Breast Cancer Detection Using Machine Learning

Atreyo Bhattacharyya

NMS Team Name: Hackwack

Team Member: Individual

The Heritage School, Kolkata, West Bengal

Class - VII

Background

Worldwide, breast cancer is the most common type of cancer in women and the second highest in terms of mortality rates. Diagnosis of breast cancer is performed when an abnormal lump is found (from self-examination or x-ray) or a tiny speck of calcium is seen (on an x-ray). After a suspicious lump is found, the doctor will conduct a diagnosis to determine whether it is cancerous and, if so, whether it has spread to other parts of the body.

This breast cancer dataset was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

Factors & Dataset

1. **mean_radius**: This column represents the mean radius of the cancer cells. It is a measure of the average distance from the center to the points on the perimeter of the cell nuclei.
2. **mean_texture**: This column represents the mean texture of the cancer cells. It measures the variation in gray-scale intensities in the image, which can provide information about the smoothness or roughness of the cell nuclei.
3. **mean_perimeter**: This column represents the mean perimeter of the cancer cells. It is the total length of the boundary of the cell nuclei.
4. **mean_area**: This column represents the mean area of the cancer cells. It is the total area covered by the cell nuclei.
5. **mean_smoothness**: This column represents the mean smoothness of the cancer cells. It measures the variation in local lengths of the radius of the cell nuclei.
6. **diagnosis**: This column represents the diagnosis of the cancer cells. It is the target variable and can take two values:

    1: Malignant (cancerous)

    0: Benign (non-cancerous)

These columns provide features that can be used to predict whether a given set of cancer cell characteristics is indicative of malignant or benign tumors. Machine learning models can be trained on these features to make predictions about the diagnosis of breast cancer.

Steps involved for Machine Learning Process

1. Data Understanding: Understand the structure of the dataset, meaning of each column, including the number of features, target variable, and data types. Check the distribution of the target variable to understand the balance between malignant and benign cases

2. Data Preprocessing:

a) - Handle Missing Values: Either by imputation or removal.

b) - Feature Scaling: Scale the features if necessary to ensure that all features contribute equally to the model fitting process.

c) - Feature Encoding: If categorical variables exist, encode them into a numerical format suitable for machine learning algorithms.

3. Exploratory Data Analysis (EDA):

a) - Univariate Analysis: Analyze individual features using histograms, box plots, or density plots to understand their distributions.

b) - Bivariate Analysis: Explore relationships between pairs of features and how they correlate with the target variable.

c) - Correlation Analysis: Calculate and visualize correlations between features using correlation matrices or heatmaps.

Steps involved

4. Model Selection:

a) - **Choose Algorithms**: Select appropriate machine learning algorithms suitable for the classification task, such as logistic regression, decision trees.

5. Model Training:

a) - **Split Data**: Split the dataset into training and testing sets (and optionally, a validation set).

b) - **Train Models**: Train the selected machine learning models on the training data.

6. Model Evaluation:

a) - **Performance Metrics**: Evaluate model performance using appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

b) - **Cross-validation**: Perform cross-validation to assess model generalization and stability.

c) - **Confusion Matrix**: Analyze the confusion matrix to understand the types of errors made by the model.

Steps involved

7. Final Model Selection:

a) - Select Best Model: Choose the best-performing model based on evaluation metrics and cross-validation results.

8. Model Interpretation:

a) - Feature Importance: Interpret the model predictions and understand the importance of different features in making predictions.

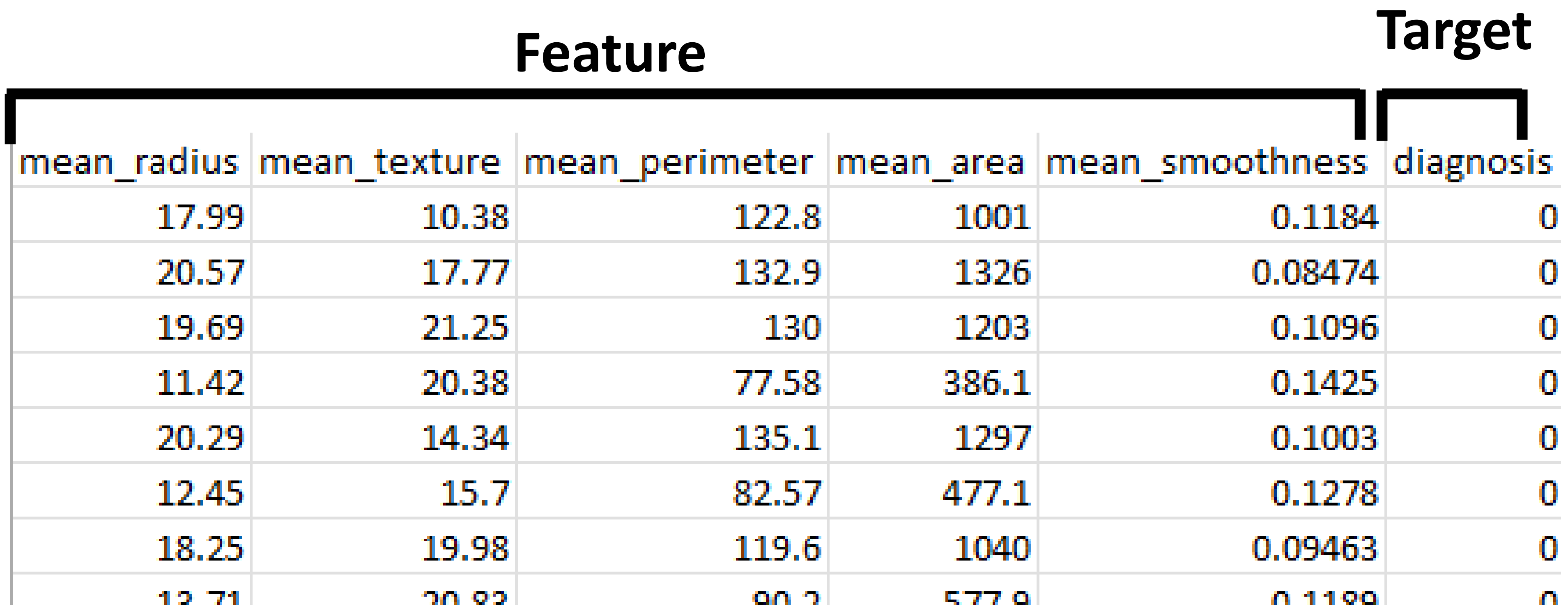
b) - Decision Boundaries: Visualize decision boundaries to understand how the model distinguishes between different classes.

9. Deployment and Monitoring (For production use):

a) - Deploy Model: Deploy the trained model into a production environment.

b) - Monitoring: Monitor the deployed model's performance and update as necessary.

Data Understanding

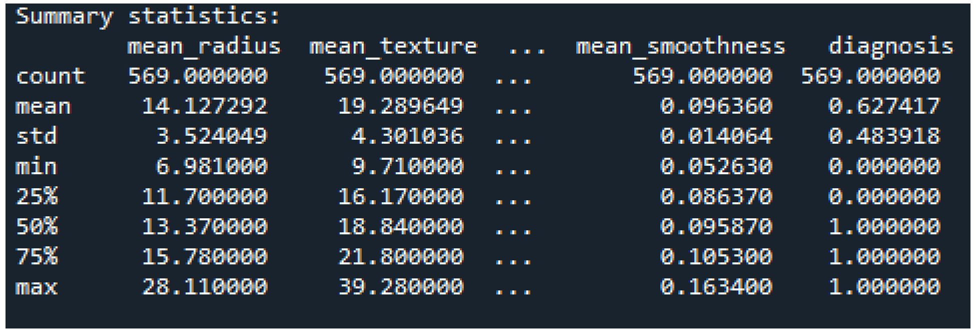

Feature
Target

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

Understanding the Data and the domain is the most important steps in any ML process.

Summary Statistics:

```
print("Summary statistics:\n", Breast_cancer_data.describe())
```

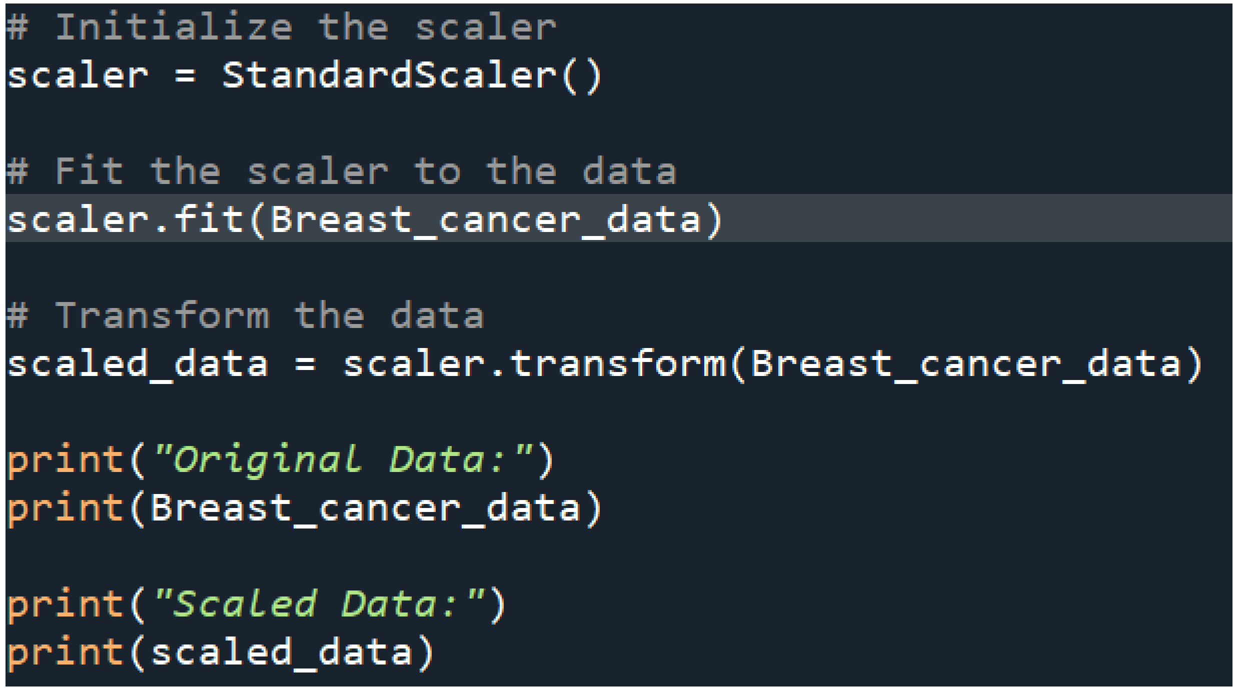


```
Summary statistics:
        mean_radius  mean_texture  ...  mean_smoothness   diagnosis
count   569.000000    569.000000  ...       569.000000  569.000000
mean     14.127292     19.289649  ...         0.096360    0.627417
std       3.524049      4.301036  ...         0.014064    0.483918
min       6.981000      9.710000  ...         0.052630    0.000000
25%      11.700000     16.170000  ...         0.086370    0.000000
50%      13.370000     18.840000  ...         0.095870    1.000000
75%      15.780000     21.800000  ...         0.105300    1.000000
max      28.110000     39.280000  ...         0.163400    1.000000
```

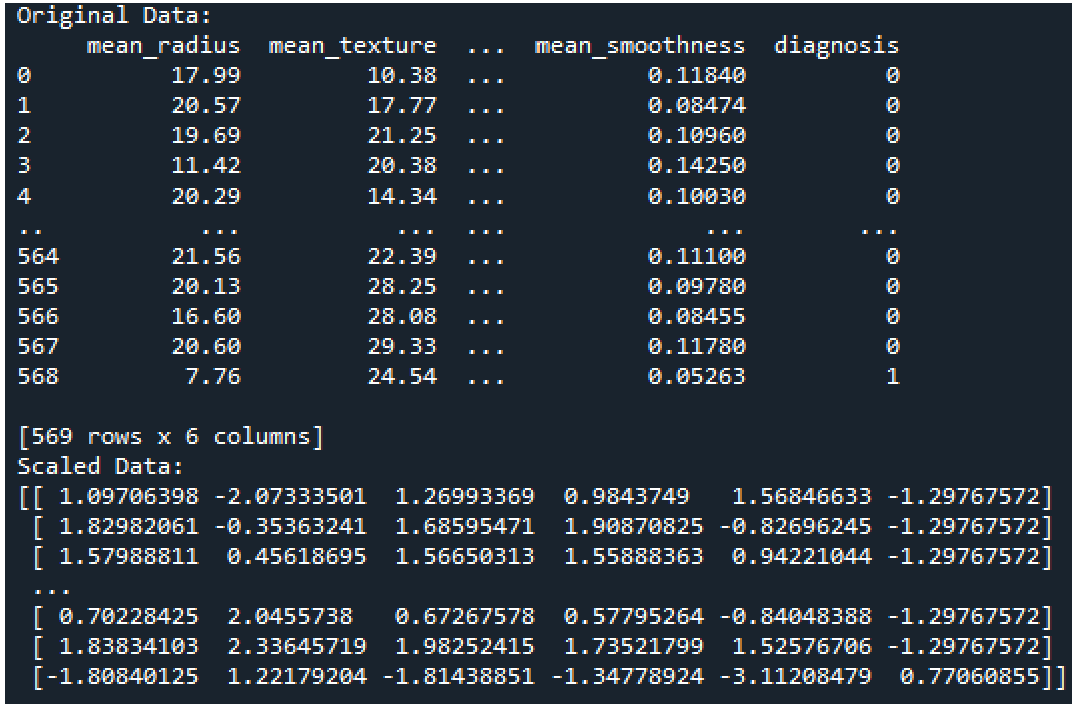
Feature Scaling

Feature scaling is a preprocessing technique used in machine learning to standardize or normalize the range of independent variables or features in the dataset. It ensures that all features have the same scale, which can be beneficial for certain machine learning algorithms that are sensitive to the scale of the input features. Feature scaling typically involves transforming the features so that they have a mean of zero and a standard deviation of one (standardization) or scaling them to a range between zero and one (normalization).

Two common methods of feature scaling are Standardization (Z-score normalization) and Normalization (Min-Max scaling)

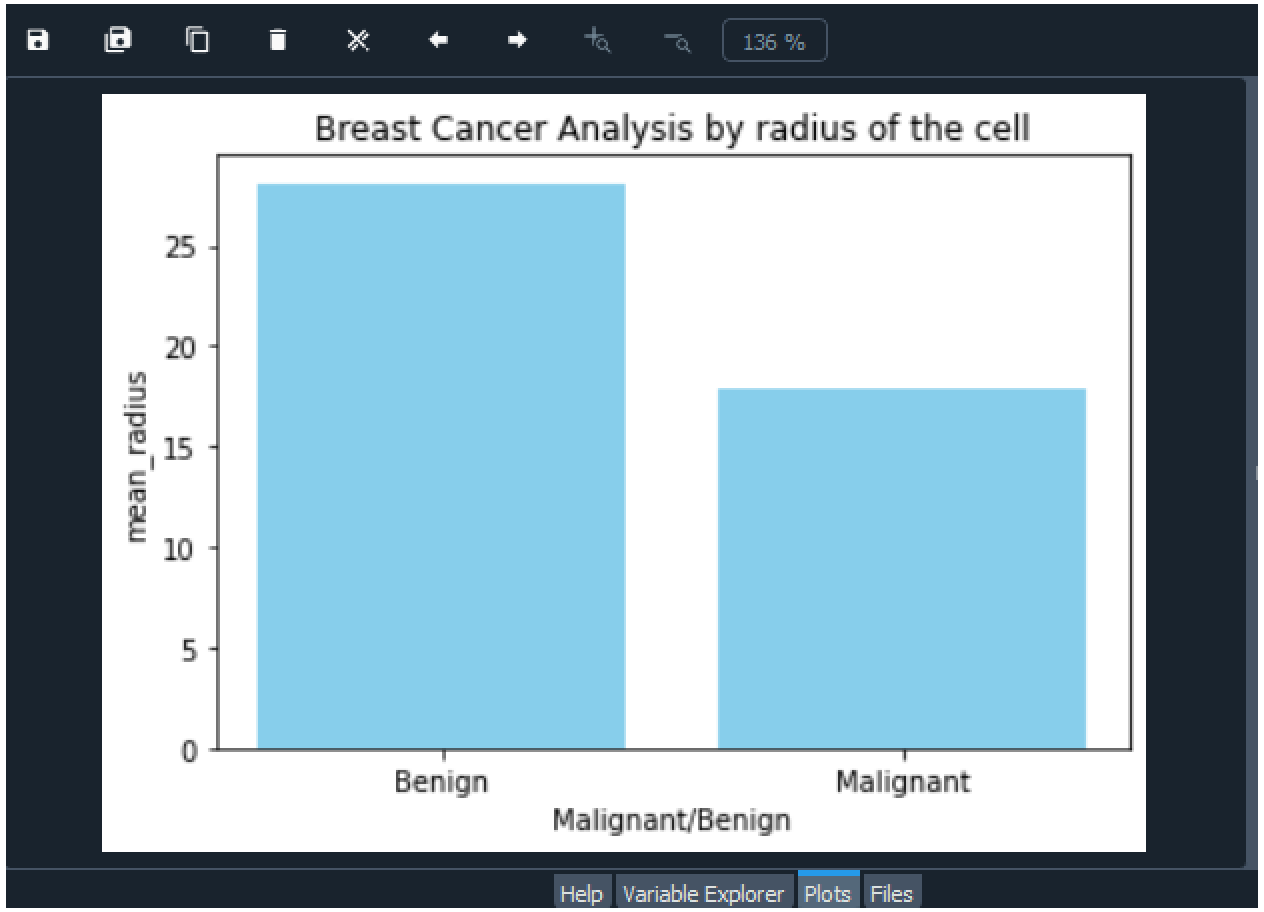


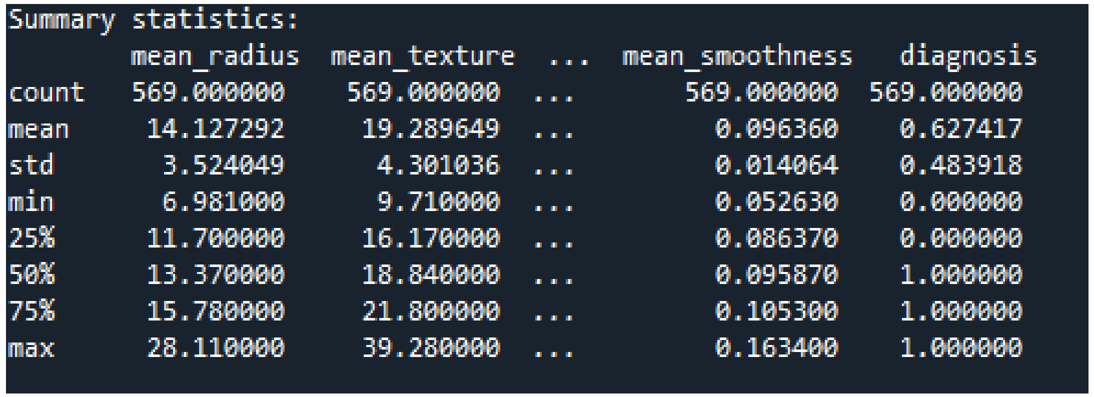
```python
# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler to the data
scaler.fit(Breast_cancer_data)

# Transform the data
scaled_data = scaler.transform(Breast_cancer_data)

print("Original Data:")
print(Breast_cancer_data)

print("Scaled Data:")
print(scaled_data)
```



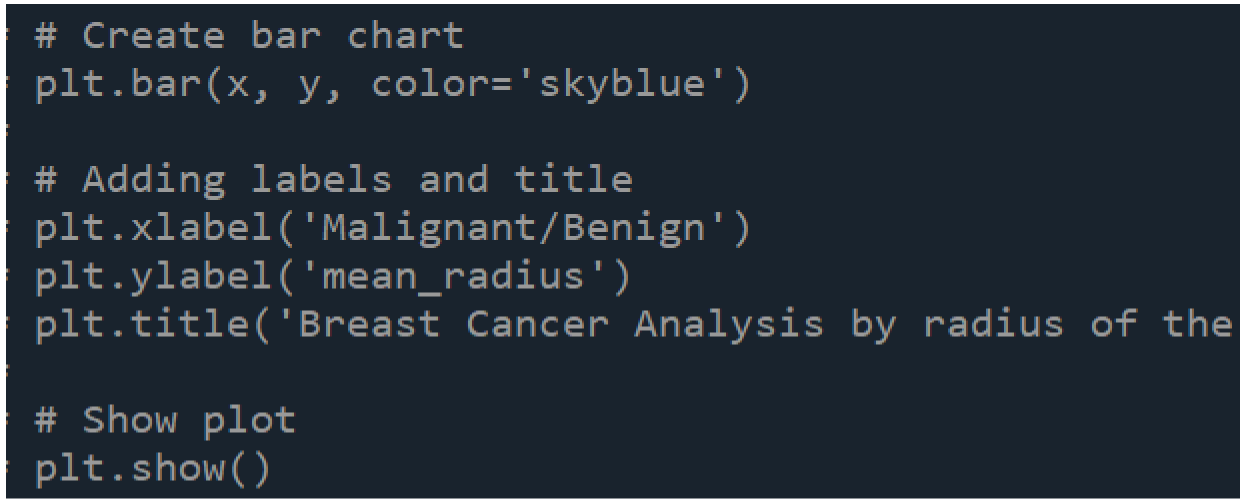
```
Original Data:
     mean_radius  mean_texture  ...  mean_smoothness  diagnosis
0          17.99         10.38  ...          0.11840          0
1          20.57         17.77  ...          0.08474          0
2          19.69         21.25  ...          0.10960          0
3          11.42         20.38  ...          0.14250          0
4          20.29         14.34  ...          0.10030          0
..           ...           ...  ...              ...        ...
564        21.56         22.39  ...          0.11100          0
565        20.13         28.25  ...          0.09780          0
566        16.60         28.08  ...          0.08455          0
567        20.60         29.33  ...          0.11780          0
568         7.76         24.54  ...          0.05263          1

[569 rows x 6 columns]
Scaled Data:
[[ 1.09706398 -2.07333501  1.26993369  0.9843749   1.56846633 -1.29767572]
 [ 1.82982061 -0.35363241  1.68595471  1.90870825 -0.82696245 -1.29767572]
 [ 1.57988811  0.45618695  1.56650313  1.55888363  0.94221044 -1.29767572]
 ...
 [ 0.70228425  2.0455738   0.67267578  0.57795264 -0.84048388 -1.29767572]
 [ 1.83834103  2.33645719  1.98252415  1.73521799  1.52576706 -1.29767572]
 [-1.80840125  1.22179204 -1.81438851 -1.34778924 -3.11208479  0.77060855]]
```

Exploratory Data Analysis – Bar Chart





```
Summary statistics:
       mean_radius  mean_texture  ...  mean_smoothness   diagnosis
count   569.000000    569.000000  ...       569.000000  569.000000
mean     14.127292     19.289649  ...         0.096360    0.627417
std       3.524049      4.301036  ...         0.014064    0.483918
min       6.981000      9.710000  ...         0.052630    0.000000
25%      11.700000     16.170000  ...         0.086370    0.000000
50%      13.370000     18.840000  ...         0.095870    1.000000
75%      15.780000     21.800000  ...         0.105300    1.000000
max      28.110000     39.280000  ...         0.163400    1.000000
```

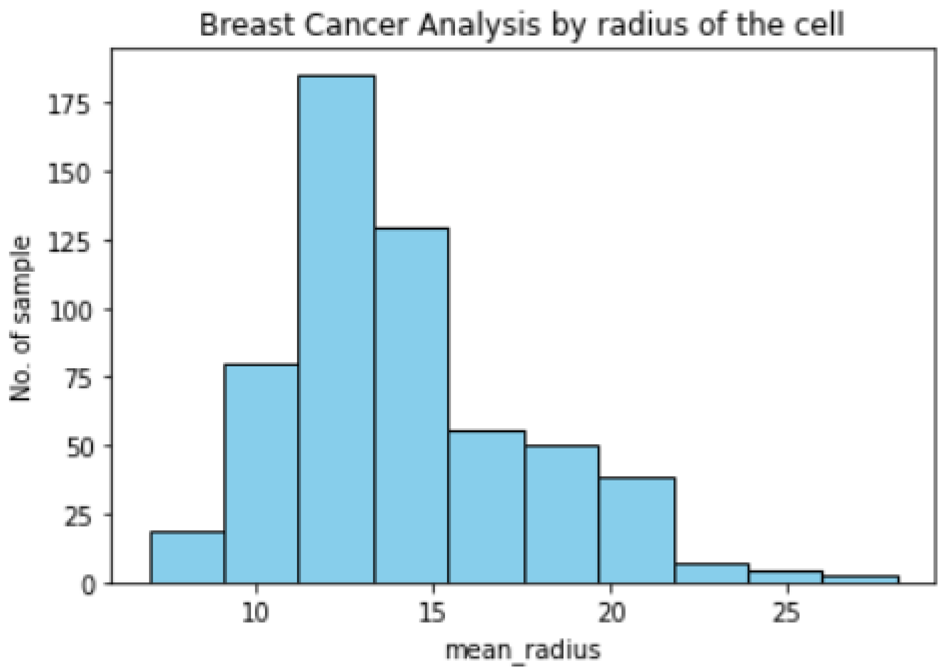


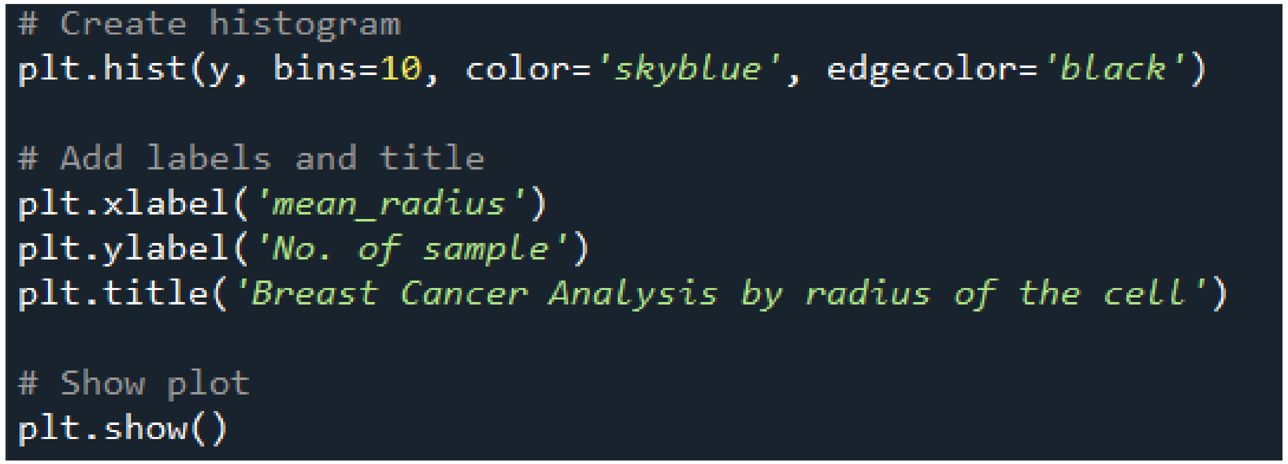
```
# Create bar chart
plt.bar(x, y, color='skyblue')

# Adding labels and title
plt.xlabel('Malignant/Benign')
plt.ylabel('mean_radius')
plt.title('Breast Cancer Analysis by radius of the

# Show plot
plt.show()
```

Bar Chart Plot to understand the weightage of Malignancy on the Variables

Exploratory Data Analysis – Histogram Plot

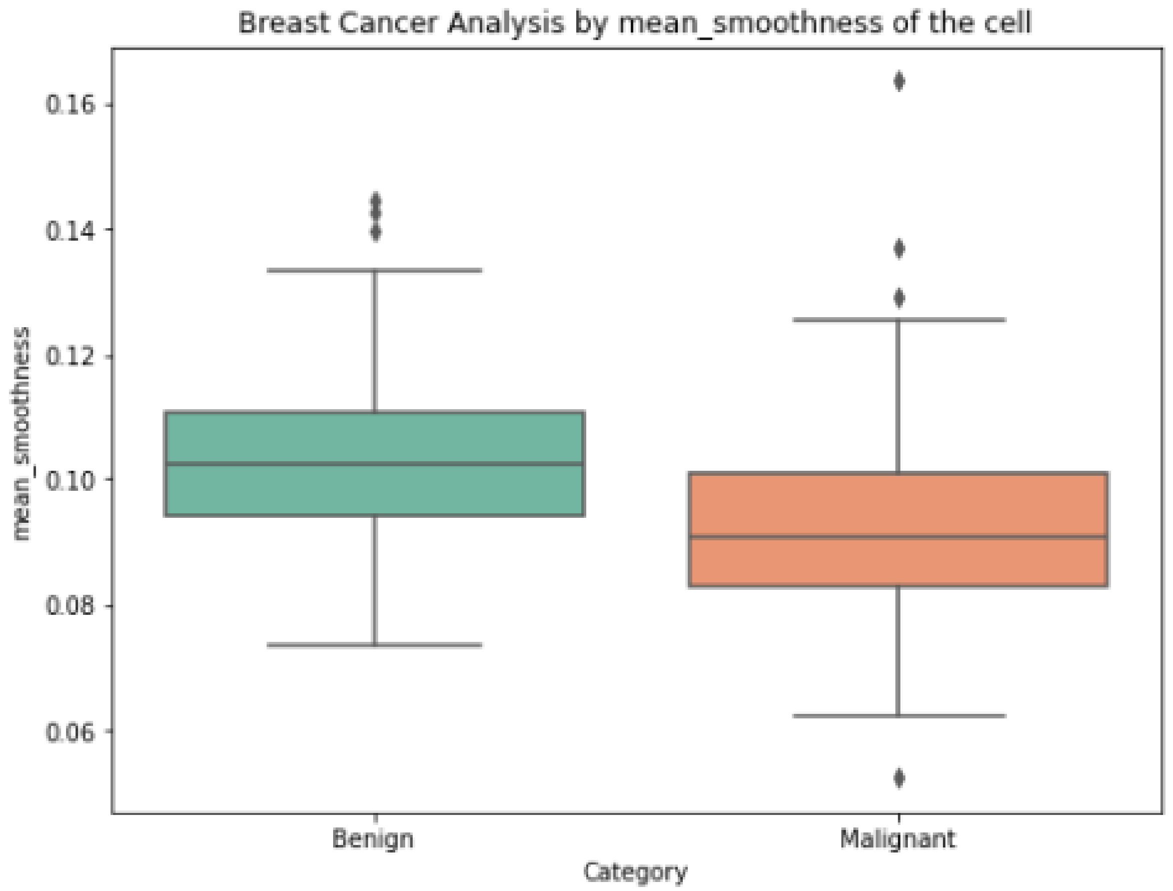
Histograms are important in exploratory data analysis (EDA) for "Visualizing Data Distribution", "Identifying Patterns and Trends", "Comparing Distributions" and "Assessing Data Quality". Selecting an appropriate bin size is crucial for effectively representing the data distribution without oversimplifying or overcomplicating it. Here I have selected 10.





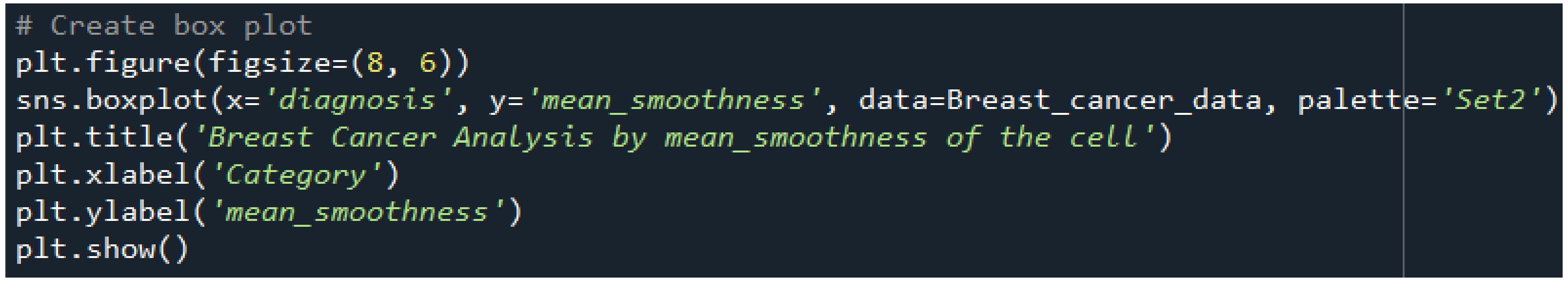
```
# Create histogram
plt.hist(y, bins=10, color='skyblue', edgecolor='black')

# Add labels and title
plt.xlabel('mean_radius')
plt.ylabel('No. of sample')
plt.title('Breast Cancer Analysis by radius of the cell')

# Show plot
plt.show()
```

Exploratory Data Analysis – Box Plot



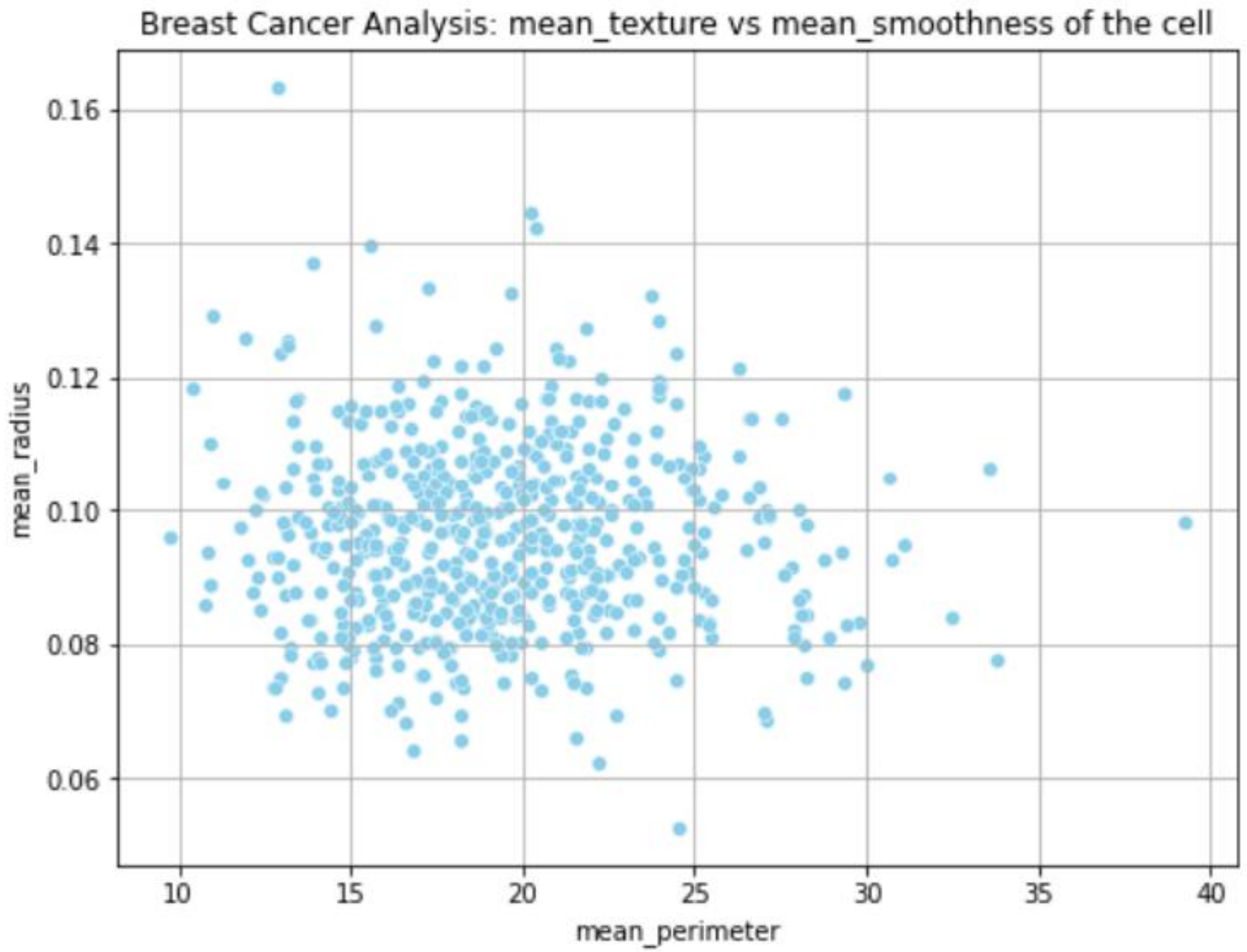
Box plots are important in Exploratory Data Analysis (EDA) to "**Visualize Distribution**", "**Identify Outliers**", "**Compare Distributions**", "**Assess Skewness**", "**Handling Missing Values**" and "**Visualizing Spread and Variability**".

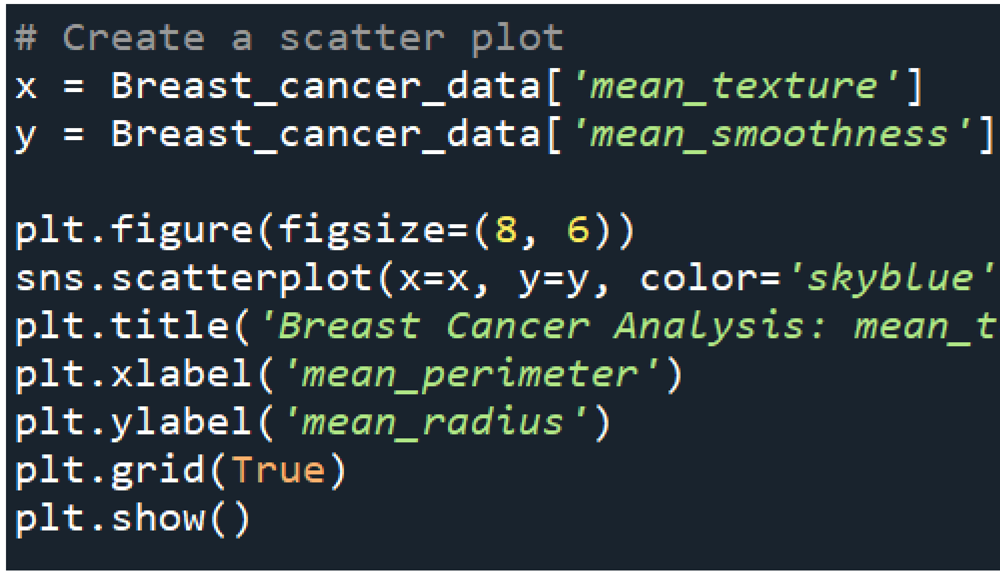
In case of Malignancy mean(smoothness) is less compared to Benign



```
# Create box plot
plt.figure(figsize=(8, 6))
sns.boxplot(x='diagnosis', y='mean_smoothness', data=Breast_cancer_data, palette='Set2')
plt.title('Breast Cancer Analysis by mean_smoothness of the cell')
plt.xlabel('Category')
plt.ylabel('mean_smoothness')
plt.show()
```

Exploratory Data Analysis : Bivariate Analysis

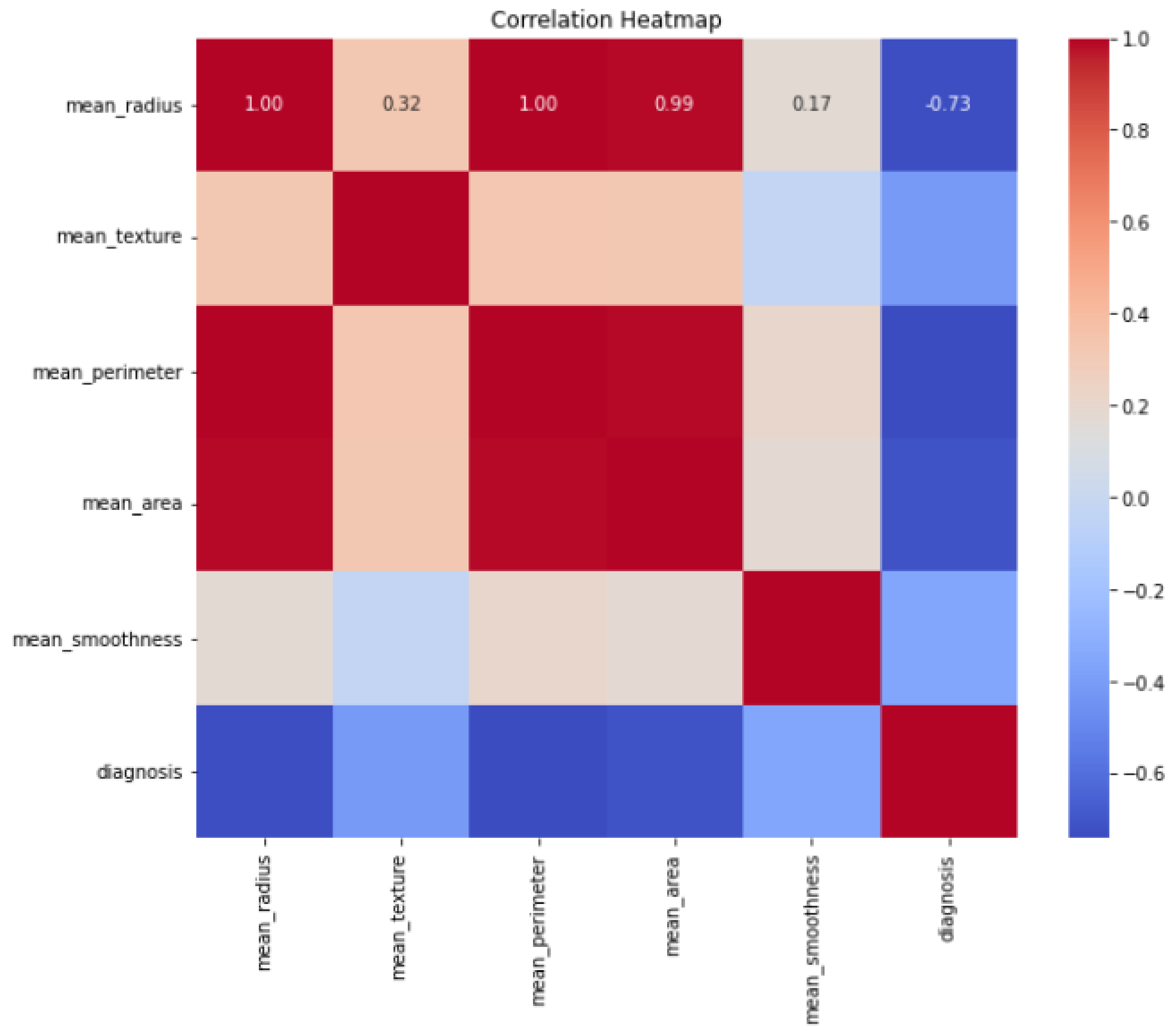


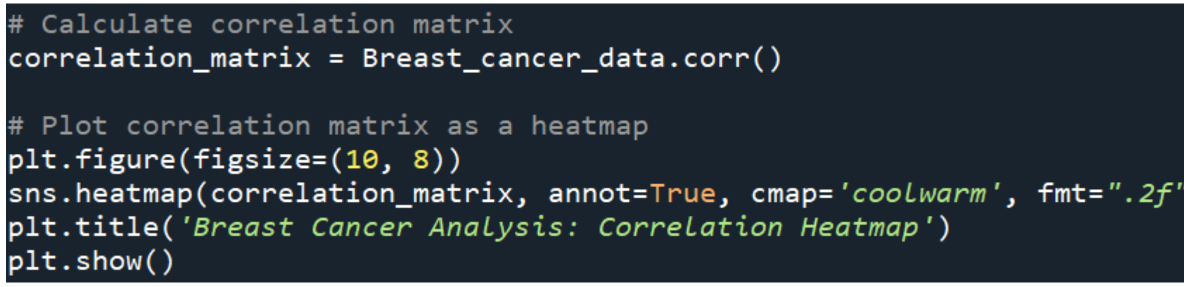


```
# Create a scatter plot
x = Breast_cancer_data['mean_texture']
y = Breast_cancer_data['mean_smoothness']

plt.figure(figsize=(8, 6))
sns.scatterplot(x=x, y=y, color='skyblue'
plt.title('Breast Cancer Analysis: mean_t
plt.xlabel('mean_perimeter')
plt.ylabel('mean_radius')
plt.grid(True)
plt.show()
```

No Definite Pattern coming out

EDA: Correlation Analysis: Heat Map





```
# Calculate correlation matrix
correlation_matrix = Breast_cancer_data.corr()

# Plot correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f
plt.title('Breast Cancer Analysis: Correlation Heatmap')
plt.show()
```

A correlation analysis heatmap visualizes the correlation matrix of a dataset using colors to represent the strength and direction of the relationships between pairs of variables. In summary, correlation analysis heatmaps provide a visually appealing and informative way to explore the relationships between variables in a dataset. They are valuable tools for identifying patterns, trends, and dependencies among variables, aiding in feature selection, and assessing collinearity in multivariate analysis.

Model Selection : Logistics Regression

Choosing a logistic regression model over other models for breast cancer data analysis depends on various factors, including the dataset characteristics, interpretability, and performance requirements. Here are some considerations for choosing logistic regression:

1. **Interpretability**: Logistic regression provides coefficients that are interpretable as log-odds ratios. This makes it easier to interpret the impact of each feature on the probability of the target variable (in this case, breast cancer diagnosis).
2. **Linear Relationship**: Logistic regression assumes a linear relationship between the independent variables and the log-odds of the target variable. If the relationship is approximately linear or can be transformed to be linear, logistic regression can be effective.
3. **Feature Importance**: Logistic regression provides insight into the importance of features through the magnitude and sign of coefficients. This can help in identifying key features associated with breast cancer diagnosis.
4. **Computational Efficiency**: Logistic regression is computationally efficient, especially with large datasets compared to some more complex models like neural networks or ensemble methods.
5. **Binary Classification**: Logistic regression is specifically designed for binary classification problems, which is suitable for predicting breast cancer diagnosis (malignant or benign).

Limitation of Logistics Regression

1. **Assumption of Linearity**: Logistic regression assumes a linear relationship between the independent variables and the log-odds of the target variable. If the relationship is non-linear, logistic regression may not perform well.
2. **Limited Capacity**: Logistic regression may not capture complex relationships present in the data as effectively as more complex models like decision trees or neural networks.
3. **Highly Imbalanced Data**: If the dataset is highly imbalanced (i.e., one class significantly outnumbering the other), logistic regression may not perform well without additional techniques such as class weighting or resampling.

logistic regression can be a good choice for breast cancer data analysis if the relationships are approximately linear.

Model Training (Split & Train)

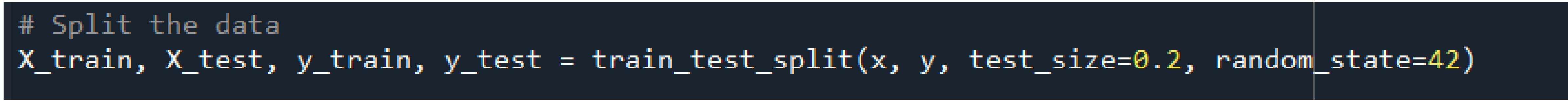
1. Split the Data

a) **Simple Holdout Method**: The dataset is split into two parts: a training set and a testing set. Typically, a larger portion of the data (e.g., 70-80%) is used for training, and the remaining portion is used for testing.

b) **K-Fold Cross-Validation**: The dataset is divided into k folds. The model is trained k times, each time using k-1 folds for training and the remaining fold for testing. The final performance metric is the average performance across all folds.

c) **Stratified Sampling**: This technique ensures that the distribution of classes in the training and testing sets is similar to the distribution in the original dataset. It is particularly useful for imbalanced datasets.

Among these techniques**, random splitting** is often preferred for several reasons, such as Ease of Implementation, Statistical Soundness, Generalization, Reproducibility and Flexibility

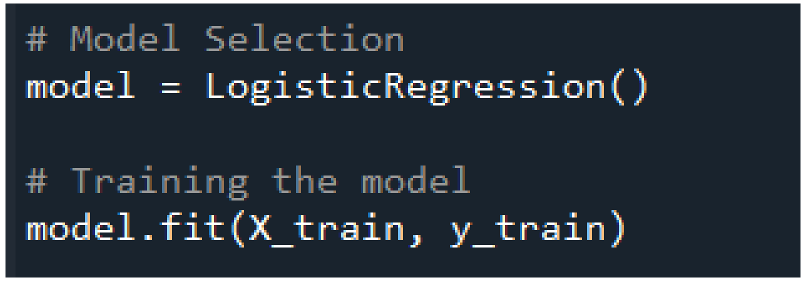


```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Model Training (Split & Train)

Model fitting is the process of training a machine learning model to learn patterns and relationships within the training data. It involves adjusting the parameters of the model to minimize the difference between the predicted outputs and the actual outputs.

1. **Initialization:** Initialize the model with some initial parameters or weights. The choice of initial parameters may depend on the specific algorithm used for training the model.
2. **Training:** Train the model using the training data. During training, the model adjusts its parameters iteratively to minimize a loss function, which measures the difference between the predicted outputs and the actual outputs. The training process typically involves the following steps:
3. **Validation:** After each training iteration or epoch, evaluate the model's performance on a separate validation dataset (if available) to monitor its generalization ability and prevent overfitting. Adjust the model hyperparameters (e.g., learning rate, regularization strength) based on the validation performance.
4. **Stopping Criterion:** Determine when to stop training the model based on predefined stopping criteria, such as convergence of the loss function or reaching a maximum number of epochs.
5. **Final Evaluation:** Once training is complete, evaluate the model's performance on a separate test dataset to assess its generalization ability on unseen data.



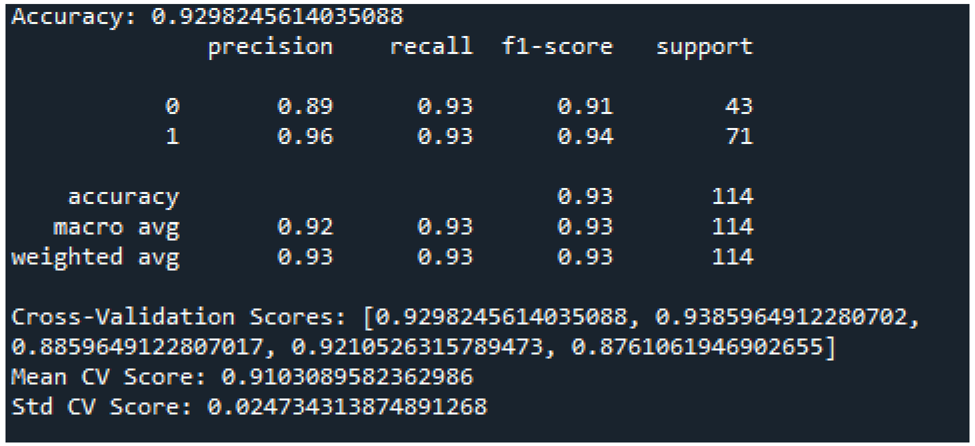
```
# Model Selection
model = LogisticRegression()

# Training the model
model.fit(X_train, y_train)
```

Model Validation, Accuracy & interpretation

Model Accuracy: Accuracy measures the proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances. It is calculated as the ratio of the number of correct predictions to the total number of predictions. Accuracy is a useful metric when the classes in the dataset are balanced (i.e., similar number of instances for each class). However, accuracy can be misleading when dealing with imbalanced datasets, where one class significantly outnumbers the other. In such cases, a high accuracy value may not necessarily indicate good model performance if the majority class dominates the predictions.

Precision: Precision measures the proportion of true positive predictions out of all positive predictions (true positives and false positives). It focuses on the accuracy of positive predictions and helps assess the model's ability to minimize false positive predictions. Precision is particularly important in scenarios where false positives are costly or undesirable (e.g., in medical diagnosis or fraud detection).

Precision is calculated as the ratio of true positives to the sum of true positives and false positives.



```
Accuracy: 0.9298245614035088
              precision    recall  f1-score   support

           0       0.89      0.93      0.91        43
           1       0.96      0.93      0.94        71

    accuracy                           0.93       114
   macro avg       0.92      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114

Cross-Validation Scores: [0.9298245614035088, 0.9385964912280702,
0.8859649122807017, 0.9210526315789473, 0.8761061946902655]
Mean CV Score: 0.9103089582362986
Std CV Score: 0.024734313874891268
```

Model Validation and Accuracy

Recall, also known as sensitivity or true positive rate, is an evaluation metric used in classification tasks to measure the proportion of actual positive instances (true positives) that are correctly identified by the mode. Recall = True Positive / (True Positive +False Negative)

True Positives (TP) are the instances that are correctly classified as positive by the model.

False Negatives (FN) are the instances that are actually positive but are incorrectly classified as negative by the model. A high recall value indicates that the model is effectively identifying a large proportion of actual positive instances.

F1 Score: The F1 score is the harmonic mean of precision and recall. It combines both precision and recall into a single metric and is particularly useful when there is an imbalance between the number of positive and negative instances in the dataset. The F1 score ranges from 0 to 1, where a higher value indicates better model performance. It reaches its best value at 1 and its worst at 0.

Support: Support represents the number of actual occurrences of each class in the dataset. It provides information about the distribution of classes in the dataset and is useful for understanding the reliability of the evaluation metrics.

In the context of classification, support is often displayed as the number of instances (or samples) belonging to each class.

For example, if a binary classification task has two classes (positive and negative), the support for the positive class represents the number of positive instances in the dataset.

Model Deployment & Monitoring

Since it’s a school project, hence model deployment in Production and Monitoring is kept out of scope for now.

Thank You