

2025

# Breast Cancer

Breakthroughs in Cancer Treatment: Targeted Therapy and Immunotherapy Explained

Get Started



---

# Data Overview

- **Description:** This dataset includes characteristics of breast cell nuclei for classifying tumors as malignant or benign.
- **Goal:** Build a model to predict if a breast tumor is malignant or benign.



# About the Project

## Approach:

1. Load Data
2. Data preprocessing (handling nulls, scaling, encoding).
3. Feature scaling with StandardScaler.
4. Splitting dataset into train/test for evaluation.
5. Training multiple ML models (Logistic Regression, XGBClassifier, RandomForestClassifier, etc.).





# Exploratory Data Analysis (EDA)

- Dataset contains 569 samples and 30 numerical features (tumor measurements).
- **Target variable:** Diagnosis → Benign (0) vs Malignant (1).
- Class distribution: Slightly imbalanced but manageable (Benign ~62%, Malignant ~38%).
- **Feature distributions:** Malignant tumors generally show higher values in radius, perimeter, area, and concavity compared to benign..
- **Countplot:** The dataset shows more benign cases compared to malignant cases, indicating a slight imbalance.
- **Barplot:** Malignant tumors have a higher average radius than benign tumors, showing that tumor size is a strong predictor.
- **Correlation Heatmap:** Features such as radius\_mean, perimeter\_mean, and area\_mean are strongly correlated, highlighting relationships among tumor size features.

# Project Flow

- Data Loading
- Data Preprocessing
- EDA
- Feature Scaling
- Model Selection & Training
- Evaluation Metrics
- Conclusions





# Important Libraries

- Numpy
- Pandas
- Matplotlib.pyplot
- seaborn

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

# Data Loading

- There are 33 columns and 569 rows in dataset
- Data type: float64(31), int64(1), object(1)

```
data = pd.read_csv(r'D:\ML_Projects\breast_cancer_data.csv')  
data.head()
```

```
data.shape
```

```
(569, 33)
```



# Data Preprocessing

- Checking Duplicates values

```
# checking for duplicates values  
data.duplicated().sum()
```

- Drop Unwanted Columns

```
# • Drop unwanted columns  
data.drop(columns=['Unnamed: 32'], inplace=True, axis = 1)
```



# Data Preprocessing

- Handling Categorical Values

```
# Handling categorical values - label encoding  
  
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
data["diagnosis"] = le.fit_transform(data["diagnosis"])
```

- + • Seprating Feature and Target

```
# seprating feature and target  
features = data.drop("diagnosis", axis=1)  
target = data["diagnosis"]  
data.head()
```



# Data Preprocessing

- Handling Imbalance data

```
import sys

!{sys.executable} -m pip install imbalanced-learn
```

```
# • Handling Imbalance data
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler()
features, target = ros.fit_resample(features, target)
```

```
target.value_counts()
```

```
diagnosis
```

```
1    357
```

```
0    357
```

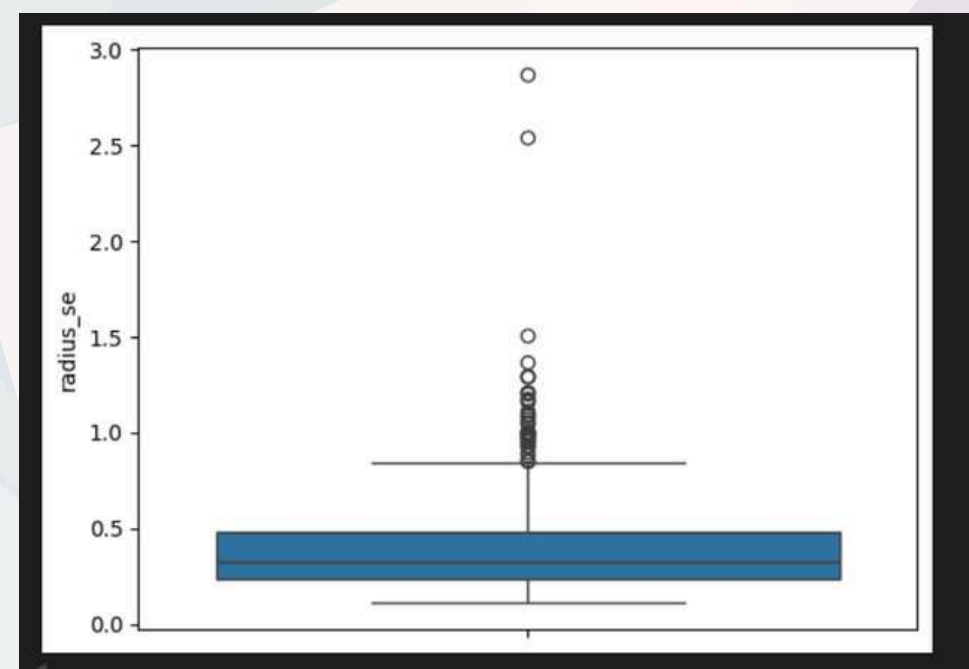
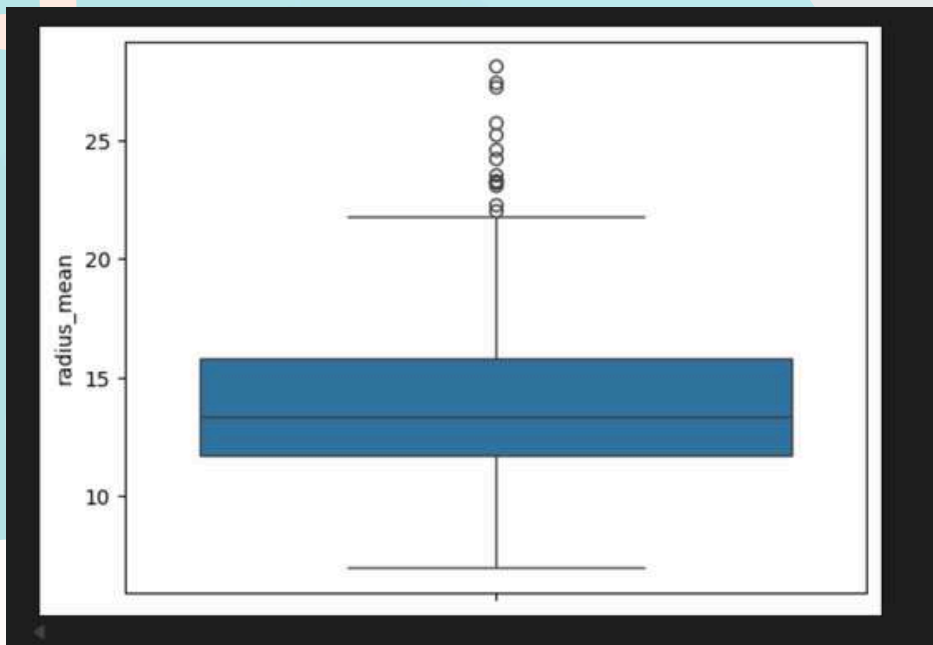
```
Name: count, dtype: int64
```



# Exploratory Data Analysis (EDA)

- Checking Outliers

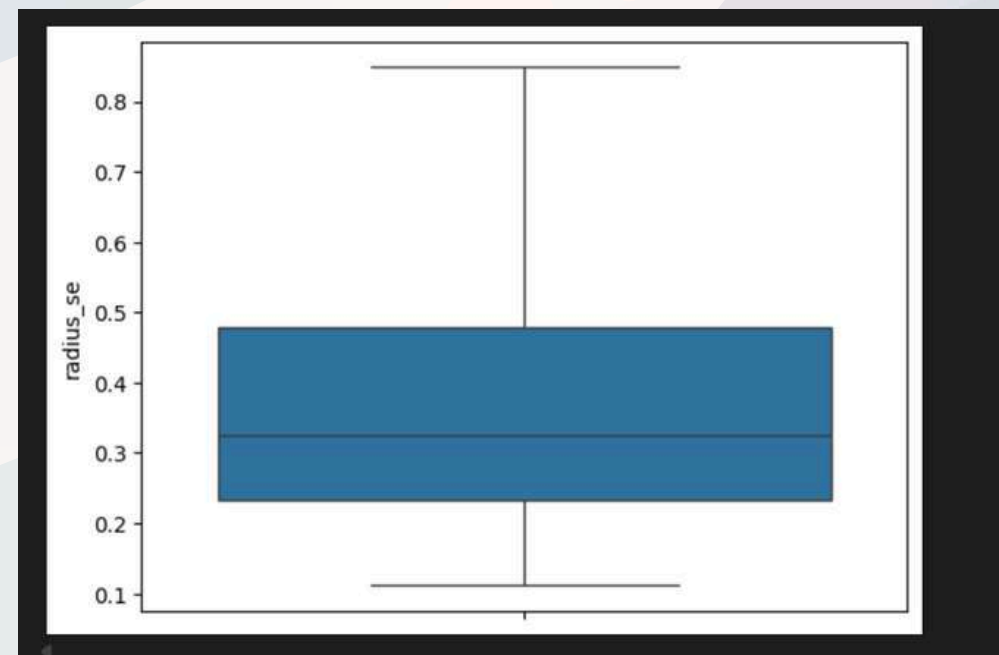
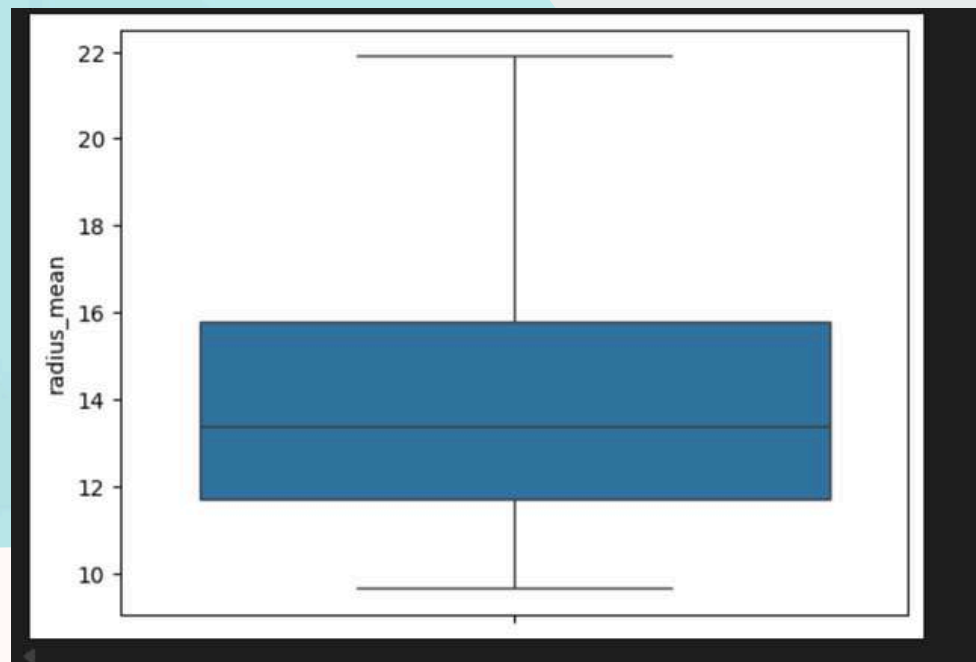
```
# checking outliers using boxplot
for i in data.select_dtypes(include=["object", "int64", "float64"]).columns:
    sns.boxplot(data[i])
    plt.show()
```



# Exploratory Data Analysis (EDA)

- Handling Outliers using IQR

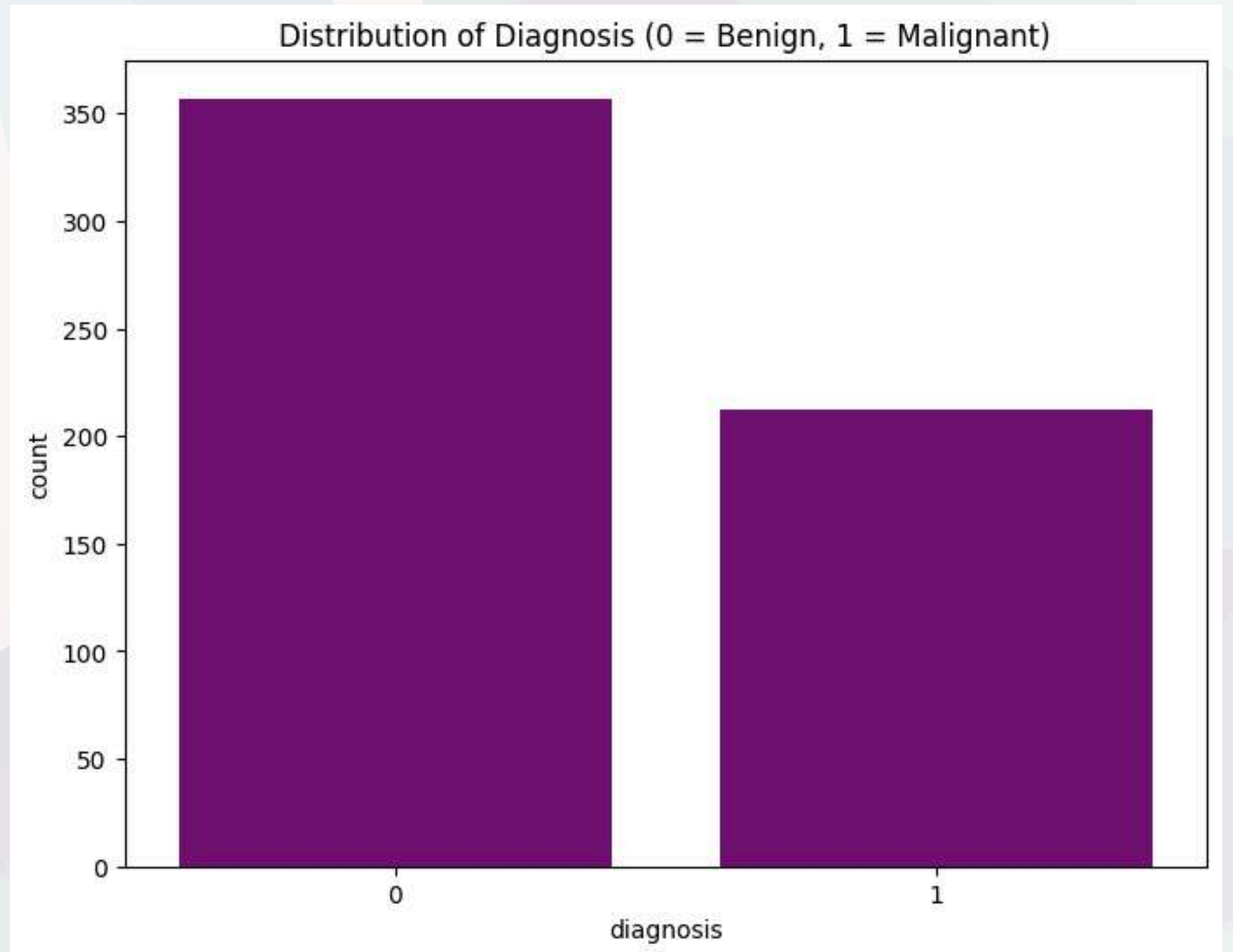
```
# handling outliers using IQR method
for i in data.select_dtypes(include=["int64", "float64"]).columns:
    q1 = data[i].quantile(0.25)
    q3 = data[i].quantile(0.75)
    iqr = q3 - q1
    ul = q3 + 1.5 * iqr
    ll = q3 - 1.5 * iqr
    data[i] = data[i].clip(lower=ll, upper=ul)
    sns.boxplot(data[i])
    plt.show()
```



# Visualization

- Distribution of Diagnosis

```
# Distribution of target variable
plt.figure(figsize=(8,6))
sns.countplot(x="diagnosis", data=data, color= "purple")
plt.title("Distribution of Diagnosis (0 = Benign, 1 = Malignant)")
plt.show()
```



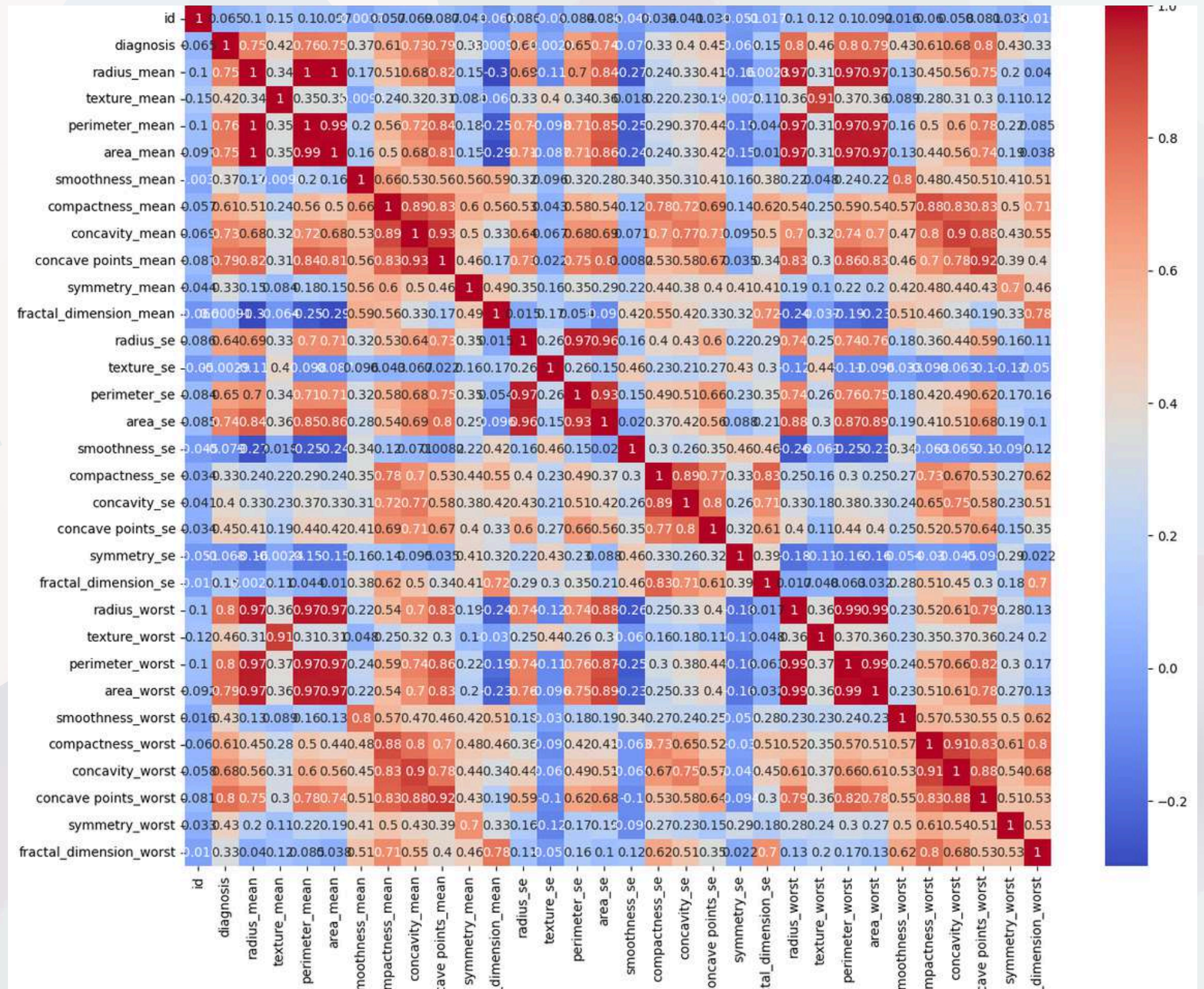
# Visualization:

- Feature Correlation

TO check which features are strongly related to each other.

Click to add a breakpoint

```
plt.figure(figsize=(15,12))
sns.heatmap(data.corr(), cmap="coolwarm", annot=True)
plt.title("Feature Correlation Heatmap")
plt.show()
```

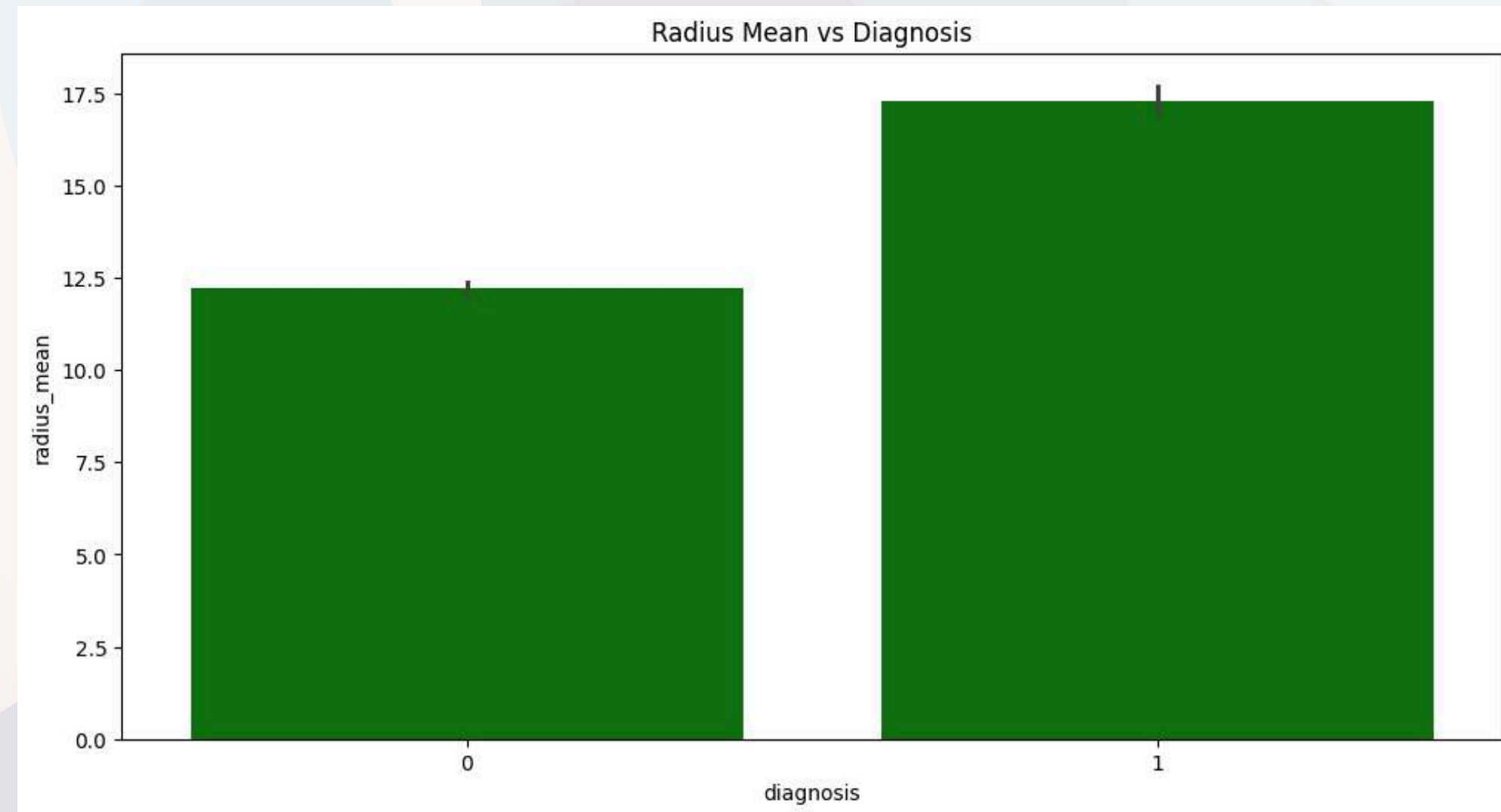


# Visualization:

## Radius Mean vs Diagnosis:

This barplot shows that malignant tumors have a higher average radius compared to benign tumors, which makes radius\_mean an important feature for prediction

```
# Radius Mean vs Diagnosis
plt.figure(figsize=(12,6))
sns.barplot(x="diagnosis", y="radius_mean", color= "green", data=data)
plt.title("Radius Mean vs Diagnosis")
plt.show()
```



# Visualization

- **Feature Scaling** → Standardizes feature values (e.g., using StandardScaler) to ensure all variables contribute equally to the model

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
features = ss.fit_transform(features)
```



# Model Selection & Fitting:

**Train-Test Split** → Divides the dataset into training and testing sets to evaluate model performance on unseen data

```
# train test split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size = 0.2, random_state = 42)
```

Generate Code Markdown

```
y_train.head()
```

264	1
56	1
204	0
571	1
527	0

Name: diagnosis, dtype: int64



# Model Fitting

## 1. LogisticRegression

- Simple, interpretable model that works well for linearly separable data.

## 2. DecisionTreeClassifier

- Easy-to-understand tree structure, captures non-linear relationships.

## 3. XGBClassifier

- Powerful gradient boosting method, handles imbalance & improves accuracy.

## 4. RandomForestClassifier

- Ensemble of decision trees, reduces overfitting, improves stability.

```
# Model Building
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, y_train)
```

▼ LogisticRegression ⓘ ?

► Parameters

```
from sklearn.tree import DecisionTreeClassifier
d_tree = DecisionTreeClassifier()
d_tree.fit(x_train, y_train)
```

▼ DecisionTreeClassifier ⓘ ?

► Parameters

```
from xgboost import XGBClassifier
xg_boost = XGBClassifier()
xg_boost.fit(x_train, y_train)
```

▼ XGBClassifier ⓘ ?

► Parameters

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
```

▼ RandomForestClassifier ⓘ ?

► Parameters



## • Model Evaluation :

```
# Model Evaluation
from sklearn.metrics import classification_report, f1_score, accuracy_score, confusion_matrix
acc_train = []
acc_test = []

def accuracy(model):
    model.fit(x_train, y_train)
    y_pred_train = model.predict(x_train)
    y_pred_test = model.predict(x_test)

    acc_train.append(accuracy_score(y_train, y_pred_train))
    acc_test.append(accuracy_score(y_test, y_pred_test))

    print("***Training Data**")

    print("Accuracy Train:", accuracy_score(y_train, y_pred_train))
    print("Classification Report Train:", classification_report(y_train, y_pred_train))
    print("F1 Score", f1_score(y_train, y_pred_train))
    print("Confusion Matrix", confusion_matrix(y_train, y_pred_train))

    print("*****10)
```

- **Classification Report** → Gives precision, recall, F1-score, and support for each class.
- **F1-Score** → Harmonic mean of precision & recall, best for imbalanced datasets.
- **Accuracy** → Percentage of correctly classified samples out of total samples.
- **Confusion Matrix** → Shows true vs. predicted values to analyze misclassifications.

## • Model Evaluation :

Combining both train and test accuracy to DataFrames

```
# combining both train and test accuracy dataframes  
df_concat = pd.concat([df_acc_train, df_acc_test])  
df_concat
```

	LogisticRegression	XGBClassifier	RandomForestClassifier	DecisionTreeClassifier
Train Accuracy	0.984238	1.000000	1.000000	1.000000
Test Accuracy	0.972028	0.972028	0.979021	0.958042



# Conclusion & Control Measures

2025

## Conclusion

- EDA shows clear separability between malignant and benign tumors
- Machine learning can help doctors detect cancer early
- Accurate predictions → timely treatment & reduced mortality

## Control & Prevention

1. Regular screening & self-exams
2. Maintain healthy lifestyle (diet, exercise, avoid alcohol/tobacco)
3. Awareness & education about symptoms
4. Genetic testing & counseling for high-risk individuals
5. Early medical consultation for breast changes

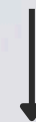




# — Thank You

---

By Surendra Gurjar



Welcome