Breast Cancer Dataset Report

Introduction:

The breast cancer dataset is a well-known and frequently used dataset in machine learning and data analysis. It contains information about breast cancer patients, focusing on their clinical and pathological features. This dataset is commonly employed for binary classification tasks, aiming to predict whether a patient will experience a recurrence of breast cancer or not. In this report, we will provide a brief overview of the dataset and its key features using machine learning algorithms and techniques.

Objective

This report aims to analyze a breast cancer dataset to understand the factors influencing recurrence events in breast cancer patients. The objective is to conduct comprehensive data analysis, including preprocessing, exploratory data analysis (EDA), and applying machine learning algorithms. The report focuses on identifying significant features, patterns, and predictive relationships within the dataset. It discusses data preprocessing methods, handling categorical data, and selecting a suitable machine learning algorithm using grid search. Furthermore, the report explores the application of AutoML libraries. Overall, this analysis provides emphasizes the potential of data analysis and machine learning in addressing this critical health concern.

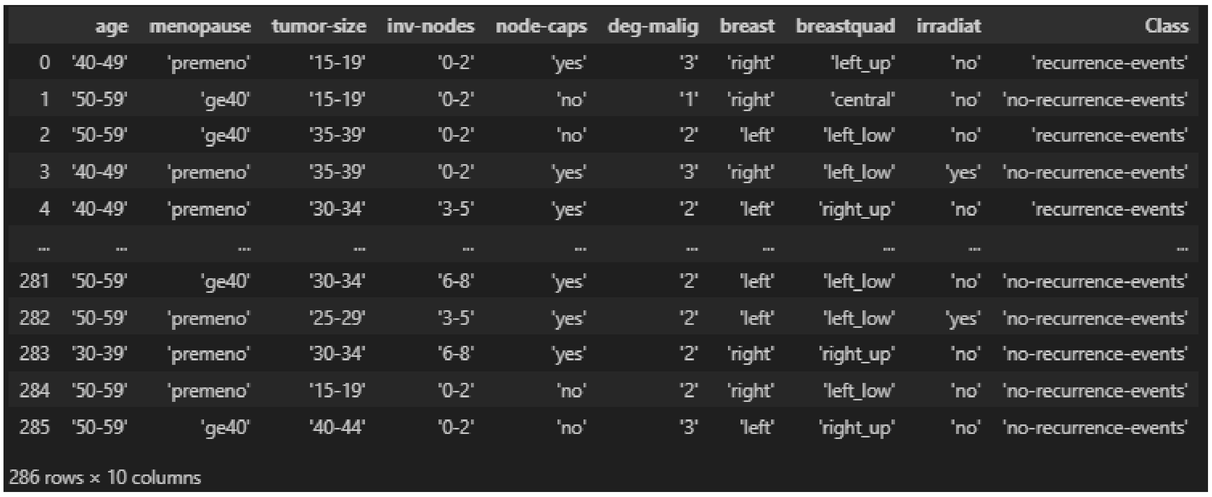
Dataset Description:

The dataset comprises the following columns:

Feature	**Data Type**	**Description**
age	string (default)	The age of the patient in years
menopause	string (default)	The menopausal status of the patient (premeno, peri, or postmenopausal).
tumor-size	string (default)	The size of the tumor.
inv-nodes	string (default)	The number of axillary lymph nodes involved.
node-caps	string (default)	Whether the axillary lymph nodes are encapsulated (yes or no).
deg-malig	number (default)	The degree of malignancy of the tumor.
breast	string (default)	The breast on which the tumor is located (left or right).
breastquad	string (default)	The quadrant within the breast where the tumor is located.
irradiat	string (default)	Whether the patient received radiotherapy (yes or no).
Class	string (default)	The class label indicating the presence or absence of recurrence-events (1 for recurrence, 0 for no recurrence).

As per the instructions given in the assignment, dataset has loaded to the environment.

```
url =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/breast-cancer
.csv'
import pandas as pd
df = pd.read_csv(url, sep=',', header=None)
df.columns =
['age','menopause','tumor-size','inv-nodes','node-caps','deg-malig','breas
t','breastquad','irradiat','Class']
#Preview the dataset
df
```



```
       age  menopause  tumor-size  inv-nodes  node-caps  deg-malig  breast  breastquad  irradiat                  Class
  0  '40-49'  'premeno'    '15-19'      '0-2'      'yes'        '3'  'right'   'left_up'      'no'     'recurrence-events'
  1  '50-59'     'ge40'    '15-19'      '0-2'       'no'        '1'  'right'   'central'      'no'  'no-recurrence-events'
  2  '50-59'     'ge40'    '35-39'      '0-2'       'no'        '2'   'left'  'left_low'      'no'     'recurrence-events'
  3  '40-49'  'premeno'    '35-39'      '0-2'      'yes'        '3'  'right'  'left_low'     'yes'  'no-recurrence-events'
  4  '40-49'  'premeno'    '30-34'      '3-5'      'yes'        '2'   'left'  'right_up'      'no'     'recurrence-events'
...    ...        ...        ...        ...        ...        ...      ...         ...       ...                    ...
281  '50-59'     'ge40'    '30-34'      '6-8'      'yes'        '2'   'left'  'left_low'      'no'  'no-recurrence-events'
282  '50-59'  'premeno'    '25-29'      '3-5'      'yes'        '2'   'left'  'left_low'     'yes'  'no-recurrence-events'
283  '30-39'  'premeno'    '30-34'      '6-8'      'yes'        '2'  'right'  'right_up'      'no'  'no-recurrence-events'
284  '50-59'  'premeno'    '15-19'      '0-2'       'no'        '2'  'right'  'left_low'      'no'  'no-recurrence-events'
285  '50-59'     'ge40'    '40-44'      '0-2'       'no'        '3'   'left'  'right_up'      'no'  'no-recurrence-events'

286 rows × 10 columns
```

As the first step, I assessed the data types in the dataset. Results are as follows.

```
#Preview the data types
print(df.dtypes)
```

Feature	**Data type**
age	object
menopause	object
tumor-size	object
inv-nodes	object
node-caps	object
deg-malig	object
breast	object

breastquad	object
irradiat	object
Class	object

Since the whole dataset has a quotation mark in the parameter data, we removed the symbol in **deg-malig column.**

```
#Remove the Quotation mark in the deg- maling feature
df['deg-malig'] = df['deg-malig'].str.replace("'", '')
```

And then removed the symbol from the entire dataset.

```
# remove the single quotes from the entire dataset
df = df.applymap(lambda x: x.strip("'") if isinstance(x, str) else x)
```

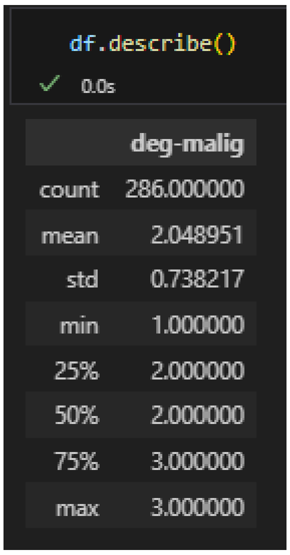
Then converted the deg-malig by pd.to_numeric

```
df['deg-malig']= pd.to_numeric(df['deg-malig'], errors='coerce')
```

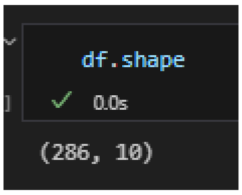
Output as follows

Feature	Data type
age	object
menopause	object
tumor-size	object
inv-nodes	object
node-caps	object
deg-malig	int64
breast	object
breastquad	object
irradiat	object
Class	object

After that I ran df.shape and df.describe() to check out the dimension of the daraset and statistical component of the data set .

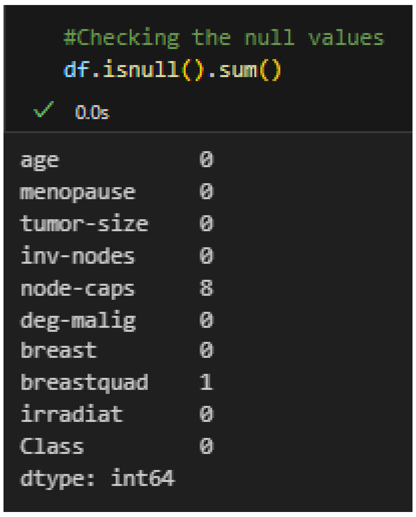


```
df.describe()
✓ 0.0s

       deg-malig
count  286.000000
mean     2.048951
std      0.738217
min      1.000000
25%      2.000000
50%      2.000000
75%      3.000000
max      3.000000
```



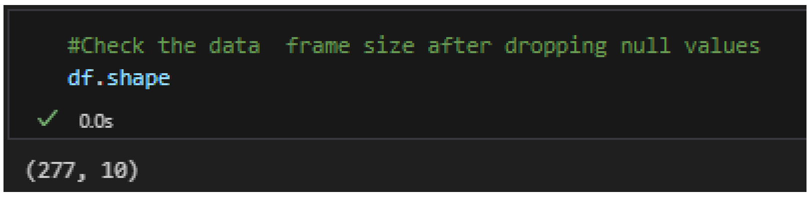
```
df.shape
✓ 0.0s

(286, 10)
```

Data Preprocessing
For this dataset, I have analysed the null values and there were 8 null values from node-caps and 01 null value from breastquad.



```
#Checking the null values
df.isnull().sum()
✓ 0.0s

age           0
menopause     0
tumor-size    0
inv-nodes     0
node-caps     8
deg-malig     0
breast        0
breastquad    1
irradiat      0
Class         0
dtype: int64
```

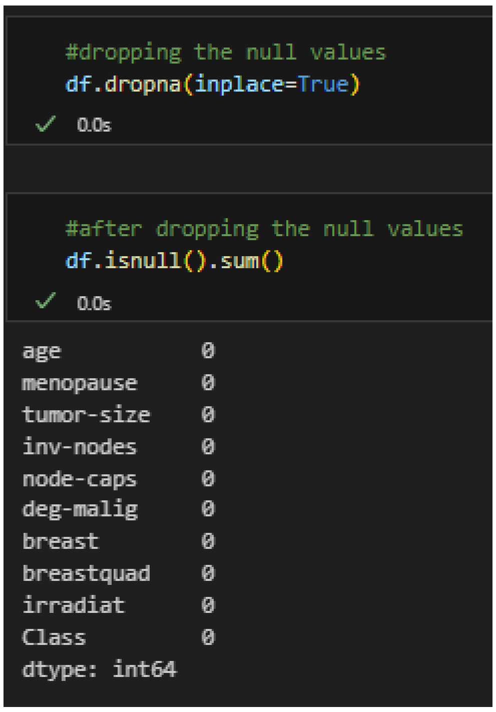
Since there were 9 null values in the dataset, dropped all the null values.



```
#Check the data  frame size after dropping null values
df.shape
✓ 0.0s

(277, 10)
```

After dropping null values, dataset dimension was reduced to (277, 10).

Since we are use encoding methods for machine learning below, did not apply any normalization and standardization for data preprocessing.



```
#dropping the null values
df.dropna(inplace=True)
✓ 0.0s

#after dropping the null values
df.isnull().sum()
✓ 0.0s

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

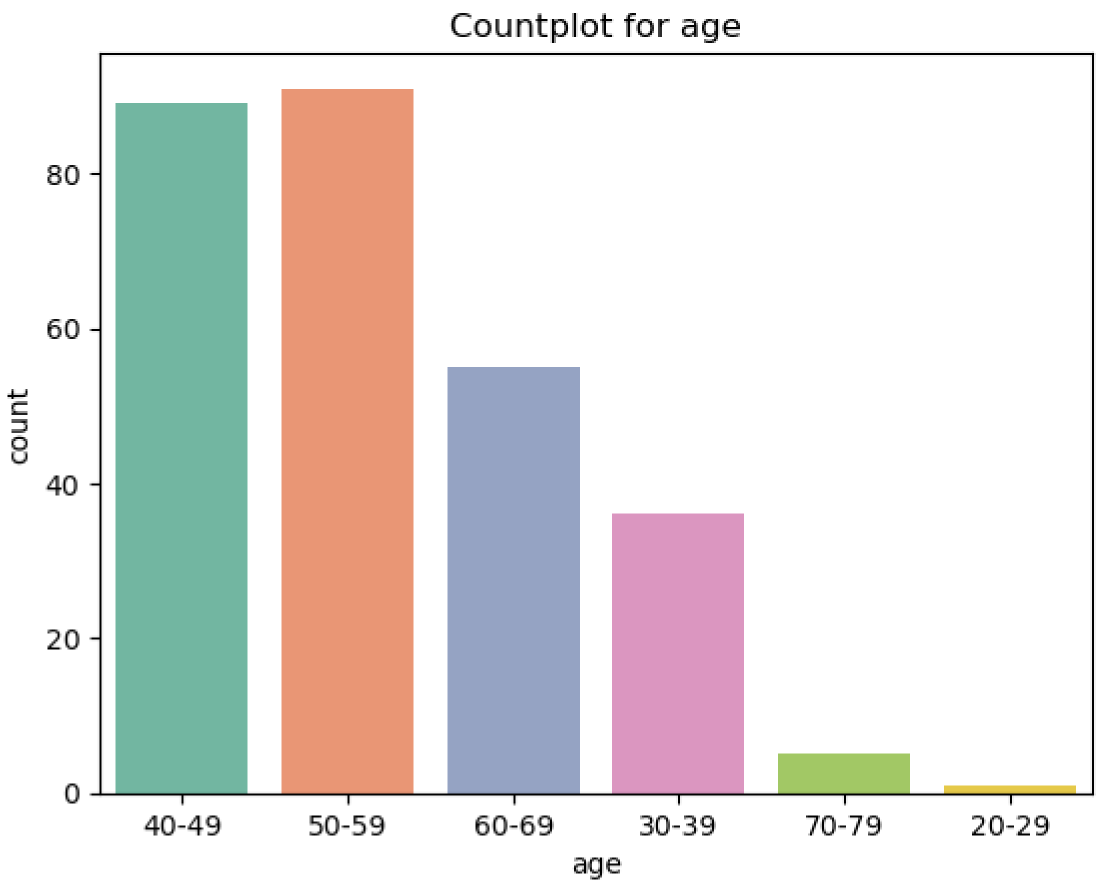
Exploratory Data Analysis and Visualization.

We have analysed all the variables' value counts for each unique value.

Age

The "Age" variable in the breast cancer dataset represents the age of the patient at the time of diagnosis.
The highest number of patients are from the 50-59 & 40-49 age groups.

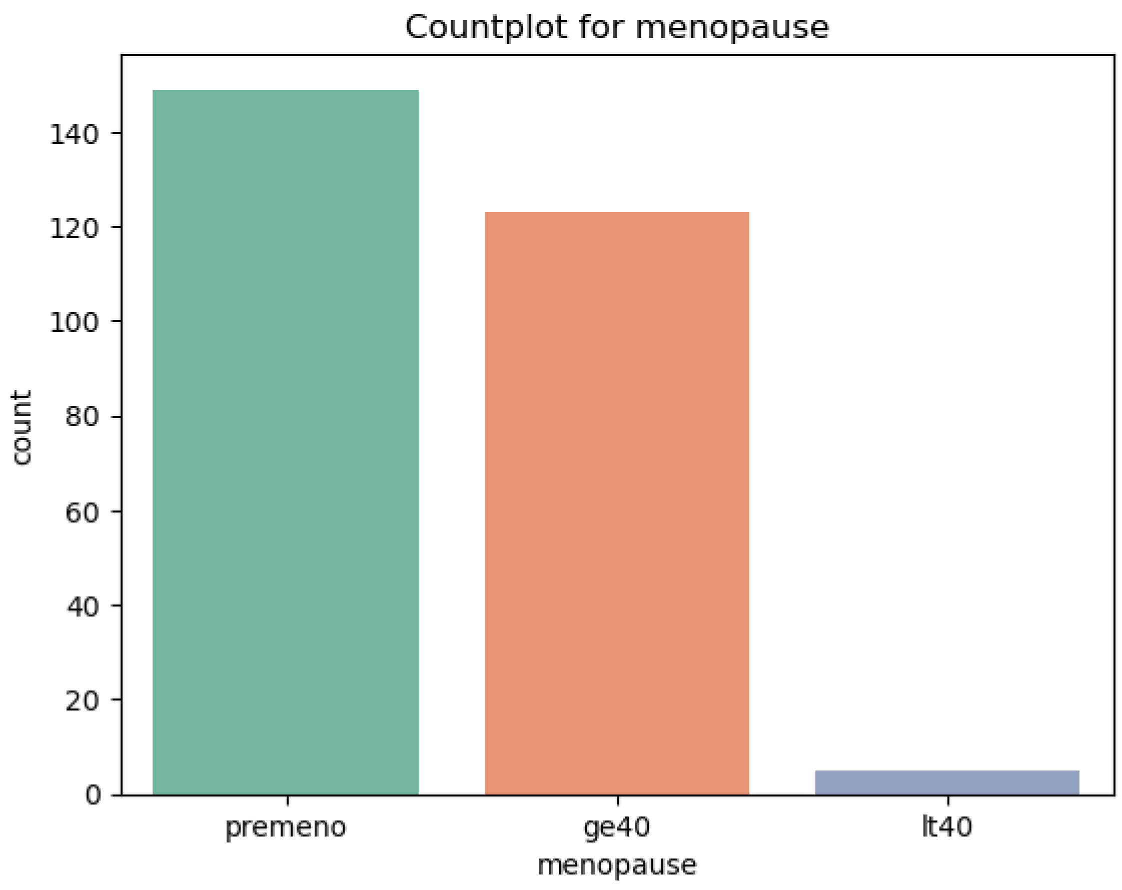
Unique value	**Count**
20-29	1
30-39	36
40-49	89
50-59	91
60-69	55
70-79	5



Menopause

The "Menopause" variable in the breast cancer dataset indicates the menopausal status of the patient. It represents whether the patient is premenopausal, perimenopausal, or postmenopausal.
There are a higher number of patients in premeno stage.

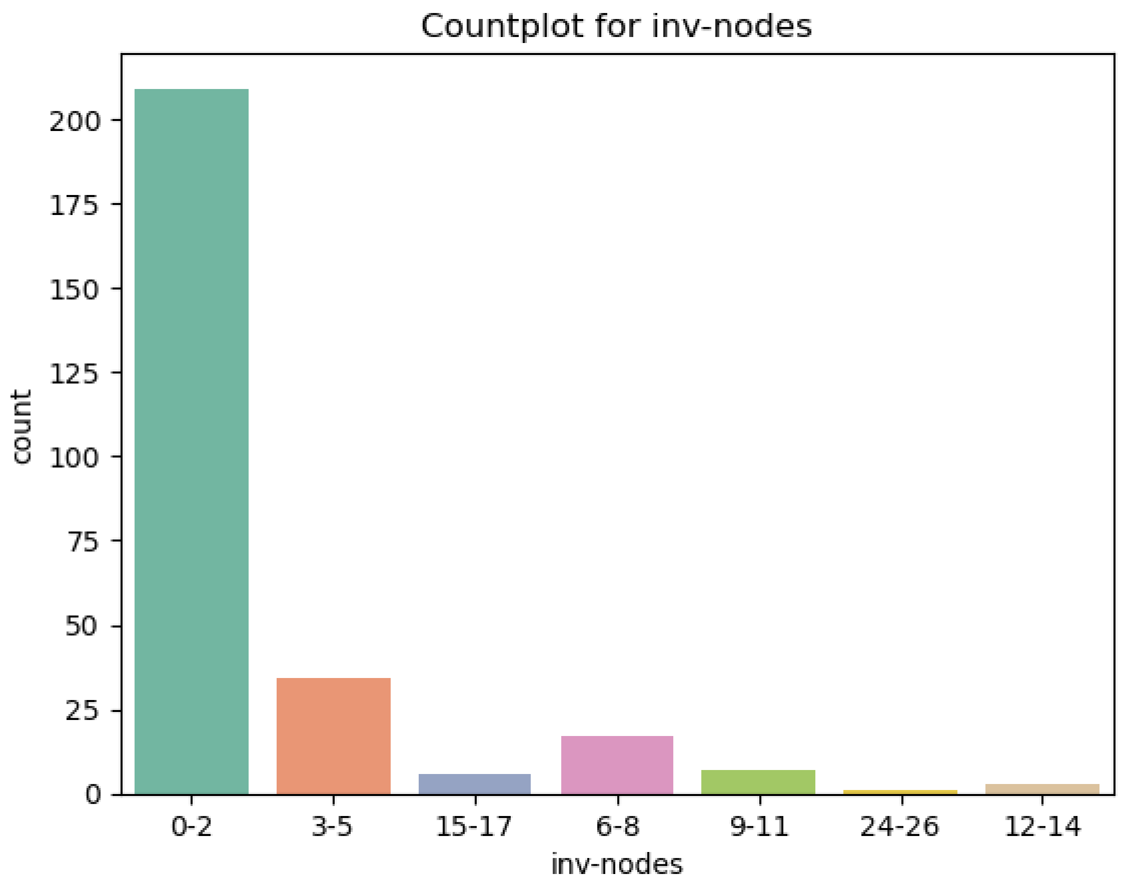
Unique value	**Count**
ge40	123
lt40	5
premeno	149



Inv-nodes

The distribution of lymph node counts can provide insights into the severity or stage of breast cancer and may be useful in prognosis and treatment planning.

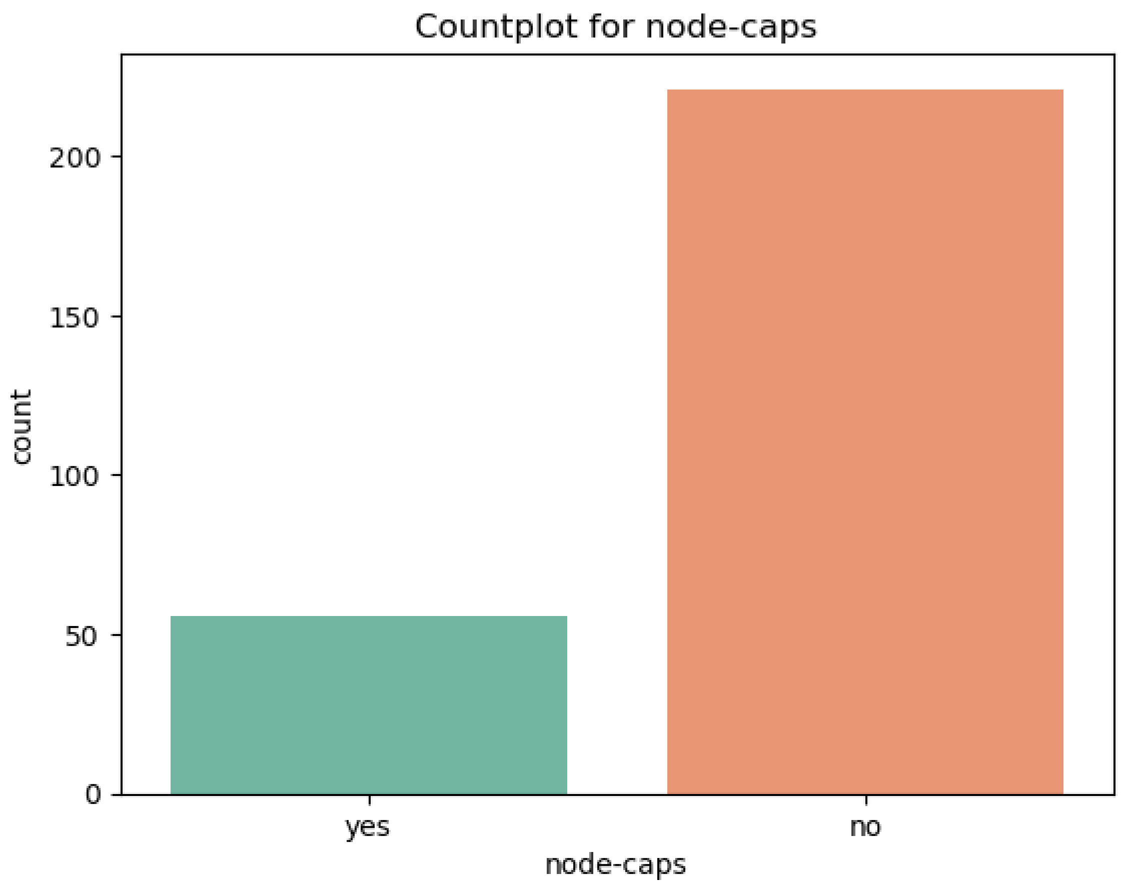
Unique value	**Count**
0-2	209
12-14	3
15-17	6
24-26	1
3-5	34
6-8	17
9-11	7



Node-caps

The "Node-caps" variable in the breast cancer dataset indicates whether the tumor has been detected in the lymph nodes near the breast. It represents the presence or absence of tumor involvement in the regional lymph nodes.
There is a higher count in no nodes-caps.

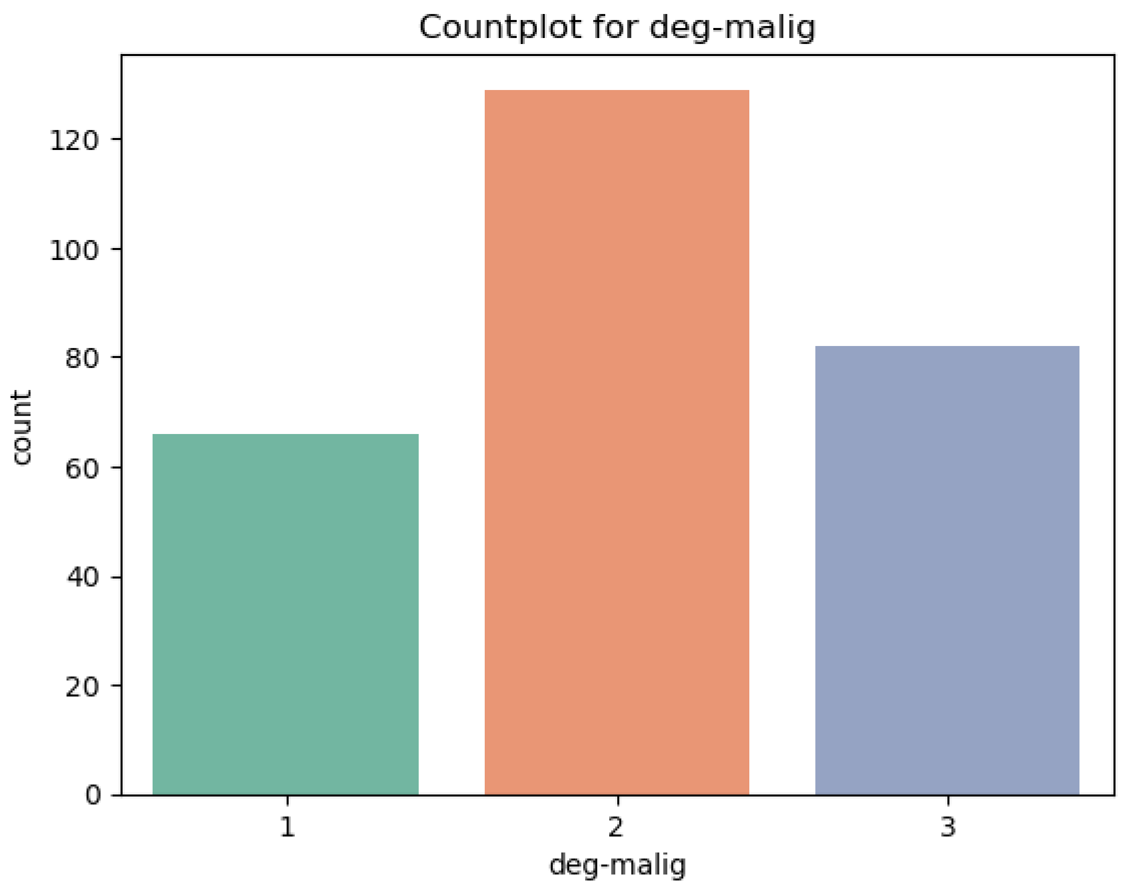
Unique value	**Count**
yes	**56**
no	**221**



Deg-malig

The "Deg-malig" variable in the breast cancer dataset represents the degree of malignancy or the level of tumor aggressiveness. It indicates the severity or aggressiveness of the breast tumor.

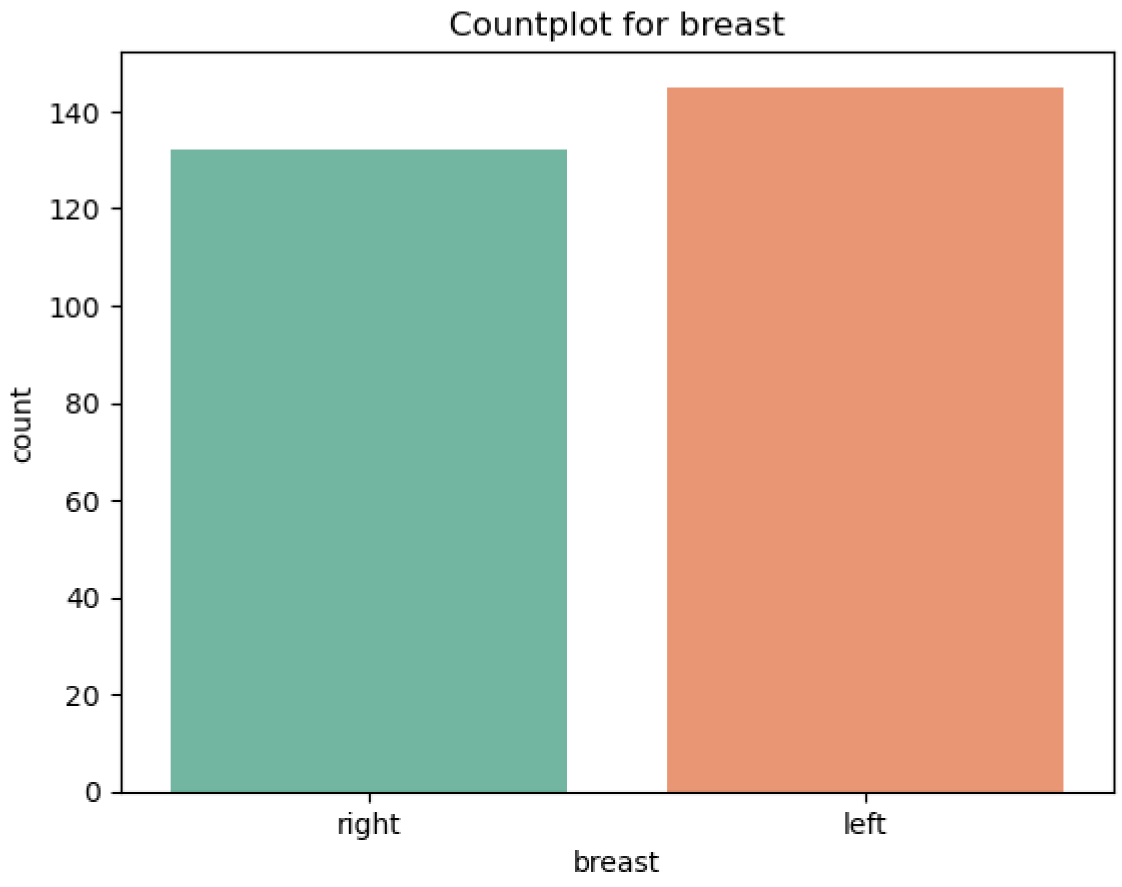
Unique value	**Count**
1	**66**
2	**129**
3	**82**



Breast

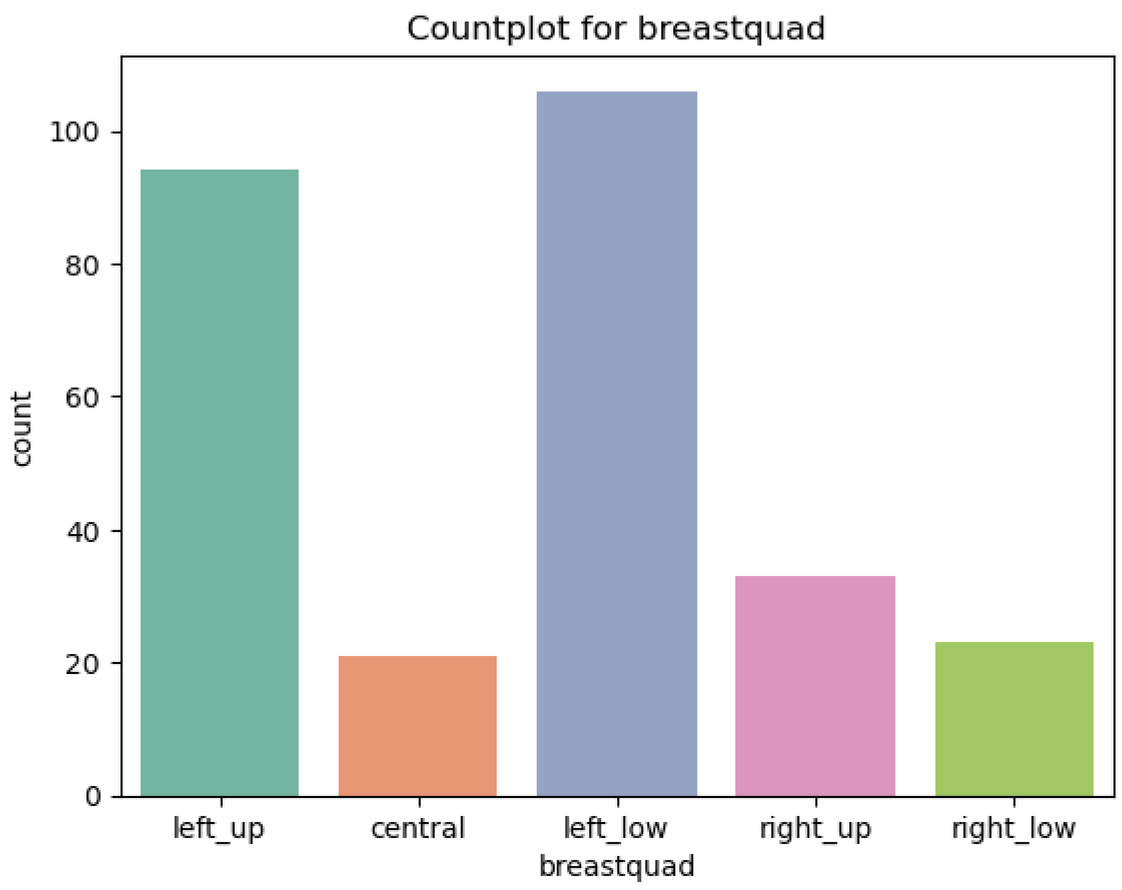
The "Breast" variable in the breast cancer dataset refers to the breast side affected by the tumor. It indicates whether the tumor is present in the left or right breast.

Unique value	**Count**
Left	**145**
Right	**132**



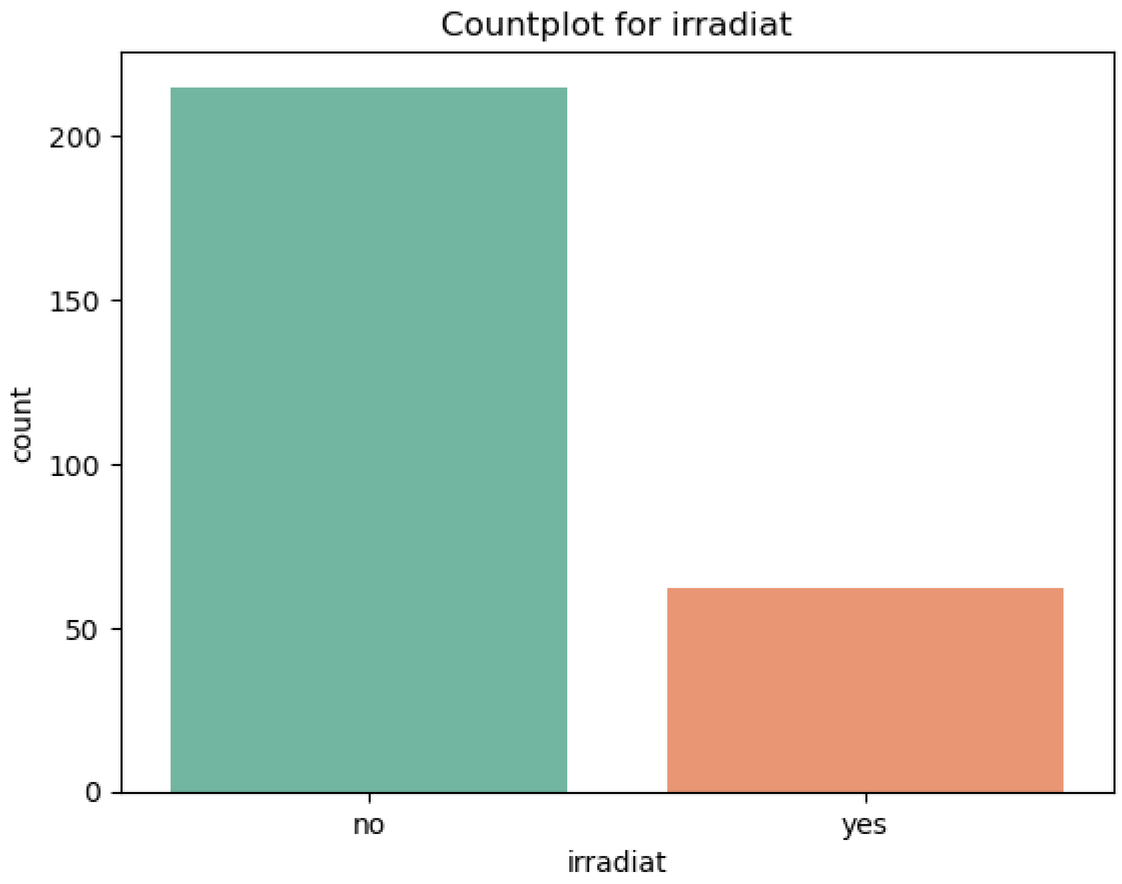
Breastquad
The "Breastquad" variable in the breast cancer dataset represents the quadrant of the breast where the tumor is located. It provides information about the specific region of the breast affected by the tumor.

Unique value	**Count**
central	**21**
left_low	**106**
left_up	**94**
right_low	**23**
right_up	**33**



Irradiat
The "irradiat" variable represents whether or not radiation therapy was administered to the breast cancer patients.

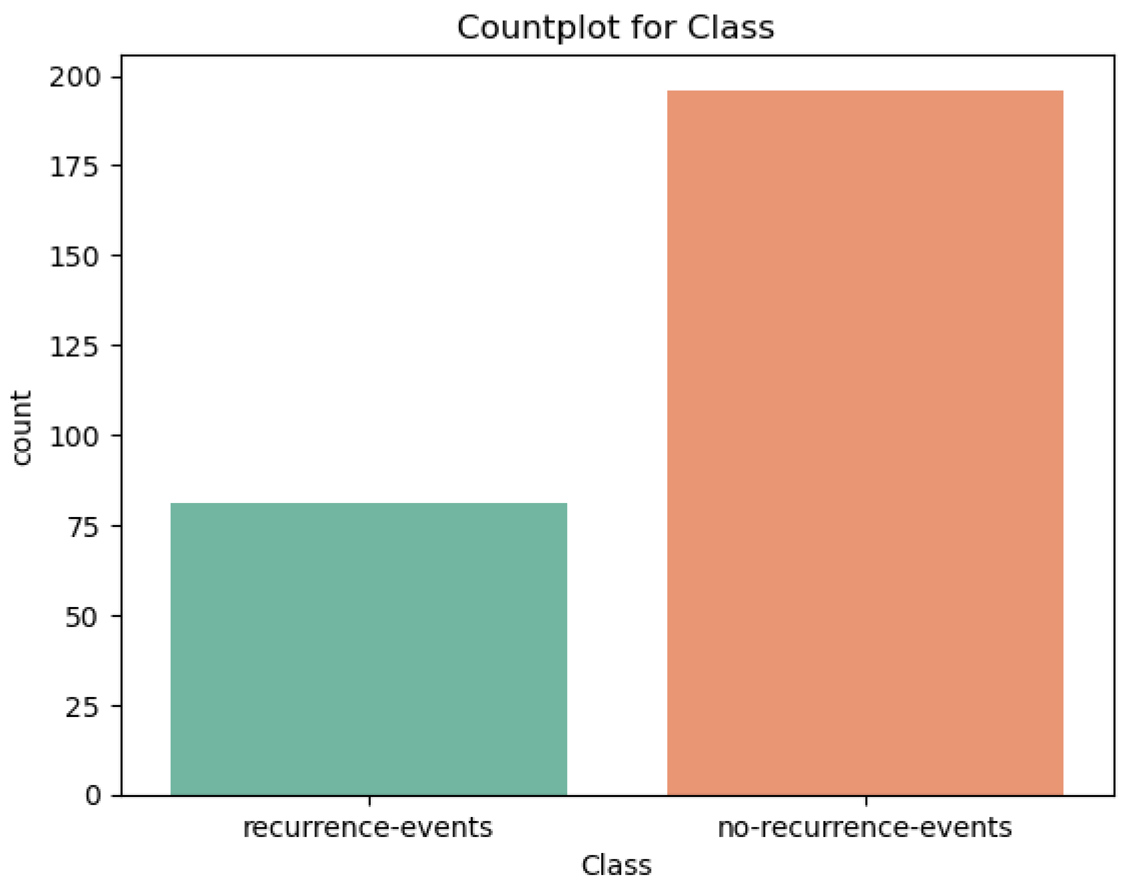
Unique value	**Count**
yes	**62**
no	**215**



Class

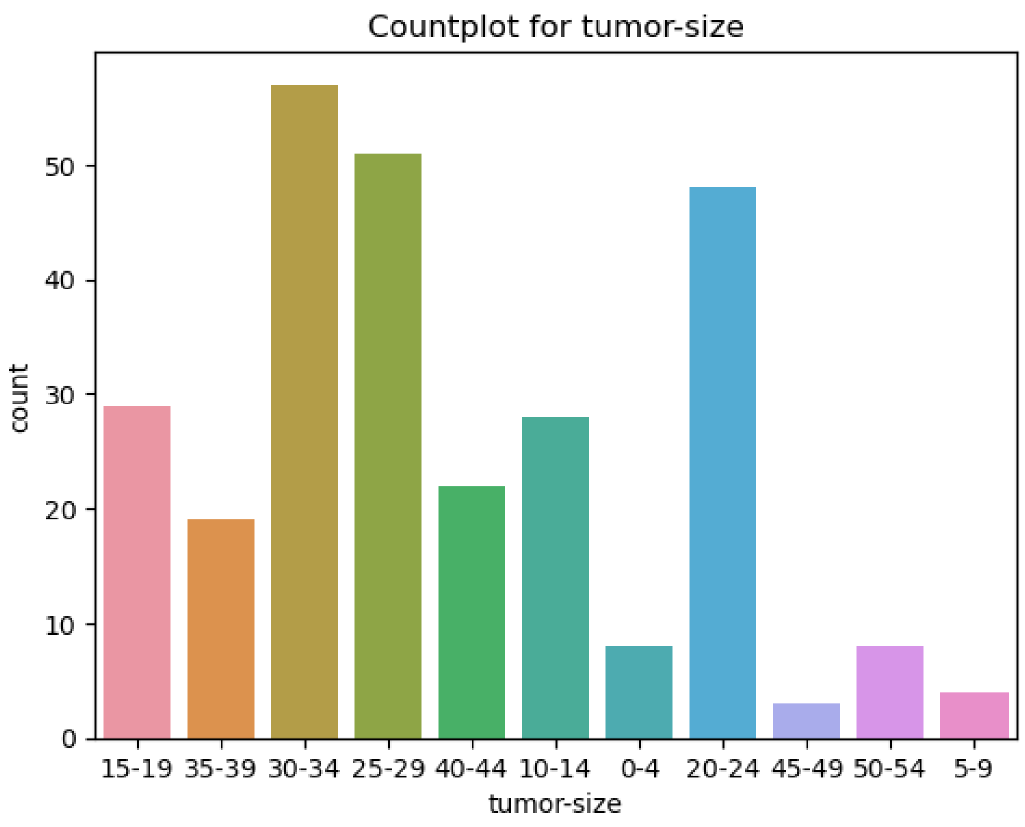
The "Class" variable represents the presence or absence of recurrence events in breast cancer patients.

Unique value	**Count**
no-recurrence-events	**196**
recurrence-events	**81**

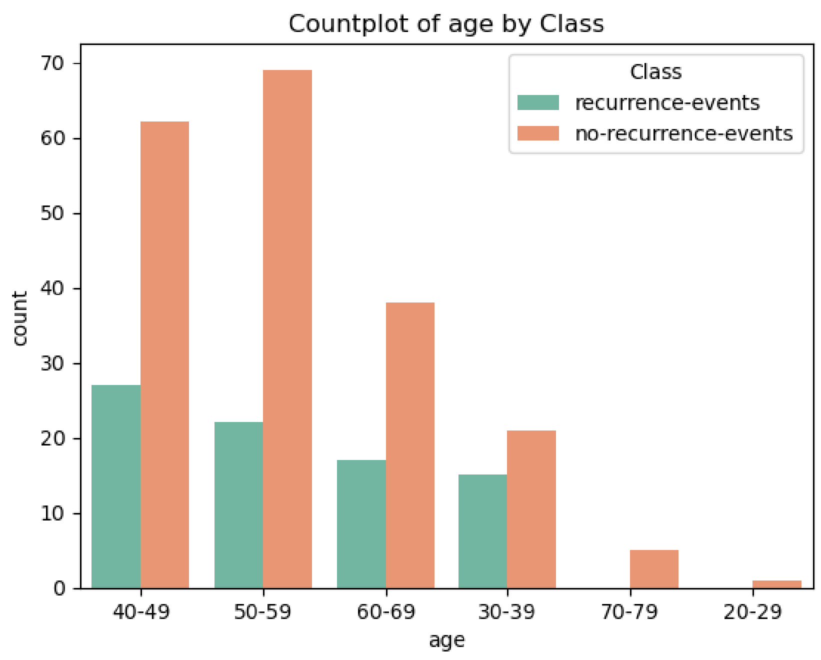
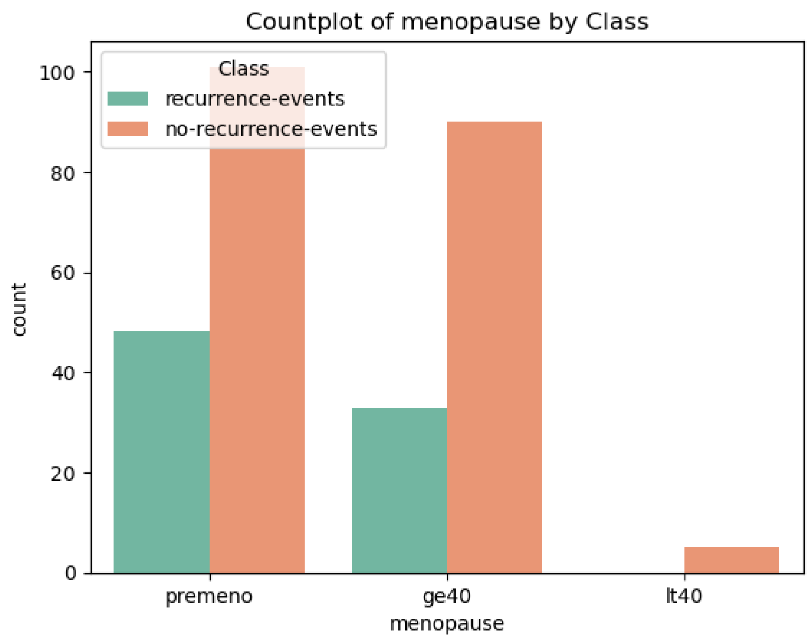


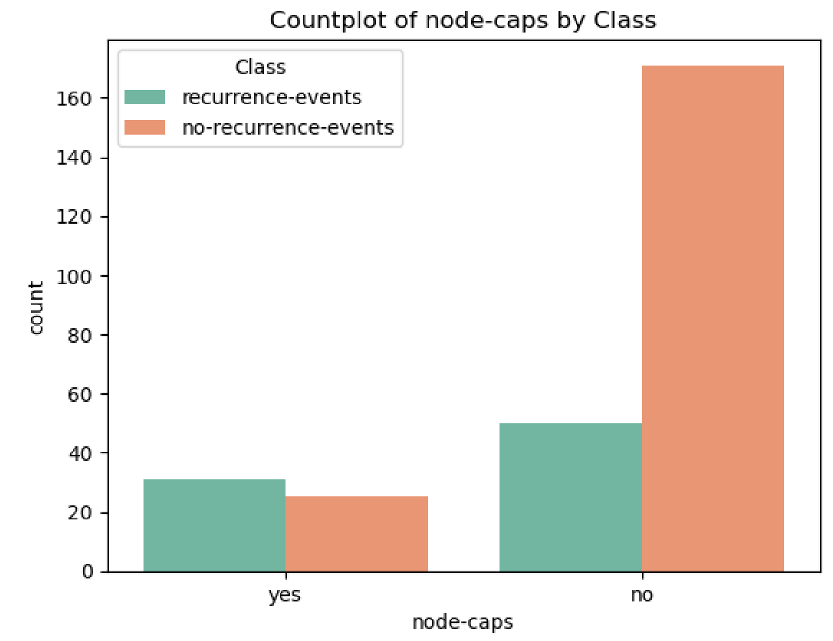
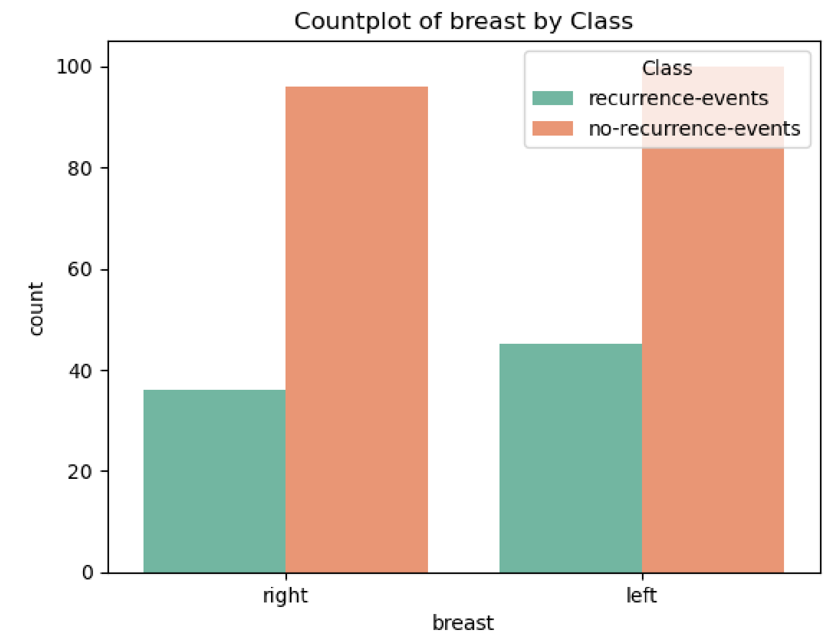
Tumor-size
The "Tumor-size" variable in the breast cancer dataset refers to the size of the tumor observed in breast cancer patients. It represents the dimensions or measurements of the tumor growth.

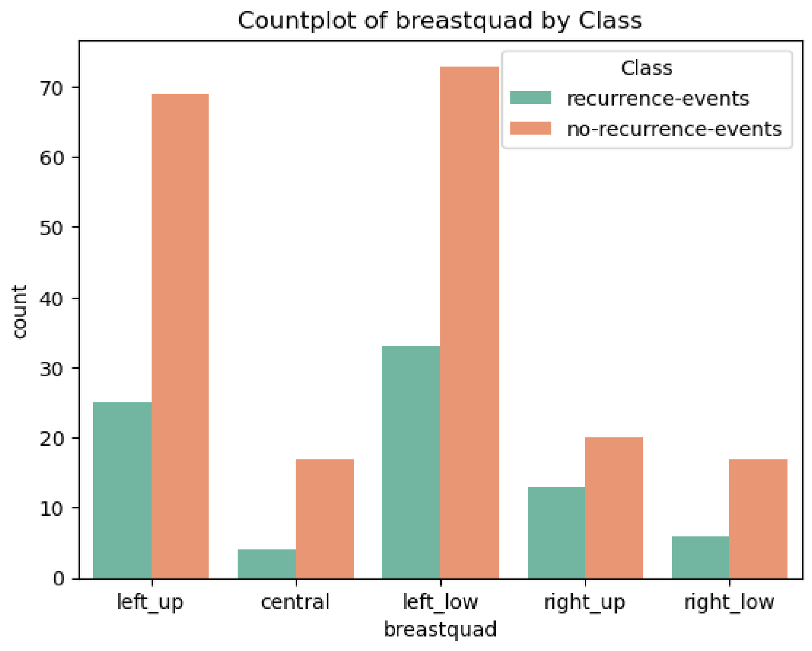
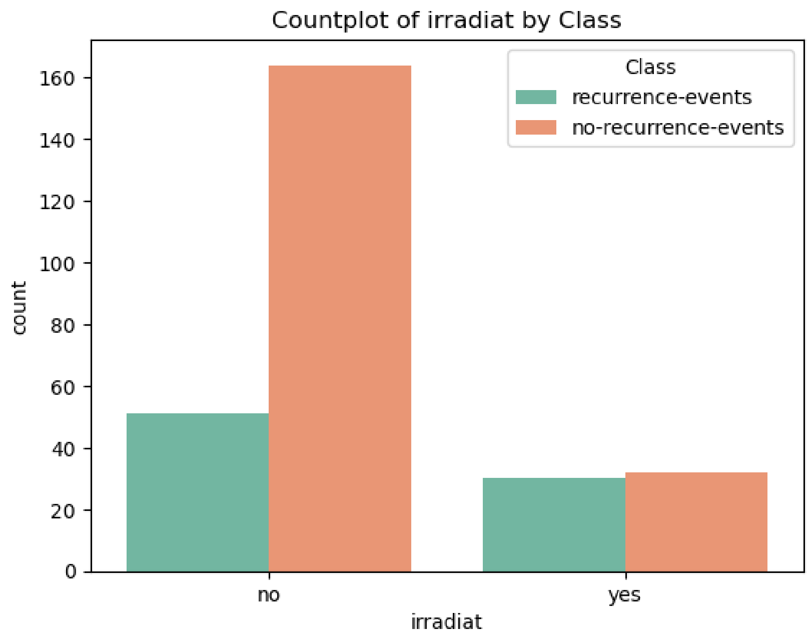
Unique value	**Count**
0-4	**8**
10-14	**28**
15-19	**29**
20-24	**48**
25-29	**51**
30-34	**57**
35-39	**19**
40-44	**22**
45-49	**3**
5-9	**4**
50-54	**8**



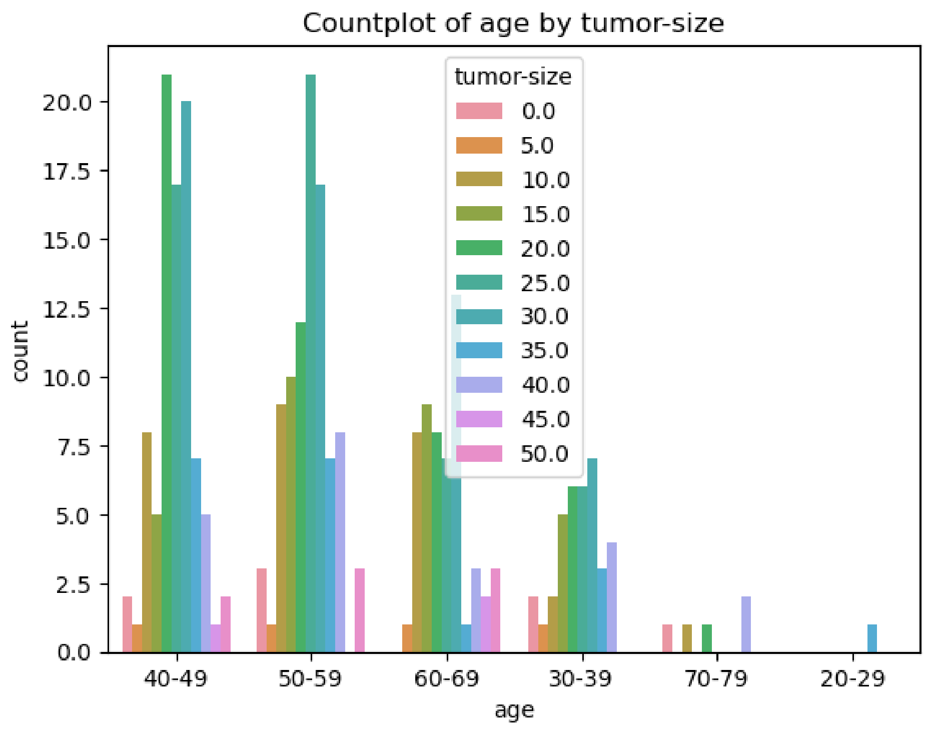
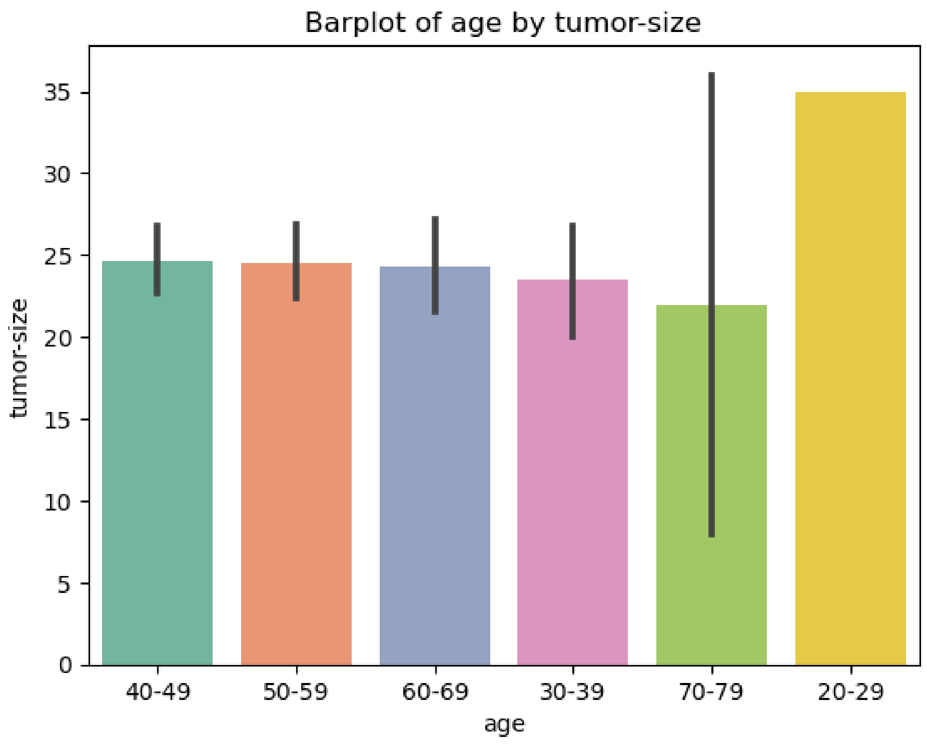
Then we analysed the relationship of the class variable with other variables.

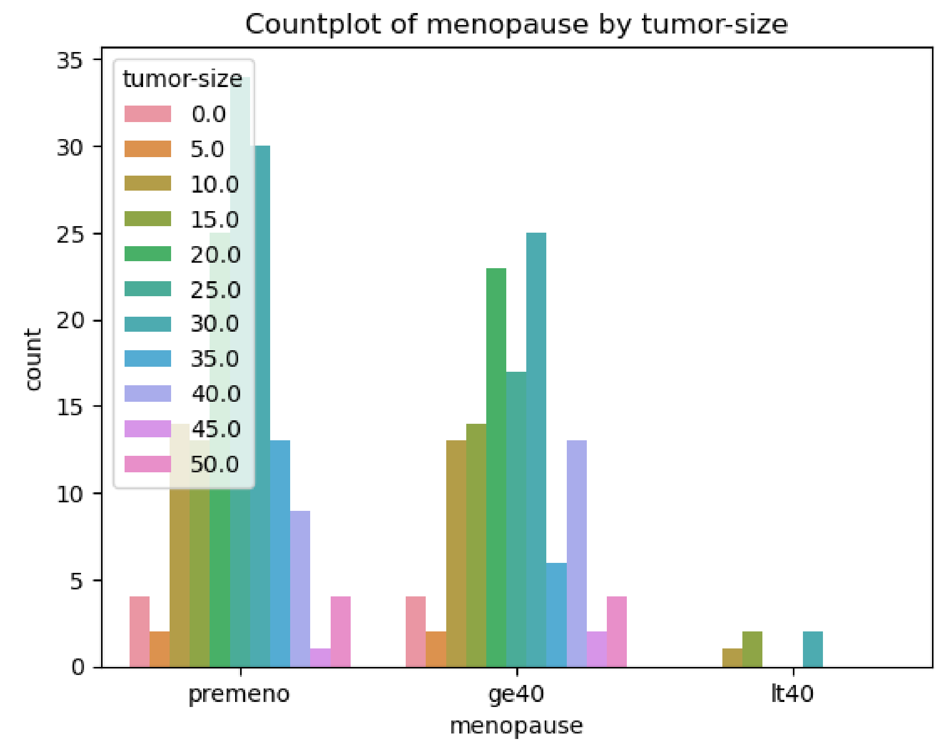
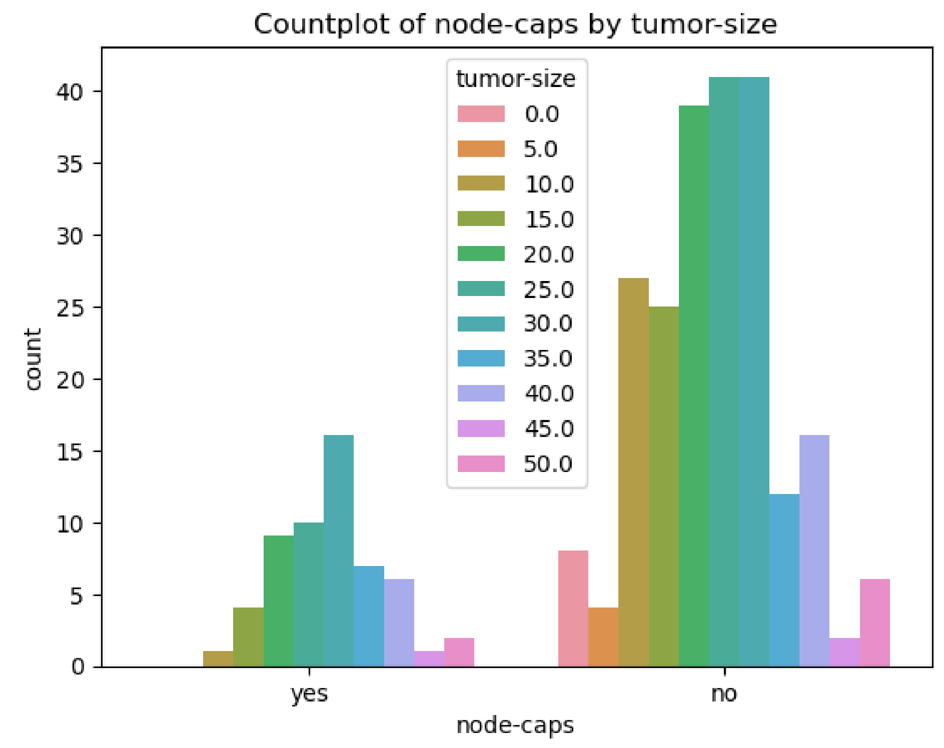
Chart	
	In this graph, we can analyse there is less recurrence events when compared to no-recurrence events. That indicates there is less likelihood to regain cancer again in patients.
	This graph indicates there is less recurrence events in all three stages of menopause while premeno has significant higher recurrence events.

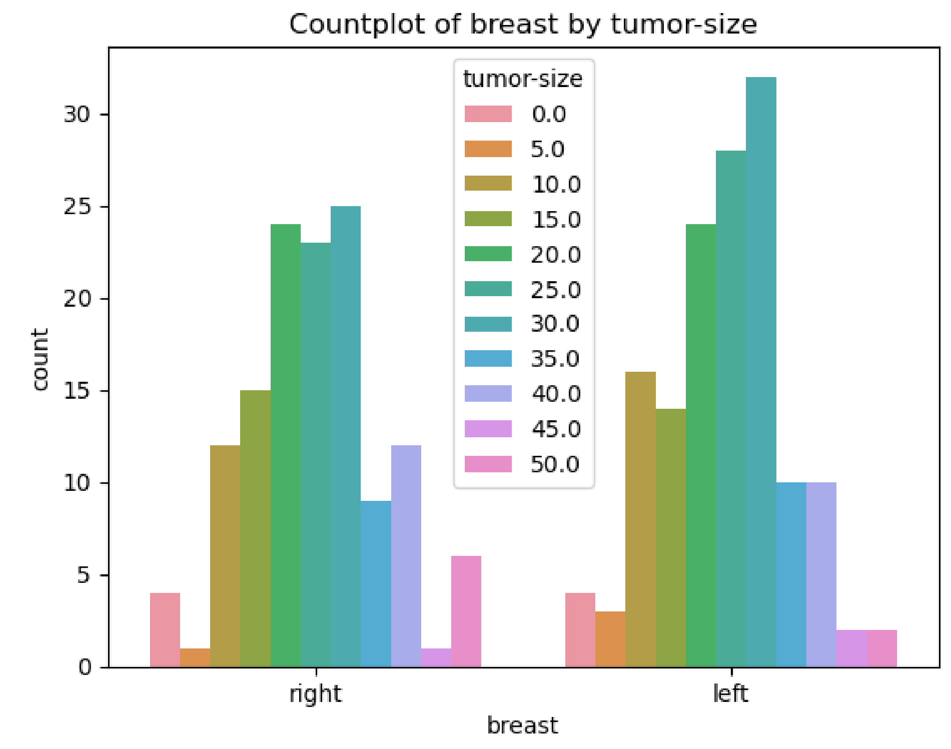
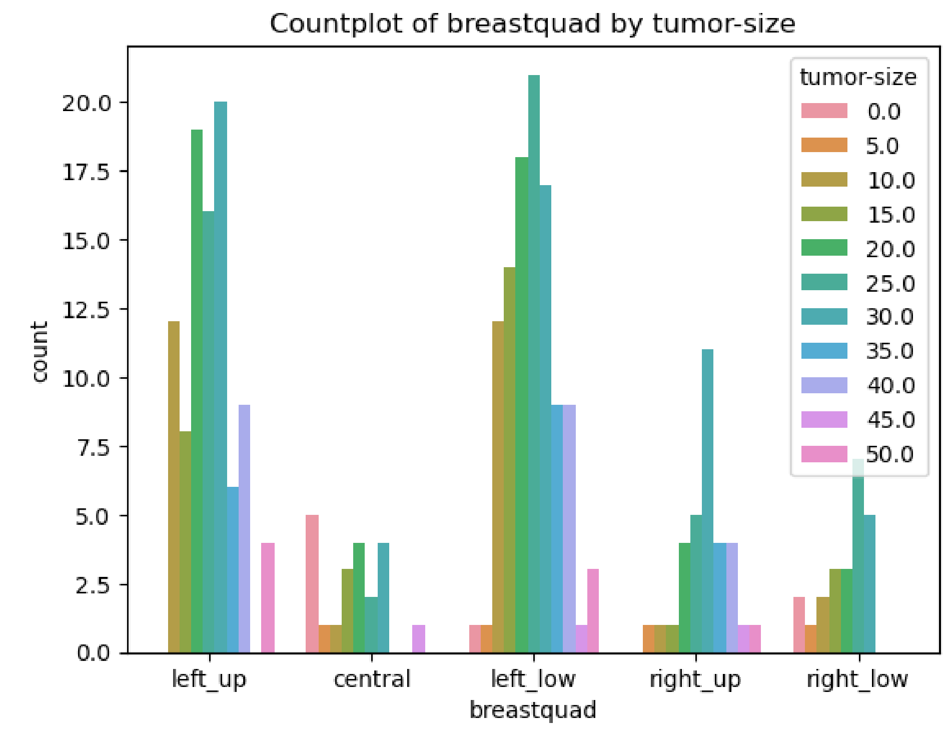
	This graph indicates higher recurrence events in patients who have node caps while patients who have no node caps have a higher number of no-recurrence events.
	This graph indicates both sides of the breast have a higher number of no-recurrence events while the left side breast has a slightly higher number of recurrence events.

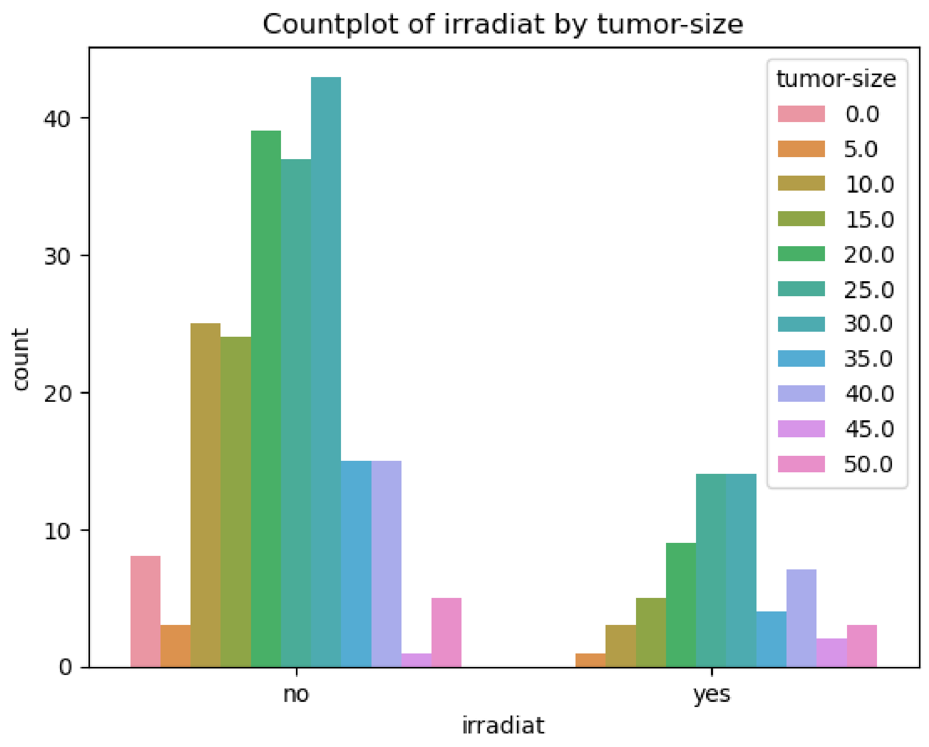
	This graph indicates left up and left-low breast quads have a high number of no-recurrence events. Central has the lowest number of both events.
	This graph indicates patients who have radiant therapy has less number of both events compared to patients who have not been treated with irradiant medication.

After the analysis of the Class variable, we have analyse the relationship of tumor size with other variables.

Graph	
	This graph indicates the age range of 40-49 & 50-59 has 20 to 30 size of tumor sizes. Lowest size of the tumor has been recorded in age 20-29.
	

	This graph indicates highest size of tumor recorded in premeno stage and the ge40 stage.
	This graph indicates patients who has no node caps have a higher size of the tumor.

	This graph indicates that left side of the breast has a higher size of the tumor has recorded.
	This graph indicates left up and left low breast quad have recorded a higher size of tumor compared to right up and right low breast quad.

	This graph indicates patients who have been treated with radiation have a lower number of tumor size when compared to patients who have not been treated with radiation.

As summary Class and tumor size have a different type of relationship with each variable when analyzed.

Categorical variable handling

There are several methods that can be used to address categorical data when it comes to the apply before machine learning algorithms.

Label Encoding:

This method assigns a unique numerical label to each category in the categorical variable. It is suitable for variables with ordinal relationships, where the numerical values have a specific order.

One-Hot Encoding:

This method creates dummy variables for each category in the categorical variable. It represents each category as a binary feature, where 1 indicates the presence of the category and 0 indicates its absence. It is suitable for variables without ordinal relationships

Ordinal Encoding:

This method assigns numerical values to categories based on their order or rank. It is useful when the categories have a specific order or hierarchy.

Target Encoding:

This method replaces each category with the mean or other statistical measures of the target variable for that category. It incorporates the target variable information into the encoding and can be useful for predicting the target variable. Such as the mean of the categorical data.

Library	**Description /Function**
scikit-learn	Provides various preprocessing methods for categorical data, such as LabelEncoder, OneHotEncoder, and OrdinalEncoder
pandas	Offers functions like get_dummies() for one-hot encoding and replace() for label encoding.
category_encoders	A library specifically designed for categorical encoding, offering various methods including target encoding.

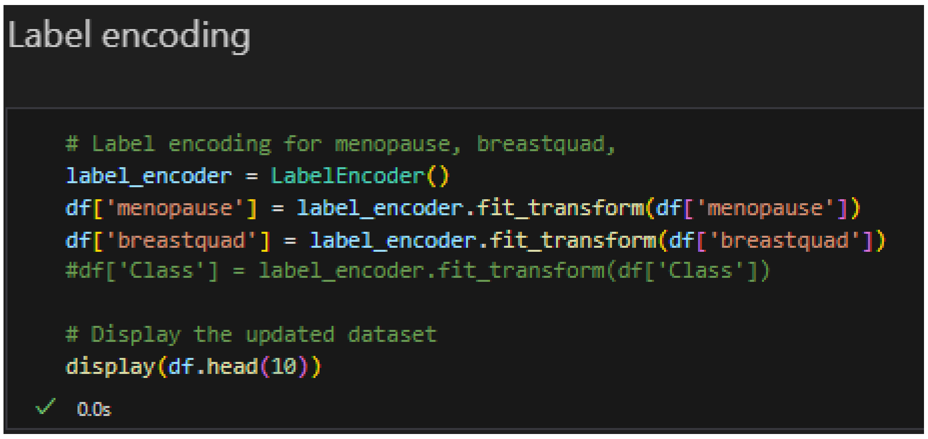
- These methods and libraries offer flexibility in preparing categorical data for machine learning models that require numerical inputs.
- The choice of method depends on the nature of the categorical variable and the specific requirements of the analysis or modeling task.

Application of Choosing the Best Performing Encoding Method

For the breast cancer dataset, I have carefully analyzed the categorical data and chosen to perform **Label Encoding, One-Hot Encoding, and Mean Target Encoding.**

	Label Encoding	**One-Hot Encoding**	**Mean Target Encoding**
Variables	**breastquad**	**breast**	**age**
	menopause	**node-caps**	**inv-nodes**
		irradiat	**tumor-size**

Label Encoding

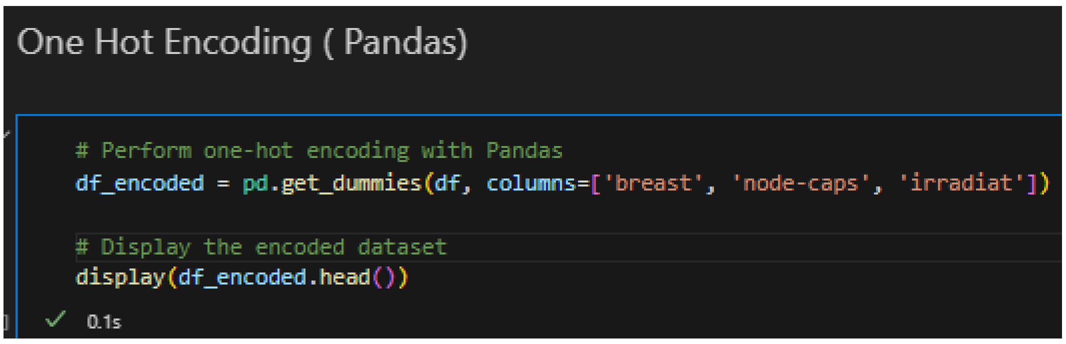


Label encoding

```
# Label encoding for menopause, breastquad,
label_encoder = LabelEncoder()
df['menopause'] = label_encoder.fit_transform(df['menopause'])
df['breastquad'] = label_encoder.fit_transform(df['breastquad'])
#df['Class'] = label_encoder.fit_transform(df['Class'])

# Display the updated dataset
display(df.head(10))
```

✓ 0.0s

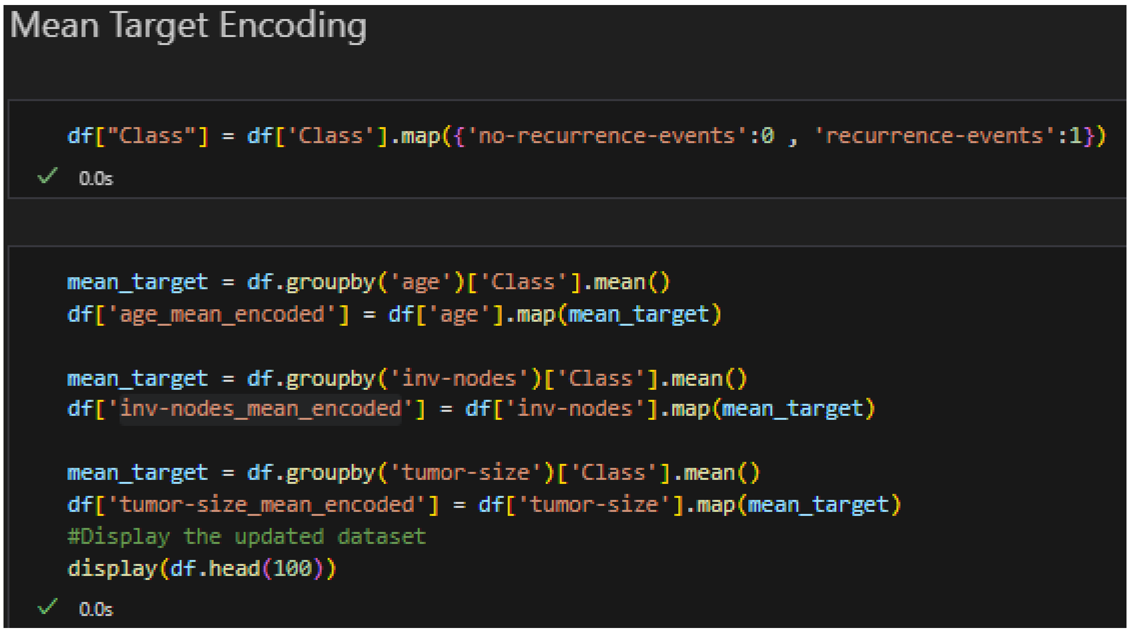
One-Hot Encoding



One Hot Encoding (Pandas)

```
# Perform one-hot encoding with Pandas
df_encoded = pd.get_dummies(df, columns=['breast', 'node-caps', 'irradiat'])

# Display the encoded dataset
display(df_encoded.head())
```

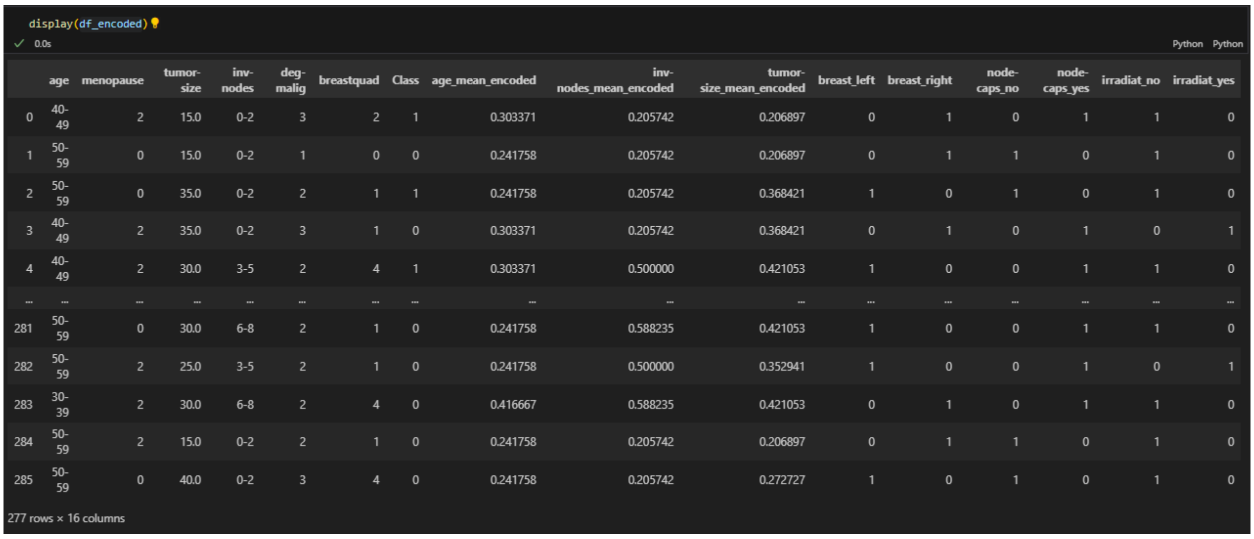
✓ 0.1s



Mean Target Encoding

```python
df["Class"] = df['Class'].map({'no-recurrence-events':0 , 'recurrence-events':1})
```
✓ 0.0s

```python
mean_target = df.groupby('age')['Class'].mean()
df['age_mean_encoded'] = df['age'].map(mean_target)

mean_target = df.groupby('inv-nodes')['Class'].mean()
df['inv-nodes_mean_encoded'] = df['inv-nodes'].map(mean_target)

mean_target = df.groupby('tumor-size')['Class'].mean()
df['tumor-size_mean_encoded'] = df['tumor-size'].map(mean_target)
#Display the updated dataset
display(df.head(100))
```
✓ 0.0s



```python
display(df_encoded)
```
✓ 0.0s

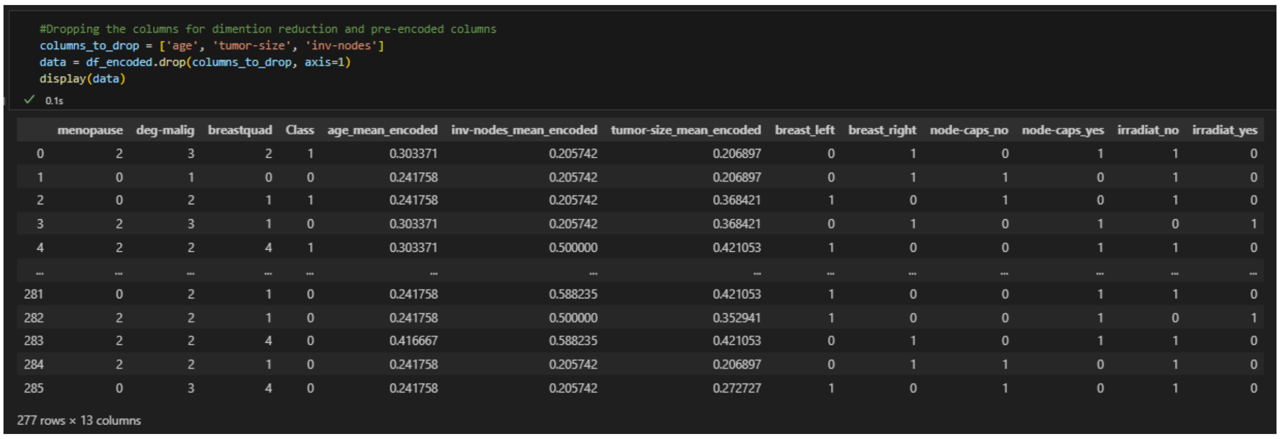
Python Python

	age	menopause	tumor-size	inv-nodes	deg-malig	breastquad	Class	age_mean_encoded	inv-nodes_mean_encoded	tumor-size_mean_encoded	breast_left	breast_right	node-caps_no	node-caps_yes	irradiat_no	irradiat_yes
0	40-49	2	15.0	0-2	3	2	1	0.303371	0.205742	0.206897	0	1	0	1	1	0
1	50-59	0	15.0	0-2	1	0	0	0.241758	0.205742	0.206897	0	1	1	0	1	0
2	50-59	0	35.0	0-2	2	1	1	0.241758	0.205742	0.368421	1	0	1	0	1	0
3	40-49	2	35.0	0-2	3	1	0	0.303371	0.205742	0.368421	0	1	0	1	0	1
4	40-49	2	30.0	3-5	2	4	1	0.303371	0.500000	0.421053	1	0	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
281	50-59	0	30.0	6-8	2	1	0	0.241758	0.588235	0.421053	1	0	0	1	1	0
282	50-59	2	25.0	3-5	2	1	0	0.241758	0.500000	0.352941	1	0	0	1	0	1
283	30-39	2	30.0	6-8	2	4	0	0.416667	0.588235	0.421053	0	1	0	1	1	0
284	50-59	2	15.0	0-2	2	1	0	0.241758	0.205742	0.206897	0	1	1	0	1	0
285	50-59	0	40.0	0-2	3	4	0	0.241758	0.205742	0.272727	1	0	1	0	1	0

277 rows × 16 columns

Grid Search & Apply a Machine Learning Algorithm

To perform the grid search, we will choose from sklearn.ensemble import RandomForestClassifier.



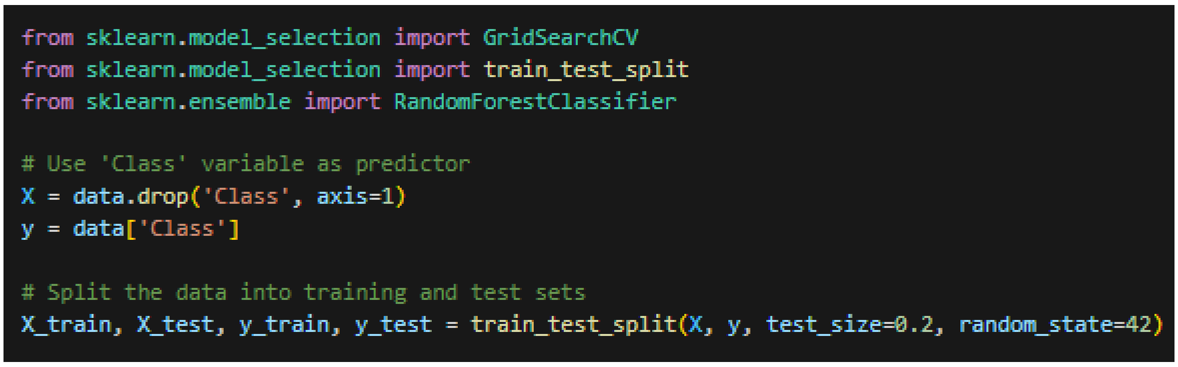
```
#Dropping the columns for dimention reduction and pre-encoded columns
columns_to_drop = ['age', 'tumor-size', 'inv-nodes']
data = df_encoded.drop(columns_to_drop, axis=1)
display(data)
```

✓ 0.1s

	menopause	deg-malig	breastquad	Class	age_mean_encoded	inv-nodes_mean_encoded	tumor-size_mean_encoded	breast_left	breast_right	node-caps_no	node-caps_yes	irradiat_no	irradiat_yes
0	2	3	2	1	0.303371	0.205742	0.206897	0	1	0	1	1	0
1	0	1	0	0	0.241758	0.205742	0.206897	0	1	1	0	1	0
2	0	2	1	1	0.241758	0.205742	0.368421	1	0	1	0	1	0
3	2	3	1	0	0.303371	0.205742	0.368421	0	1	0	1	0	1
4	2	2	4	1	0.303371	0.500000	0.421053	1	0	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
281	0	2	1	0	0.241758	0.588235	0.421053	1	0	0	1	1	0
282	2	2	1	0	0.241758	0.500000	0.352941	1	0	0	1	0	1
283	2	2	4	0	0.416667	0.588235	0.421053	0	1	0	1	1	0
284	2	2	1	0	0.241758	0.205742	0.206897	0	1	1	0	1	0
285	0	3	4	0	0.241758	0.205742	0.272727	1	0	1	0	1	0

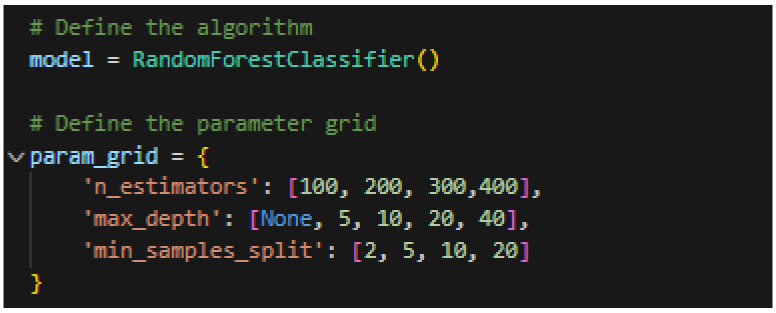
277 rows × 13 columns

Import the sklearn library



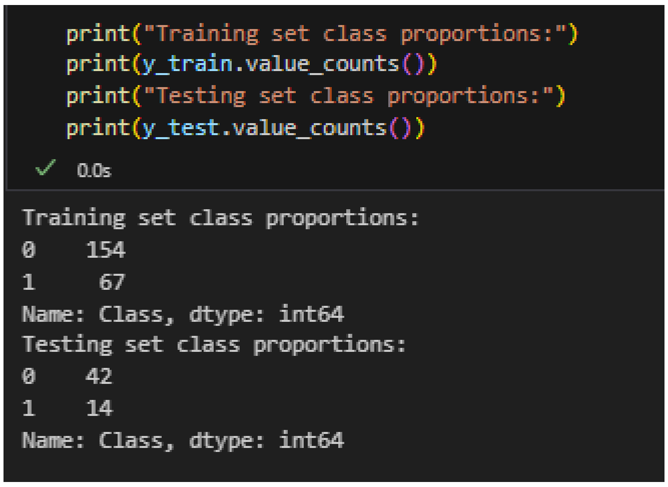
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Use 'Class' variable as predictor
X = data.drop('Class', axis=1)
y = data['Class']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Define the RandomForestClassifier model



```
# Define the algorithm
model = RandomForestClassifier()

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300,400],
    'max_depth': [None, 5, 10, 20, 40],
    'min_samples_split': [2, 5, 10, 20]
}
```

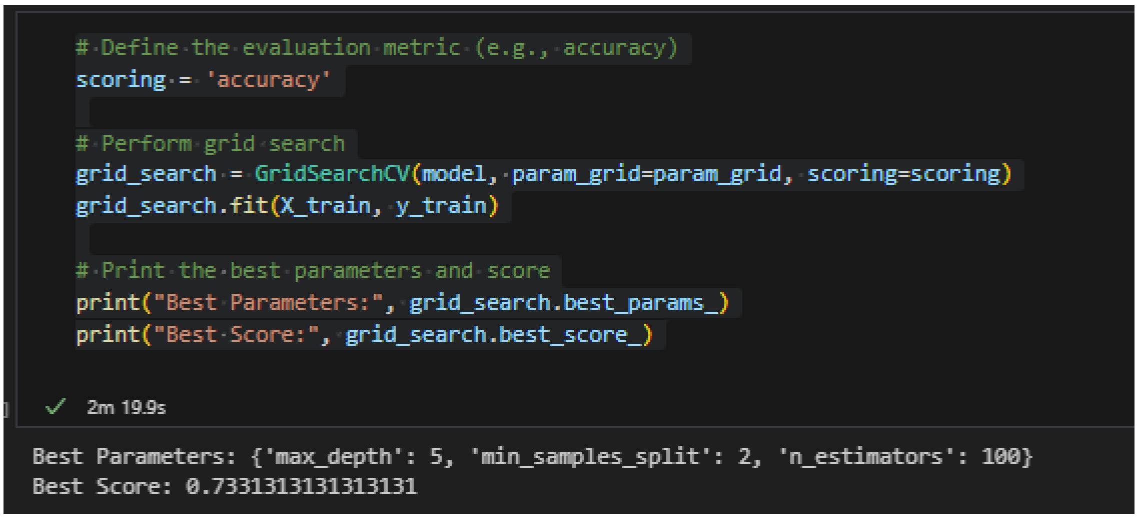
Algorithm Output



```
print("Training set class proportions:")
print(y_train.value_counts())
print("Testing set class proportions:")
print(y_test.value_counts())
```

✓ 0.0s

```
Training set class proportions:
0    154
1     67
Name: Class, dtype: int64
Testing set class proportions:
0    42
1    14
Name: Class, dtype: int64
```



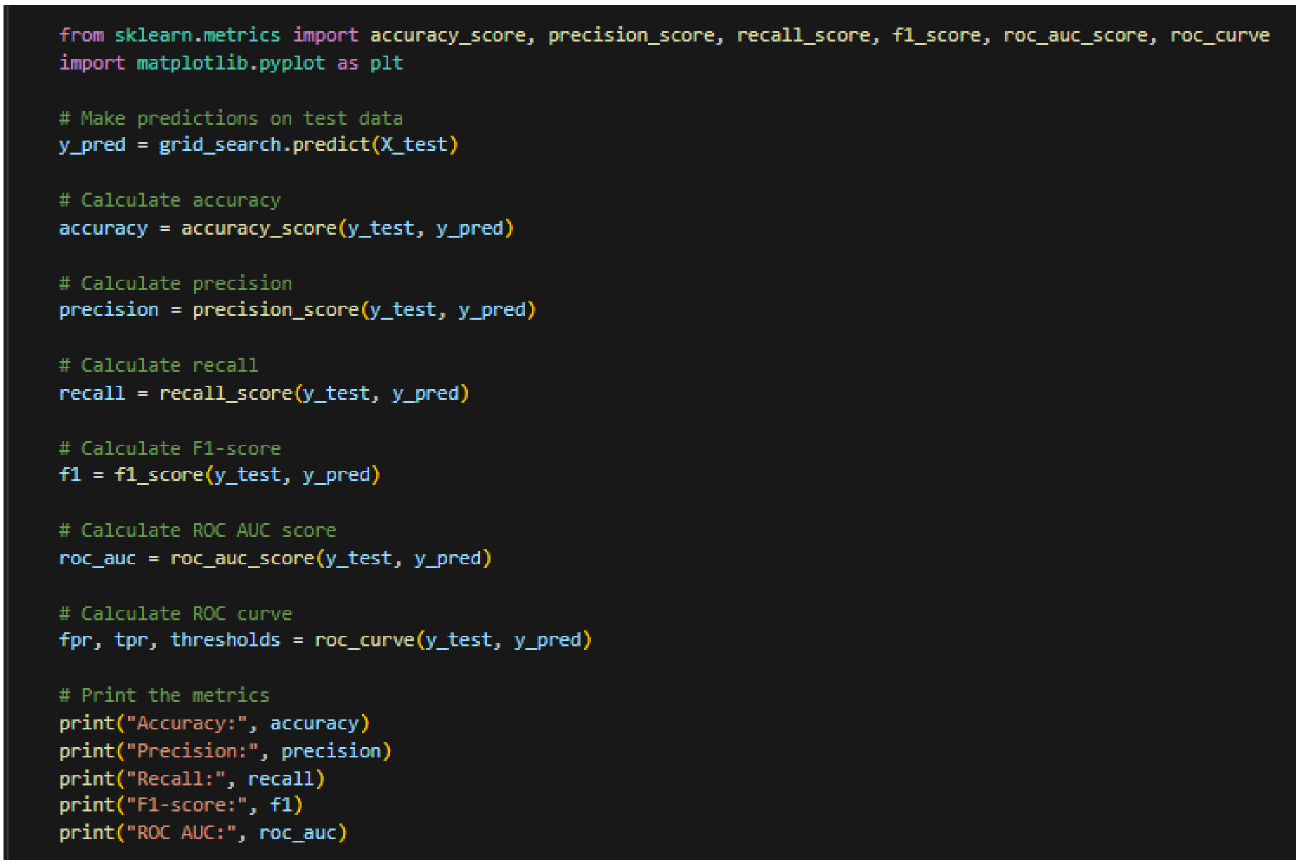
```
# Define the evaluation metric (e.g., accuracy)
scoring = 'accuracy'

# Perform grid search
grid_search = GridSearchCV(model, param_grid=param_grid, scoring=scoring)
grid_search.fit(X_train, y_train)

# Print the best parameters and score
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

✓ 2m 19.9s

```
Best Parameters: {'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 100}
Best Score: 0.7331313131313131
```

Best Parameters: {'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 100}
Best Score: 0.7331313131313131

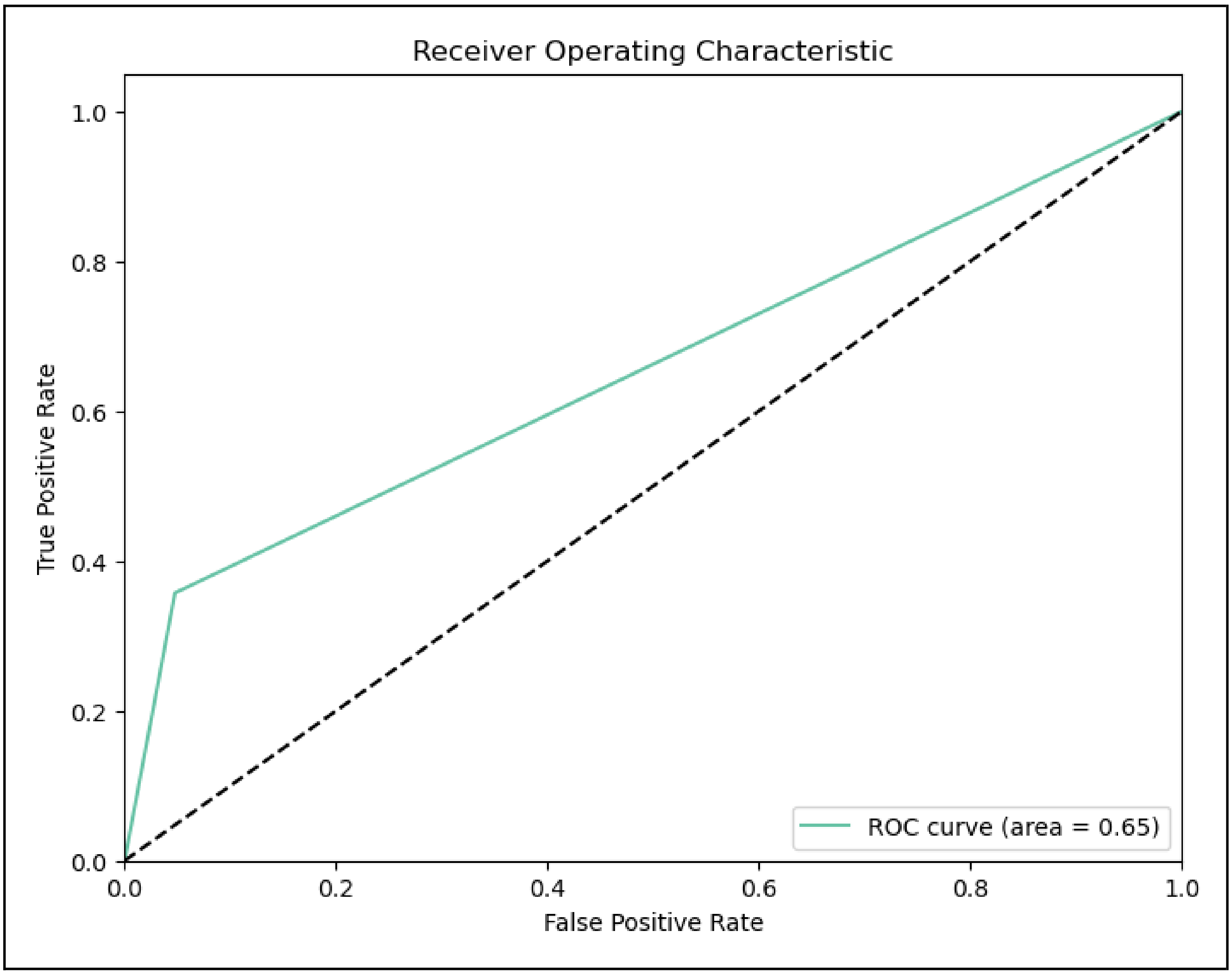
Evaluating ML model performance



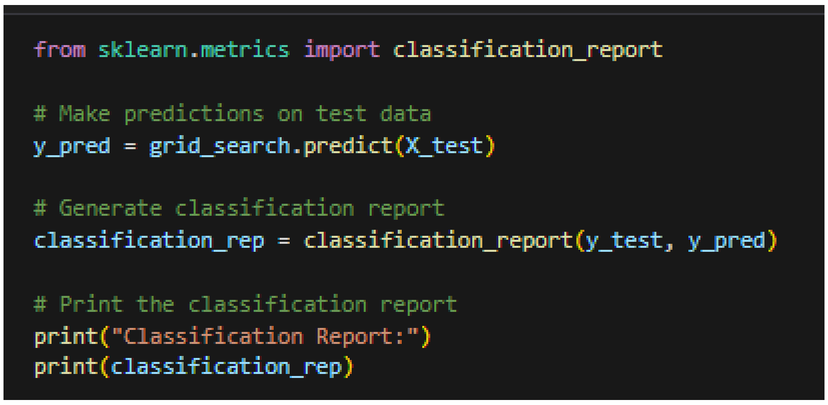
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Make predictions on test data
y_pred = grid_search.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate precision
precision = precision_score(y_test, y_pred)

# Calculate recall
recall = recall_score(y_test, y_pred)

# Calculate F1-score
f1 = f1_score(y_test, y_pred)

# Calculate ROC AUC score
roc_auc = roc_auc_score(y_test, y_pred)

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)

# Print the metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("ROC AUC:", roc_auc)
```

```
Accuracy: 0.8035714285714286
Precision: 0.7142857142857143
Recall: 0.35714285714285715
F1-score: 0.47619047619047623
ROC AUC: 0.6547619047619047
```

Plot the ROC curve


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.65)

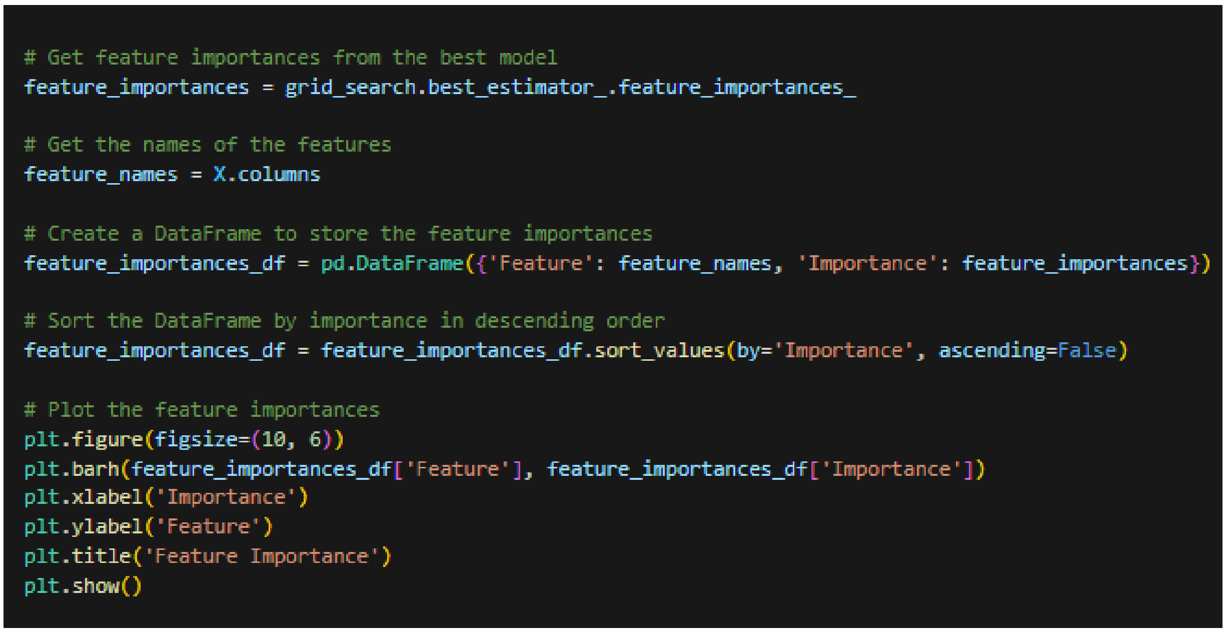
Classification Report:



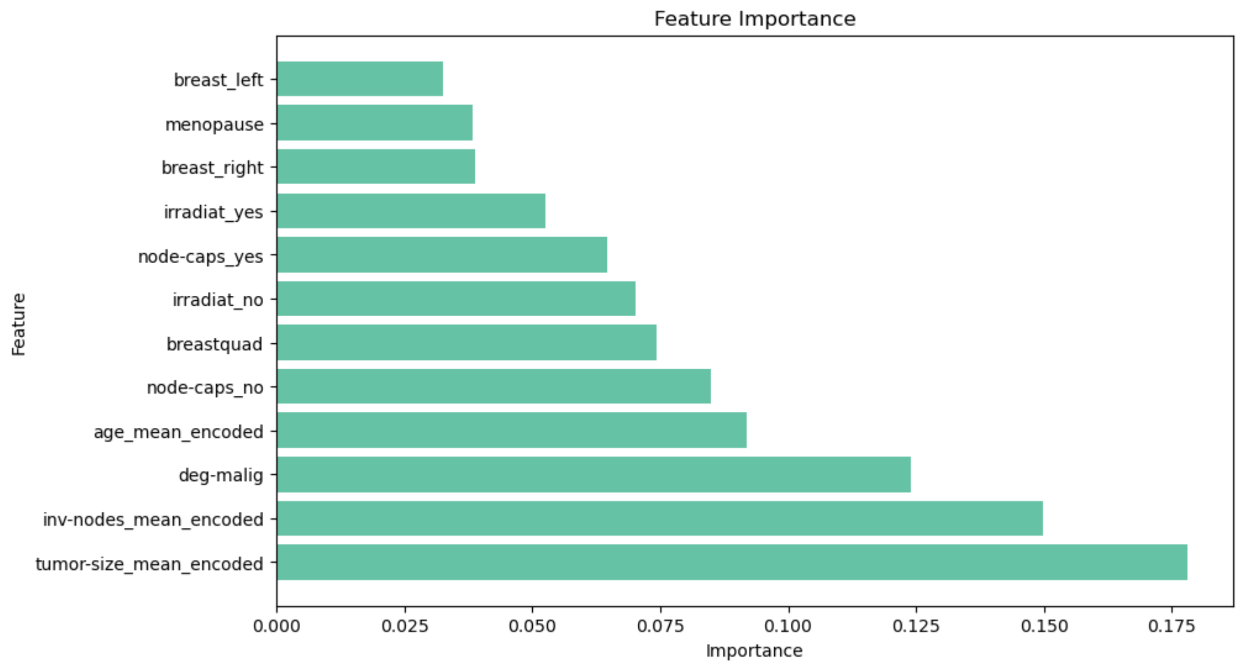
```python
from sklearn.metrics import classification_report

# Make predictions on test data
y_pred = grid_search.predict(X_test)

# Generate classification report
classification_rep = classification_report(y_test, y_pred)

# Print the classification report
print("Classification Report:")
print(classification_rep)
```

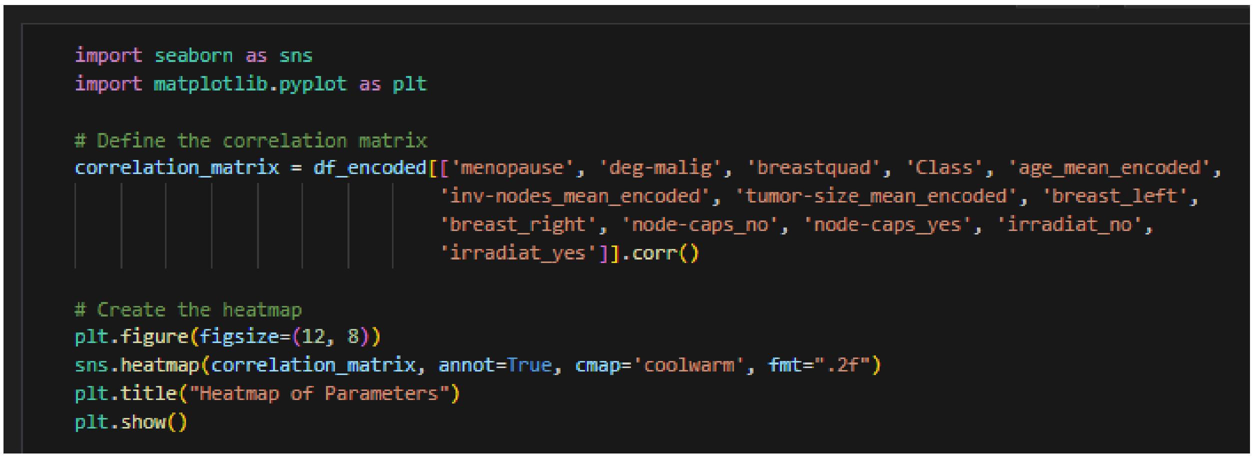
	precision	recall	f1-score	support
0	0.82	0.95	0.88	42
1	0.71	0.36	0.48	14
accuracy			0.80	56
macro avg	0.77	0.65	0.68	56
weighted avg	0.79	0.80	0.78	56

Analysing Feature Importance



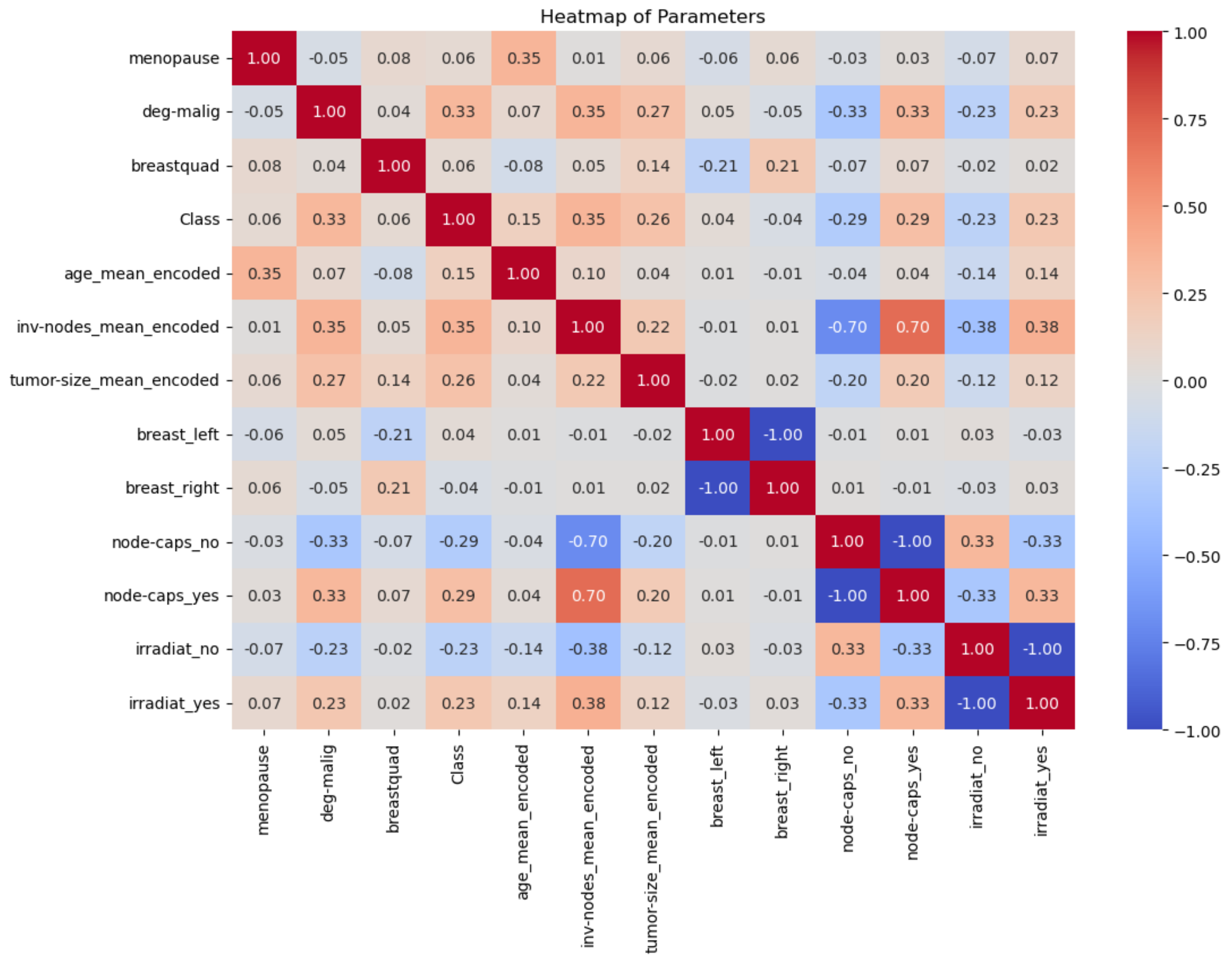
```python
# Get feature importances from the best model
feature_importances = grid_search.best_estimator_.feature_importances_

# Get the names of the features
feature_names = X.columns

# Create a DataFrame to store the feature importances
feature_importances_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})

# Sort the DataFrame by importance in descending order
feature_importances_df = feature_importances_df.sort_values(by='Importance', ascending=False)

# Plot the feature importances
plt.figure(figsize=(10, 6))
plt.barh(feature_importances_df['Feature'], feature_importances_df['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance')
plt.show()
```





```python
import seaborn as sns
import matplotlib.pyplot as plt

# Define the correlation matrix
correlation_matrix = df_encoded[['menopause', 'deg-malig', 'breastquad', 'Class', 'age_mean_encoded',
                                 'inv-nodes_mean_encoded', 'tumor-size_mean_encoded', 'breast_left',
                                 'breast_right', 'node-caps_no', 'node-caps_yes', 'irradiat_no',
                                 'irradiat_yes']].corr()

# Create the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Heatmap of Parameters")
plt.show()
```

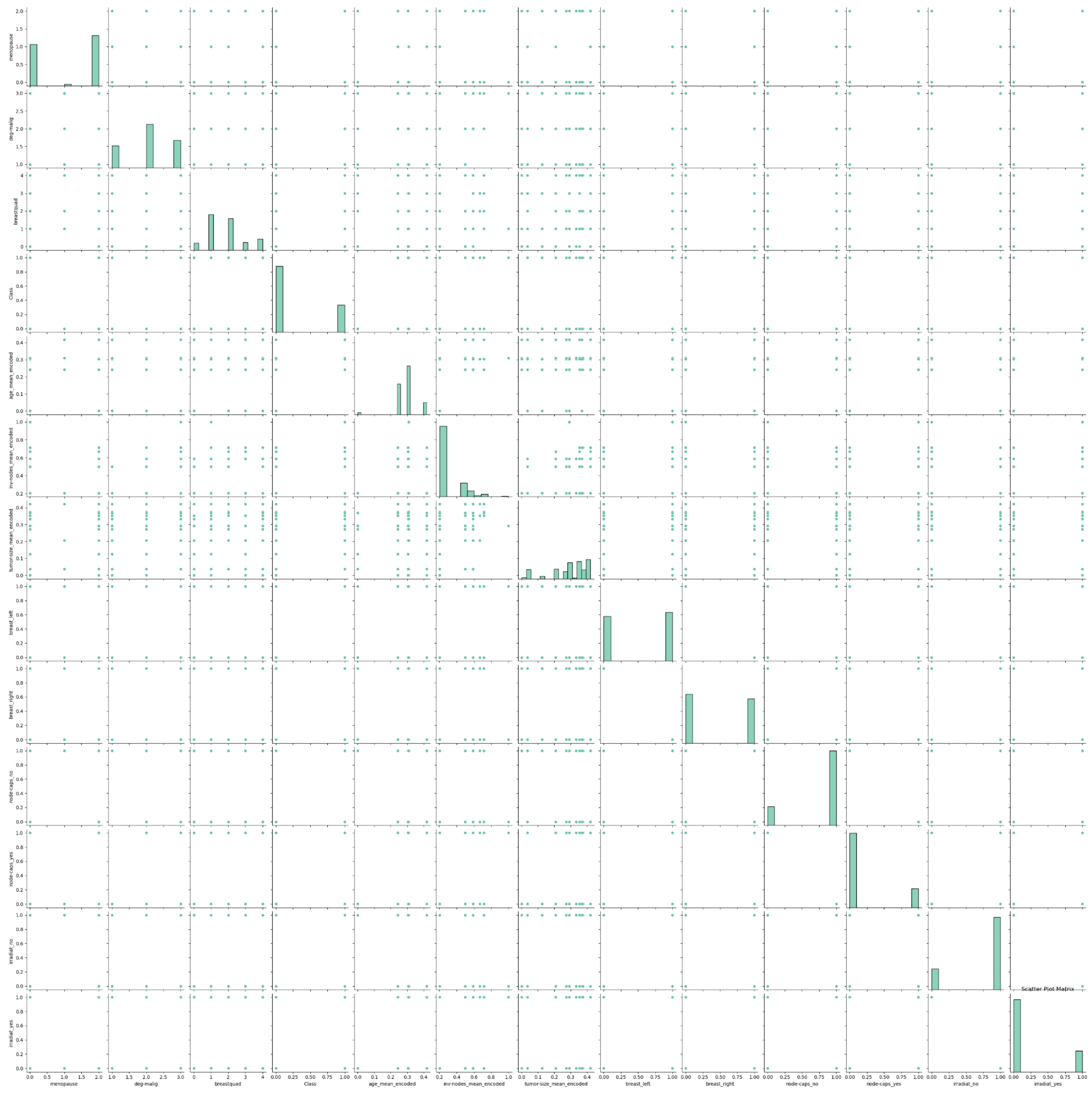
Heat Map of Encoded Dataset



	menopause	deg-malig	breastquad	Class	age_mean_encoded	inv-nodes_mean_encoded	tumor-size_mean_encoded	breast_left	breast_right	node-caps_no	node-caps_yes	irradiat_no	irradiat_yes
menopause	1.00	-0.05	0.08	0.06	0.35	0.01	0.06	-0.06	0.06	-0.03	0.03	-0.07	0.07
deg-malig	-0.05	1.00	0.04	0.33	0.07	0.35	0.27	0.05	-0.05	-0.33	0.33	-0.23	0.23
breastquad	0.08	0.04	1.00	0.06	-0.08	0.05	0.14	-0.21	0.21	-0.07	0.07	-0.02	0.02
Class	0.06	0.33	0.06	1.00	0.15	0.35	0.26	0.04	-0.04	-0.29	0.29	-0.23	0.23
age_mean_encoded	0.35	0.07	-0.08	0.15	1.00	0.10	0.04	0.01	-0.01	-0.04	0.04	-0.14	0.14
inv-nodes_mean_encoded	0.01	0.35	0.05	0.35	0.10	1.00	0.22	-0.01	0.01	-0.70	0.70	-0.38	0.38
tumor-size_mean_encoded	0.06	0.27	0.14	0.26	0.04	0.22	1.00	-0.02	0.02	-0.20	0.20	-0.12	0.12
breast_left	-0.06	0.05	-0.21	0.04	0.01	-0.01	-0.02	1.00	-1.00	-0.01	0.01	0.03	-0.03
breast_right	0.06	-0.05	0.21	-0.04	-0.01	0.01	0.02	-1.00	1.00	0.01	-0.01	-0.03	0.03
node-caps_no	-0.03	-0.33	-0.07	-0.29	-0.04	-0.70	-0.20	-0.01	0.01	1.00	-1.00	0.33	-0.33
node-caps_yes	0.03	0.33	0.07	0.29	0.04	0.70	0.20	0.01	-0.01	-1.00	1.00	-0.33	0.33
irradiat_no	-0.07	-0.23	-0.02	-0.23	-0.14	-0.38	-0.12	0.03	-0.03	0.33	-0.33	1.00	-1.00
irradiat_yes	0.07	0.23	0.02	0.23	0.14	0.38	0.12	-0.03	0.03	-0.33	0.33	-1.00	1.00

By analysing this heatmap we can identify node_cap_yes and inv-nodes_mean_encoded has a positive correlation.

Scatter Plot Matrix



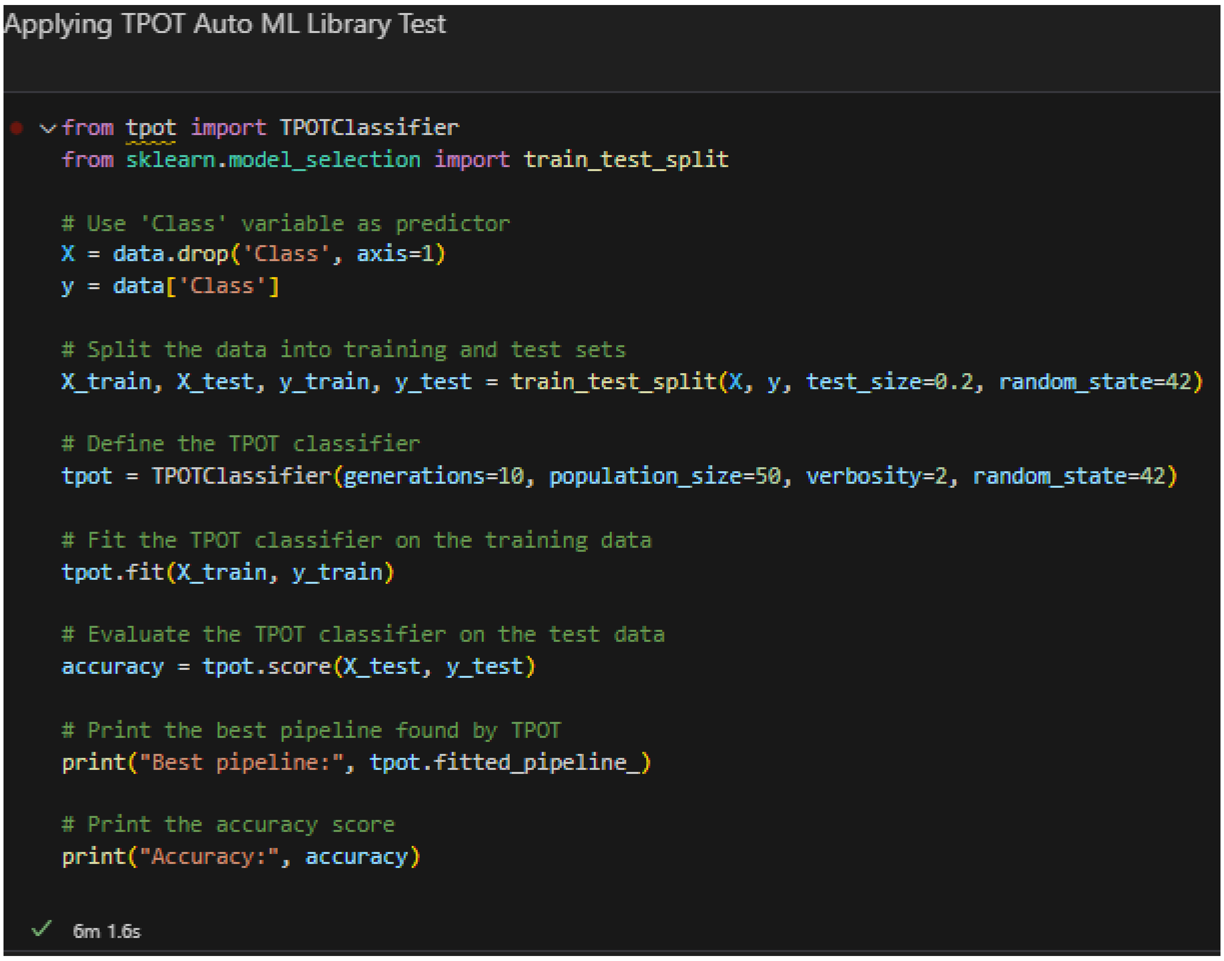
Auto Machine Learning Libraries

AutoML libraries automate the process of building machine learning models by providing functionalities such as data preprocessing, feature engineering, model selection, and hyperparameter tuning. They simplify and streamline the machine learning workflow, making it more accessible to users with varying levels of expertise. These libraries incorporate advanced algorithms and techniques to optimize model performance and handle diverse data types. They enable users to compare models, select the best-performing one, and interpret results through visualizations. AutoML libraries help expedite model development, making machine learning more accessible and efficient for researchers and practitioners.

MLjar	MLjar is an open-source AutoML library that focuses on providing a user-friendly interface and automated machine learning pipeline. It supports both classification and regression tasks and offers automatic feature engineering, hyperparameter tuning, model selection, and model explanation capabilities
TransmogrifAI	TransmogrifAI is an AutoML library developed by Salesforce. It is specifically designed for building machine learning models in the context of structured data. TransmogrifAI automates various stages of the machine learning pipeline, including feature engineering, hyperparameter tuning, and model training, to accelerate the development of predictive models.
Auto-sklearn	Auto-sklearn is an AutoML framework built on top of scikit-learn. It leverages Bayesian optimization and ensemble methods to automatically select and configure machine learning models. It also performs feature preprocessing and hyperparameter optimization.
H2O.ai	H2O AutoML is known for its ease of use and powerful capabilities. It a utomates the process of training and tuning machine learning models, with support for various algorithms and techniques.
TPOT	TPOT is a popular AutoML library that utilizes genetic programming to automate the machine learning pipeline. It explores a wide range of models, preprocessing techniques, and feature selection methods to find the best pipeline configuration.
Auto-sklearn	Auto-sklearn is a robust AutoML framework that employs Bayesian optimization and ensemble construction to automate model selection and hyperparameter tuning. It provides a user-friendly interface and handles various machine learning tasks.
MLbox	MLbox is an open-source AutoML library that offers automated feature engineering, model selection, and hyperparameter optimization. It supports both classification and regression tasks and provides an intuitive API.

AutoKeras	AutoKeras is an open-source AutoML library specifically designed for deep learning tasks. It automates the process of building and optimizing deep learning models, including preprocessing, architecture search, and hyperparameter tuning.
Google Cloud AutoML	Google Cloud AutoML is a cloud-based AutoML platform that enables users to build custom machine learning models without extensive knowledge of machine learning. It offers AutoML services for vision, language, and structured data tasks, providing a user-friendly interface and automated model training.
AutoGluon:	AutoGluon is a deep learning AutoML framework developed by Amazon. It automates the process of training and tuning deep neural networks, allowing users to quickly build high-performance models with minimal effort.

I have selected the TPOT Auto ML Library for this application

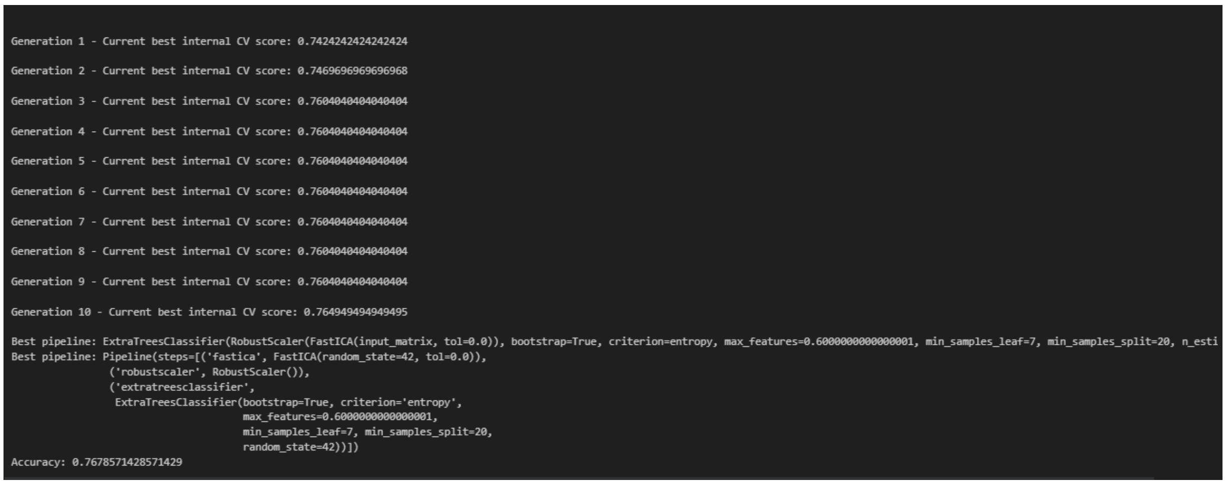


Applying TPOT Auto ML Library Test

```python
from tpot import TPOTClassifier
from sklearn.model_selection import train_test_split

# Use 'Class' variable as predictor
X = data.drop('Class', axis=1)
y = data['Class']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the TPOT classifier
tpot = TPOTClassifier(generations=10, population_size=50, verbosity=2, random_state=42)

# Fit the TPOT classifier on the training data
tpot.fit(X_train, y_train)

# Evaluate the TPOT classifier on the test data
accuracy = tpot.score(X_test, y_test)

# Print the best pipeline found by TPOT
print("Best pipeline:", tpot.fitted_pipeline_)

# Print the accuracy score
print("Accuracy:", accuracy)
```

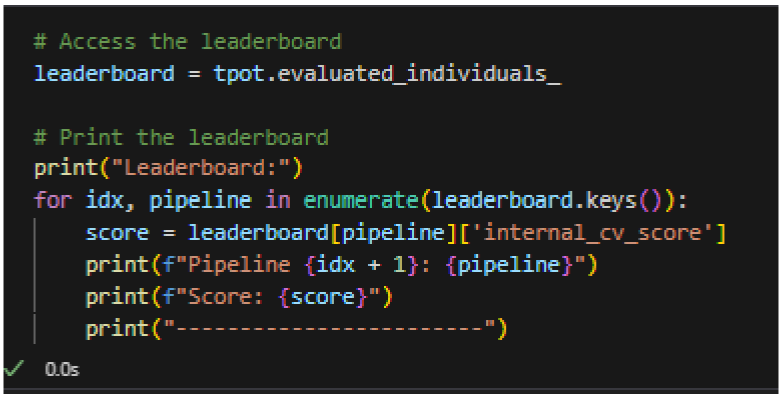
6m 1.6s

Output Result



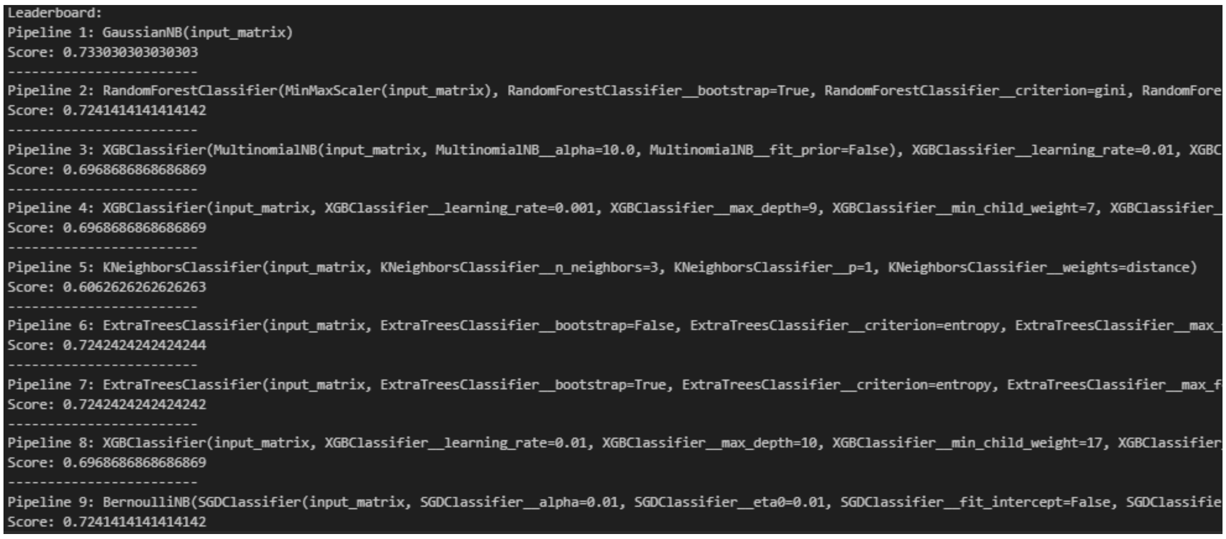
```
Generation 1 - Current best internal CV score: 0.7424242424242424

Generation 2 - Current best internal CV score: 0.7469696969696968

Generation 3 - Current best internal CV score: 0.7604040404040404

Generation 4 - Current best internal CV score: 0.7604040404040404

Generation 5 - Current best internal CV score: 0.7604040404040404

Generation 6 - Current best internal CV score: 0.7604040404040404

Generation 7 - Current best internal CV score: 0.7604040404040404

Generation 8 - Current best internal CV score: 0.7604040404040404

Generation 9 - Current best internal CV score: 0.7604040404040404

Generation 10 - Current best internal CV score: 0.764949494949495

Best pipeline: ExtraTreesClassifier(RobustScaler(FastICA(input_matrix, tol=0.0)), bootstrap=True, criterion=entropy, max_features=0.6000000000000001, min_samples_leaf=7, min_samples_split=20, n_esti
Best pipeline: Pipeline(steps=[('fastica', FastICA(random_state=42, tol=0.0)),
                ('robustscaler', RobustScaler()),
                ('extratreesclassifier',
                 ExtraTreesClassifier(bootstrap=True, criterion='entropy',
                                      max_features=0.6000000000000001,
                                      min_samples_leaf=7, min_samples_split=20,
                                      random_state=42))])
Accuracy: 0.7678571428571429
```

Accuracy: 0.7678571428571429



```python
# Access the leaderboard
leaderboard = tpot.evaluated_individuals_

# Print the leaderboard
print("Leaderboard:")
for idx, pipeline in enumerate(leaderboard.keys()):
    score = leaderboard[pipeline]['internal_cv_score']
    print(f"Pipeline {idx + 1}: {pipeline}")
    print(f"Score: {score}")
    print("------------------------")
```

0.0s



```
Leaderboard:
Pipeline 1: GaussianNB(input_matrix)
Score: 0.733030303030303
------------------------
Pipeline 2: RandomForestClassifier(MinMaxScaler(input_matrix), RandomForestClassifier__bootstrap=True, RandomForestClassifier__criterion=gini, RandomFore
Score: 0.7241414141414142
------------------------
Pipeline 3: XGBClassifier(MultinomialNB(input_matrix, MultinomialNB__alpha=10.0, MultinomialNB__fit_prior=False), XGBClassifier__learning_rate=0.01, XGBC
Score: 0.6968686868686869
------------------------
Pipeline 4: XGBClassifier(input_matrix, XGBClassifier__learning_rate=0.001, XGBClassifier__max_depth=9, XGBClassifier__min_child_weight=7, XGBClassifier_
Score: 0.6968686868686869
------------------------
Pipeline 5: KNeighborsClassifier(input_matrix, KNeighborsClassifier__n_neighbors=3, KNeighborsClassifier__p=1, KNeighborsClassifier__weights=distance)
Score: 0.6062626262626263
------------------------
Pipeline 6: ExtraTreesClassifier(input_matrix, ExtraTreesClassifier__bootstrap=False, ExtraTreesClassifier__criterion=entropy, ExtraTreesClassifier__max_
Score: 0.7242424242424244
------------------------
Pipeline 7: ExtraTreesClassifier(input_matrix, ExtraTreesClassifier__bootstrap=True, ExtraTreesClassifier__criterion=entropy, ExtraTreesClassifier__max_f
Score: 0.7242424242424242
------------------------
Pipeline 8: XGBClassifier(input_matrix, XGBClassifier__learning_rate=0.01, XGBClassifier__max_depth=10, XGBClassifier__min_child_weight=17, XGBClassifier
Score: 0.6968686868686869
------------------------
Pipeline 9: BernoulliNB(SGDClassifier(input_matrix, SGDClassifier__alpha=0.01, SGDClassifier__eta0=0.01, SGDClassifier__fit_intercept=False, SGDClassifie
Score: 0.7241414141414142
```

References

- https://pypi.org/project/TPOT/
- https://archive.ics.uci.edu/dataset/14/breast+cancer
- https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/
-