malig", "breast", "breast-quad", "irradiat"]
df = pd.read_csv("breast-cancer.data", names=headers)
print(df.head())
```

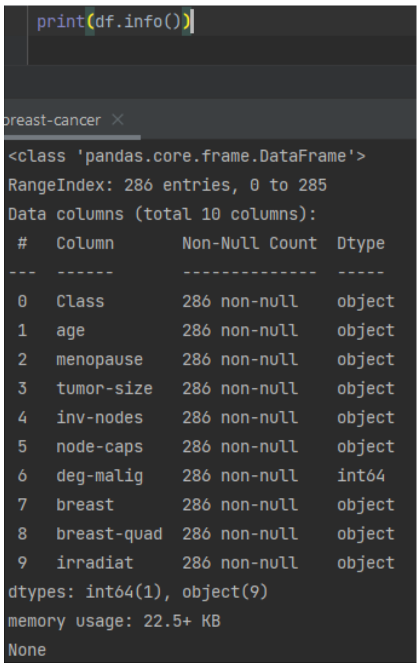
```
reast-cancer
                  Class    age menopause  ... breast breast-quad irradiat
0  no-recurrence-events  30-39   premeno  ...   left    left_low       no
1  no-recurrence-events  40-49   premeno  ...  right    right_up       no
2  no-recurrence-events  40-49   premeno  ...   left    left_low       no
3  no-recurrence-events  60-69      ge40  ...  right     left_up       no
4  no-recurrence-events  40-49   premeno  ...  right   right_low       no

[5 rows x 10 columns]
```

For the first step we went about creating headers for the data set as it didn't have one, we used the description to get the names of the headers and made a row of headers called "headers", this is so we don't have the header clashing with our index 0 of our data set. As you can see in the figure above we have successfully added the header without losing any important data.

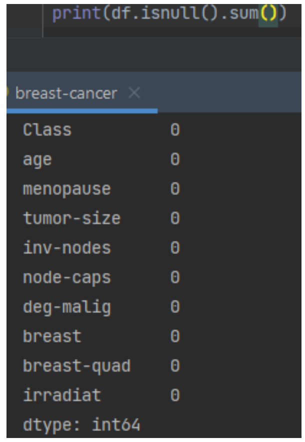
After that we loaded our dataframe and added the header variable into it and ran df.head() to make sure our implementation was correct.



The next step was checking the shape of our data to see what we are working with using df.shape and the result was accurate with 286 rows and 10 columns.



As shown in the figure, we noticed we have 1 integer and 9 objects and no null values, and after that we ran "df.duplicated().any()" to check if there are duplicated features to be dropped from the dataframe. and the result was = True Dropped the duplicates we found from above using df = df.drop_duplicates() in this case 14 duplicated rows have been dropped and our new dataset now has (272 rows, 10 columns)



```
print(df.isnull().sum())

breast-cancer
Class          0
age            0
menopause      0
tumor-size     0
inv-nodes      0
node-caps      0
deg-malig      0
breast         0
breast-quad    0
irradiat       0
dtype: int64
```

Figure 1: Class

Verified if there are any null values and no null values were detected df.isnull().sum()

4 Data Visualization

To visualize the data seaborn was used to display the data, first looked into object variables then implemented a for loop to go through the object, Sns count was used to count the number of observations per category for categories variable, get the tick location and show the plot result.

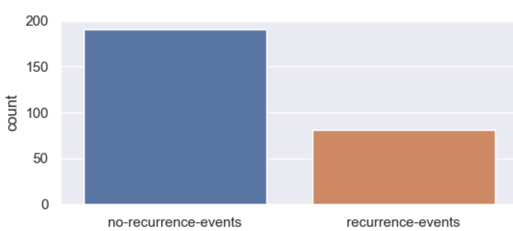


Figure 2: Class

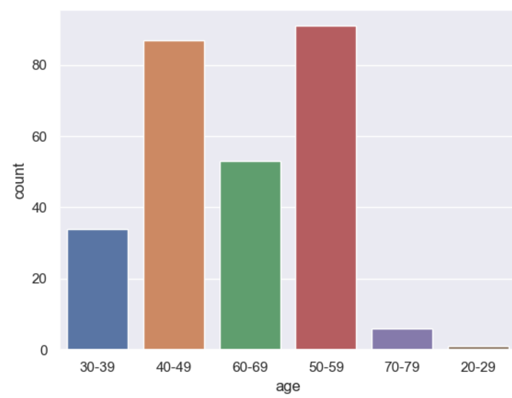


Figure 3: Age

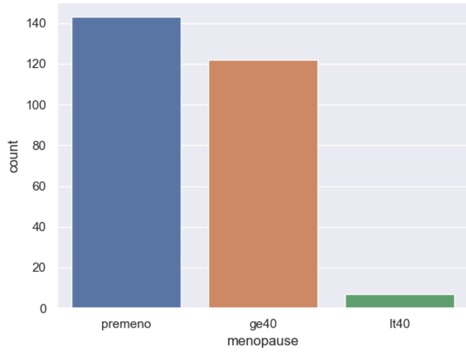


Figure 4: Menopause

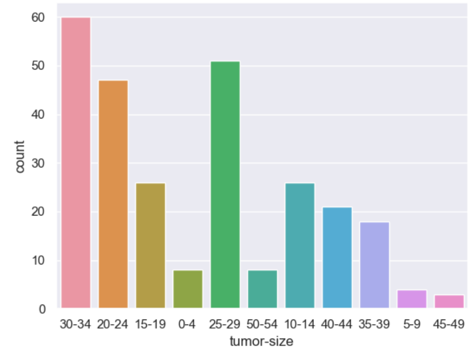


Figure 5: Tumor-size

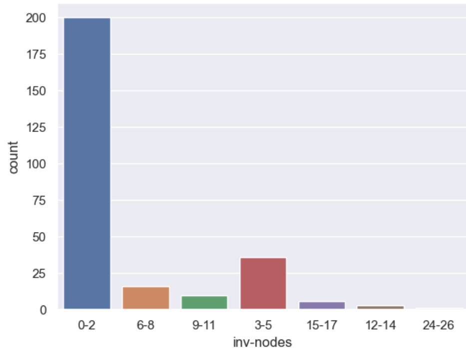


Figure 6: Inv-nodes

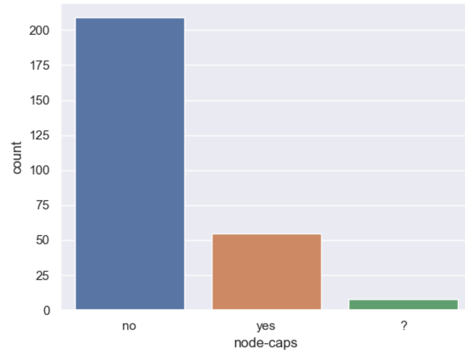


Figure 7: Node-caps

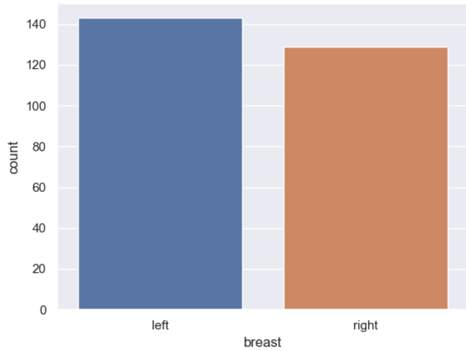


Figure 8: Breast

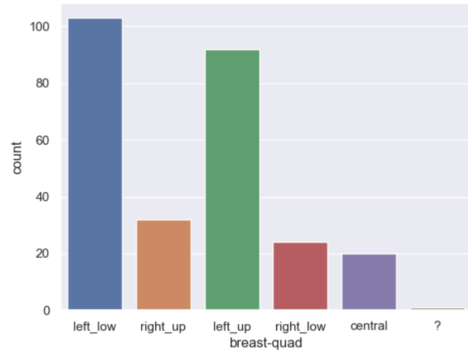


Figure 9: Breast-quad

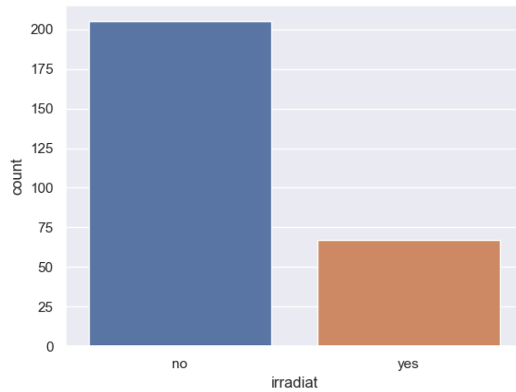
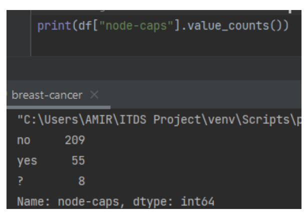


Figure 10: Irradiant

5 Data Preprocessing

As noticed before from the dataframe and description it was mentioned that there are missing values in the dataframe with the "?" value. So first searched for the column where those missing values are with and found that it is in the "node-caps" feature. so handle it by:



```
print(df["node-caps"].value_counts())
```

```
breast-cancer
"C:\Users\AMIR\ITDS Project\venv\Scripts\p
no     209
yes     55
?        8
Name: node-caps, dtype: int64
```

Figure 11: run df["node-caps"].value_counts())

The result was 8 rows containing the "?" missing value, handle it by dropping the missing value rows. df = df.loc[df['node-caps'] != '?']

6 One-Hot Encoding (Dummy Encoding)

In this part we begin our encoding since we have Cleaned and prepared our data perfectly, we chose One-Hot Encoding/Dummy Encoding to create a binary column for each category and return a sparse matrix or dense array, this was picked mainly because of the "0-10,20-30" type of string data we have in our data frame so we get an accurate result of each row, and then we changed all the strings into integers except the "Class" instance which we will do later.

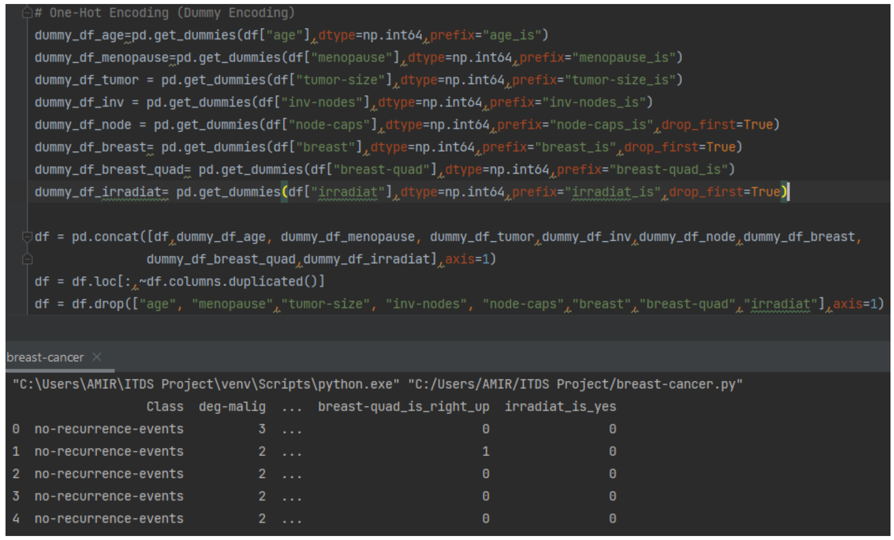


Figure 12: Dummy encoding

After the encoding we check to see what our new dataframe looks like
print(df.head())
print(df.columns)
df.info()

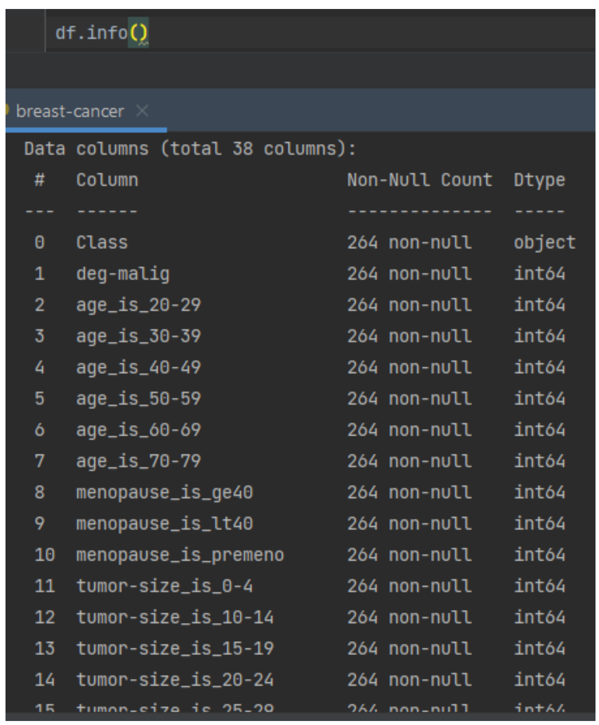


Figure 13: df.info

```
print(df.columns)
```

```
breast-cancer
Index(['Class', 'deg-malig', 'age_is_20-29', 'age_is_30-39', 'age_is_40-49',
       'age_is_50-59', 'age_is_60-69', 'age_is_70-79', 'menopause_is_ge40',
       'menopause_is_lt40', 'menopause_is_premeno', 'tumor-size_is_0-4',
       'tumor-size_is_10-14', 'tumor-size_is_15-19', 'tumor-size_is_20-24',
       'tumor-size_is_25-29', 'tumor-size_is_30-34', 'tumor-size_is_35-39',
       'tumor-size_is_40-44', 'tumor-size_is_45-49', 'tumor-size_is_5-9',
       'tumor-size_is_50-54', 'inv-nodes_is_0-2', 'inv-nodes_is_12-14',
       'inv-nodes_is_15-17', 'inv-nodes_is_24-26', 'inv-nodes_is_3-5',
       'inv-nodes_is_6-8', 'inv-nodes_is_9-11', 'node-caps_is_yes',
       'breast_is_right', 'breast-quad_is_?', 'breast-quad_is_central',
       'breast-quad_is_left_low', 'breast-quad_is_left_up',
       'breast-quad_is_right_low', 'breast-quad_is_right_up',
       'irradiat_is_yes'],
      dtype='object')
```

Figure 14: df.columns

last but not the least we transform the "Class" features into 1's and 0's to complete our encoding. This can be done easily by.

```
# Transforming "Class" features into 1's and 0's
df["Class"].value_counts()
df['Class'] = df['Class'].map({'recurrence-events': 1, 'no-recurrence-events': 0}).astype(int)
print(df["Class"].value_counts())
```

```
breast-cancer
"C:\Users\AMIR\ITDS Project\venv\Scripts\python.exe" "C:/Users/AMIR/ITDS Project/breast-cancer.py"
0    186
1     78
Name: Class, dtype: int64
```

Figure 15: Class to int

7 Modeling and Classifying our Dataframe

7.1 Train-Test Split

When classification problems exhibit a significant imbalance in the distribution of the target classes, it is good to use stratified sampling to ensure that relative class frequencies are approximately preserved in train and test sets.

```
y=df["Class"] #output column
X=df.drop(["Class"],axis=1) #train columns
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.30,random_state=123) # 0.3 testing, 0.7 training
```

Figure 16: training

First we implement our output column from our data frame which is "class", then we use that column and train it with a given axis.

7.2 Logistic Regression Classifier

	precision	recall	f1-score	support
0	0.69	0.89	0.78	53
1	0.50	0.22	0.31	27
accuracy			0.66	80
macro avg	0.60	0.55	0.54	80
weighted avg	0.63	0.66	0.62	80

```
AxesSubplot(0.125,0.125;0.62x0.755)
```

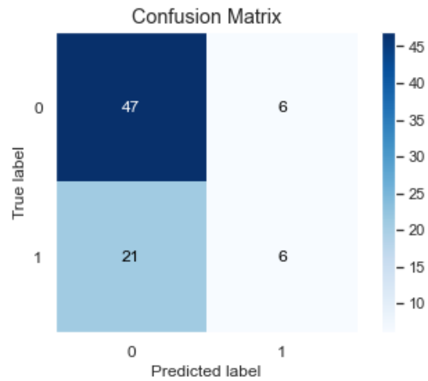


Figure 17: LRC

7.3 Naive Bayes Classifier

	precision	recall	f1-score	support
0	0.71	0.83	0.77	53
1	0.50	0.33	0.40	27
accuracy			0.66	80
macro avg	0.60	0.58	0.58	80
weighted avg	0.64	0.66	0.64	80

```
AxesSubplot(0.125,0.125;0.62x0.755)
```

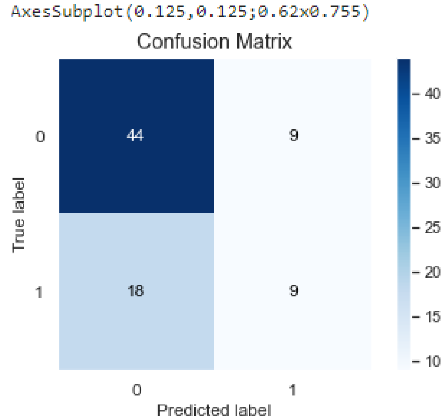


Figure 18: NBC

7.4 Random Forest Classifier

	precision	recall	f1-score	support
0	0.73	0.91	0.81	53
1	0.64	0.33	0.44	27
accuracy			0.71	80
macro avg	0.69	0.62	0.62	80
weighted avg	0.70	0.71	0.68	80

AxesSubplot(0.125,0.125;0.62x0.755)

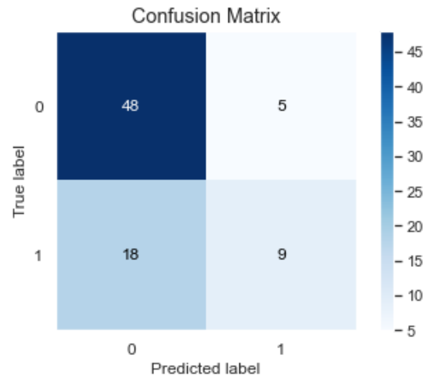


Figure 19: RFC

7.5 Gradient Boosting Classifier

	precision	recall	f1-score	support
0	0.71	0.83	0.77	53
1	0.50	0.33	0.40	27
accuracy			0.66	80
macro avg	0.60	0.58	0.58	80
weighted avg	0.64	0.66	0.64	80

AxesSubplot(0.125,0.125;0.62x0.755)

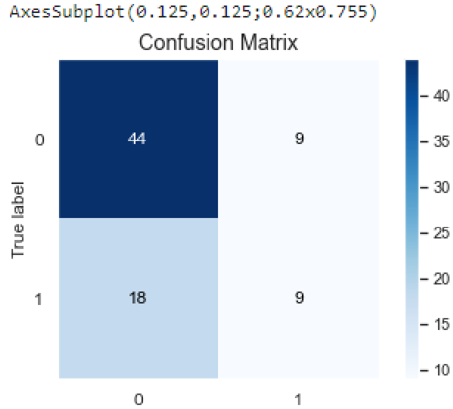


Figure 20: GBC

8 Model Performance Comparison and Conclusions

According to the dataset that we have, Logistic Regression performed much better than Naive Bayes. Gradient Boosting Classifier performed the best among the others.

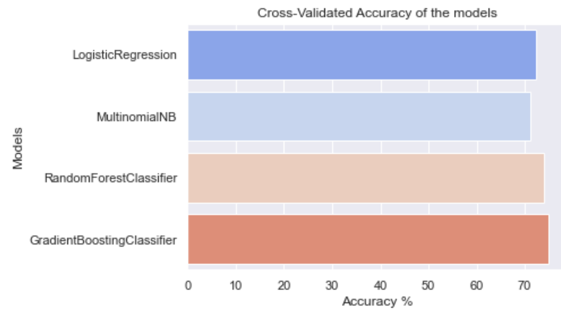


Figure 21: accuracy

	Models	Accuracy
0	LogisticRegression	72.3
0	MultinomialNB	71.2
0	RandomForestClassifier	73.9
0	GradientBoostingClassifier	75.0

Figure 22: results

9 cluster instances with kmeans

```
import sklearn
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
km = KMeans(n_clusters=2)

km_pred = km.fit_predict(X_test)
print(km_pred)
```

```
east-cancer
"C:\Users\AMIR\ITDS Project\venv\Scripts\python.exe" "C:/Users/AMIR/ITDS Project/breast-ca
[0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0
 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 0 1 1 0
 0 0 1 0 1 1]

Process finished with exit code 0
```

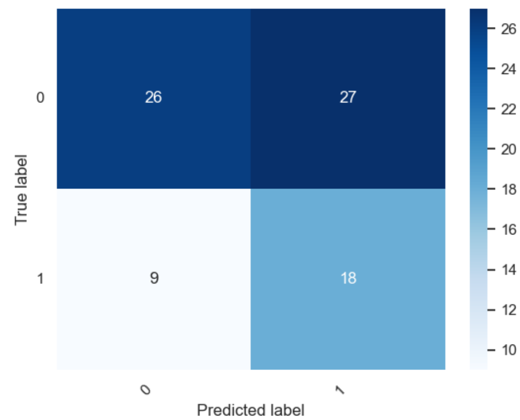


Figure 23: scattered plot