import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn.preprocessing import StandardScaler

# Load the breast cancer dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"

column\_names = ['ID', 'Diagnosis', 'Mean Radius', 'Mean Texture', 'Mean Perimeter', 'Mean Area', 'Mean Smoothness',

'Mean Compactness', 'Mean Concavity', 'Mean Concave Points', 'Mean Symmetry', 'Mean Fractal Dimension',

'SE Radius', 'SE Texture', 'SE Perimeter', 'SE Area', 'SE Smoothness', 'SE Compactness', 'SE Concavity',

'SE Concave Points', 'SE Symmetry', 'SE Fractal Dimension', 'Worst Radius', 'Worst Texture',

'Worst Perimeter', 'Worst Area', 'Worst Smoothness', 'Worst Compactness', 'Worst Concavity',

'Worst Concave Points', 'Worst Symmetry', 'Worst Fractal Dimension']

data = pd.read\_csv(url, header=None, names=column\_names)

# Drop unnecessary columns (e.g., ID column)

data = data.drop(['ID'], axis=1)

# Convert categorical diagnosis labels to numerical (Malignant: 1, Benign: 0)

data['Diagnosis'] = data['Diagnosis'].map({'M': 1, 'B': 0})

# Separate features (X) and target variable (y)

X = data.drop('Diagnosis', axis=1)

y = data['Diagnosis']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features (important for SVM)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Initialize the SVM classifier

svm\_classifier = SVC(kernel='linear', random\_state=42)

# Train the classifier on the scaled training data

svm\_classifier.fit(X\_train\_scaled, y\_train)

# Make predictions on the test set

y\_pred = svm\_classifier.predict(X\_test\_scaled)

# Obtain the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Display the confusion matrix

print("Confusion Matrix:")

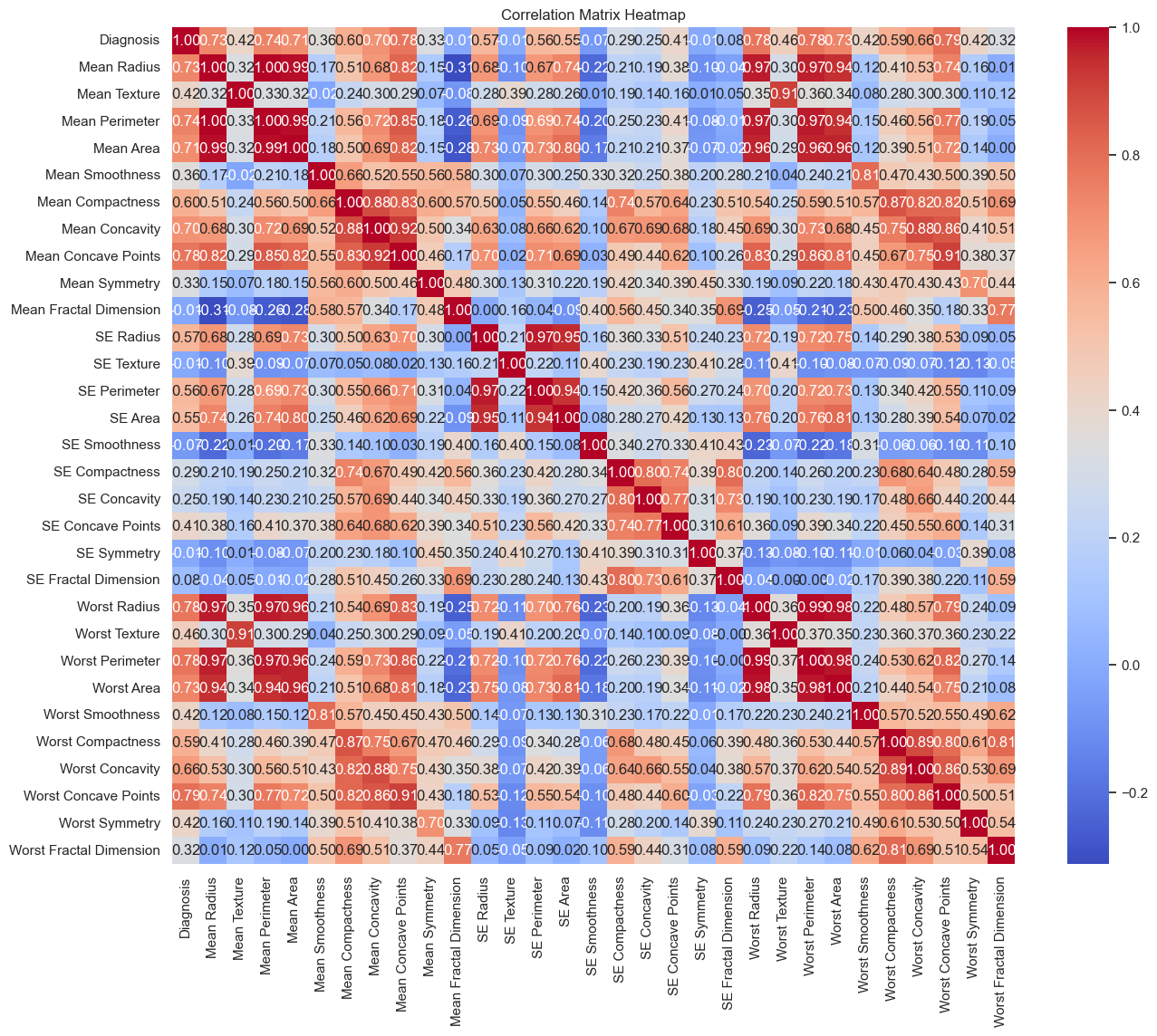
print(cm)

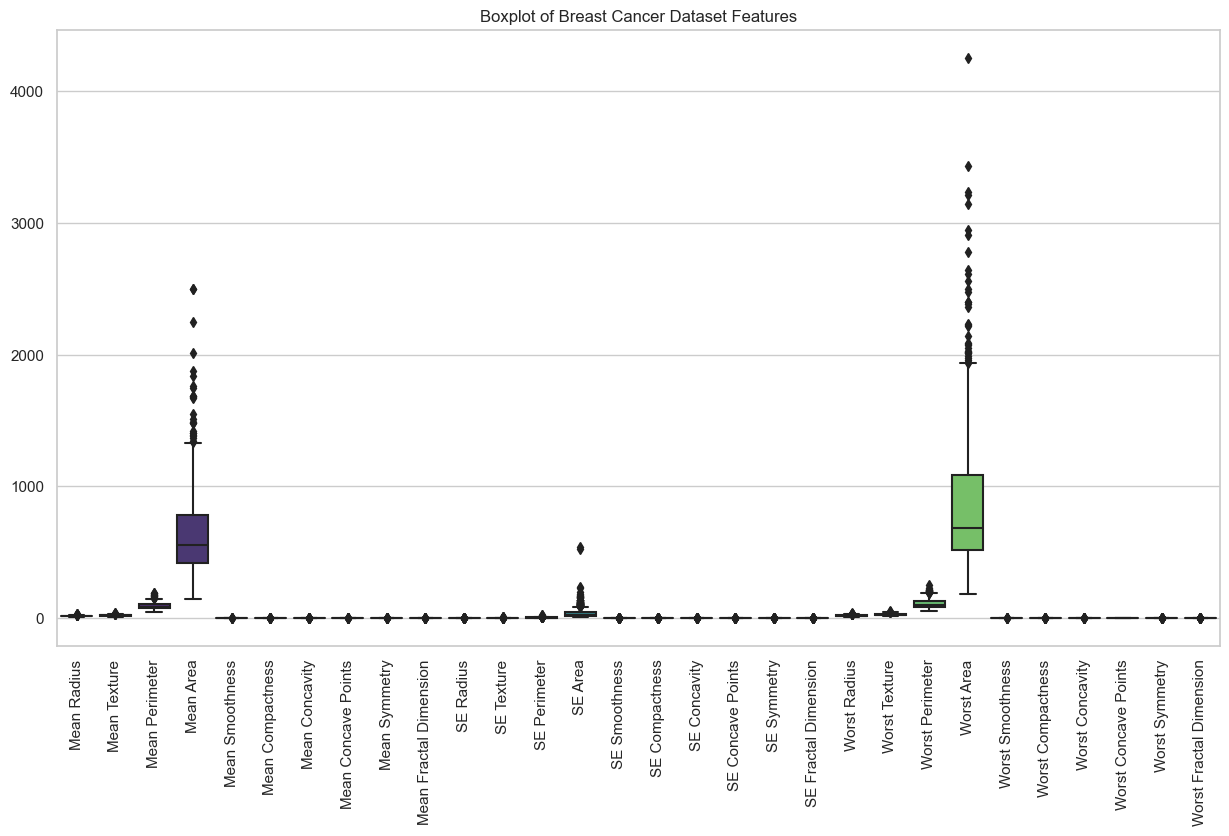
Confusion Matrix:

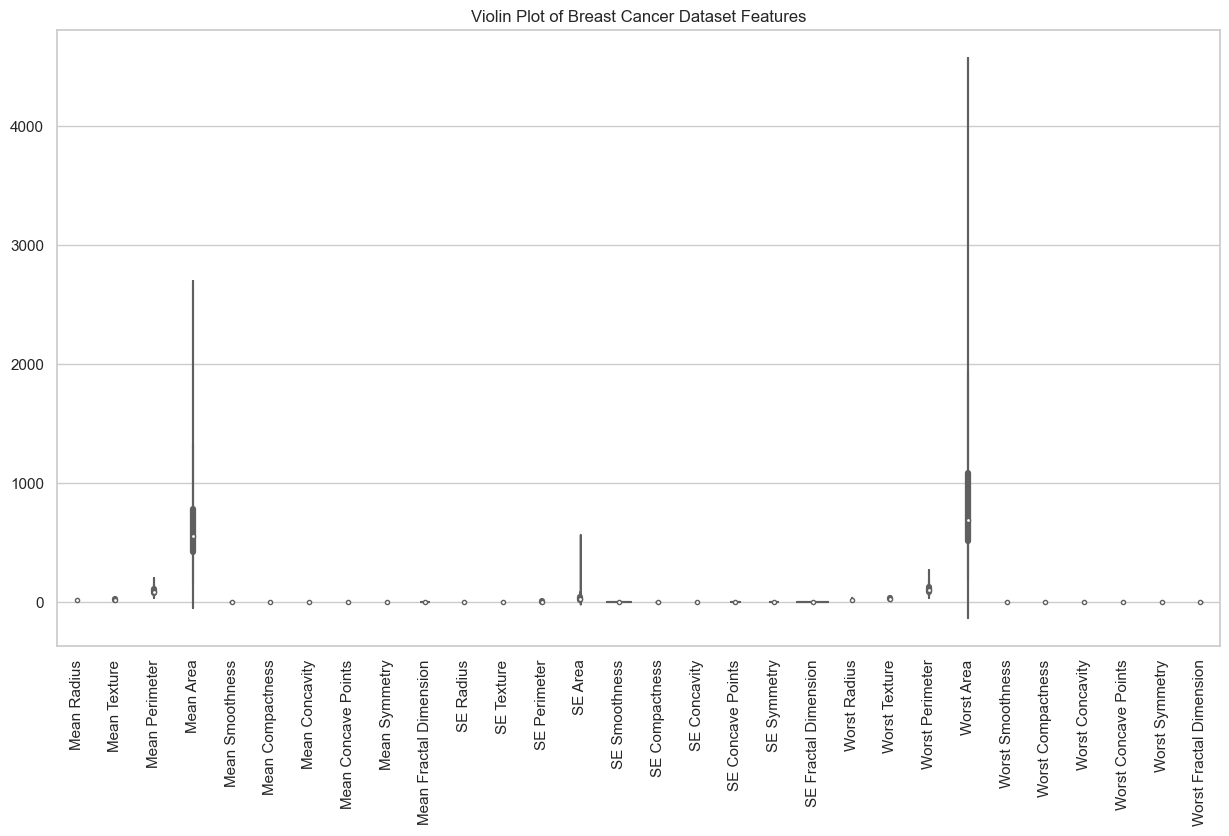
[[68 3]

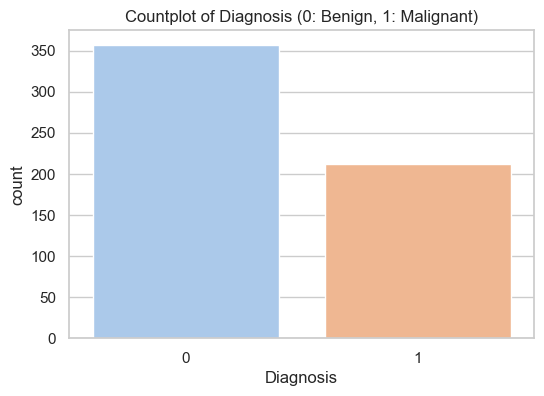
[ 2 41]]

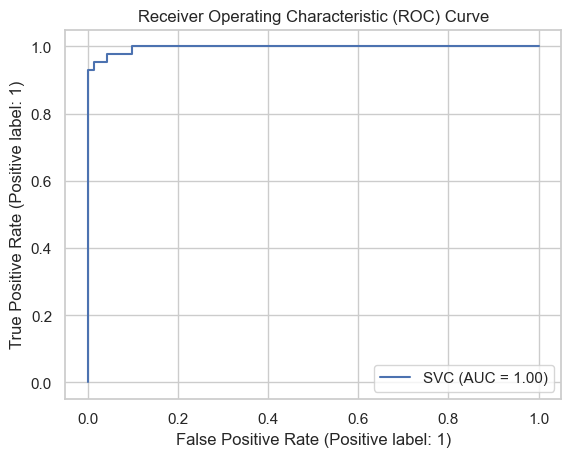


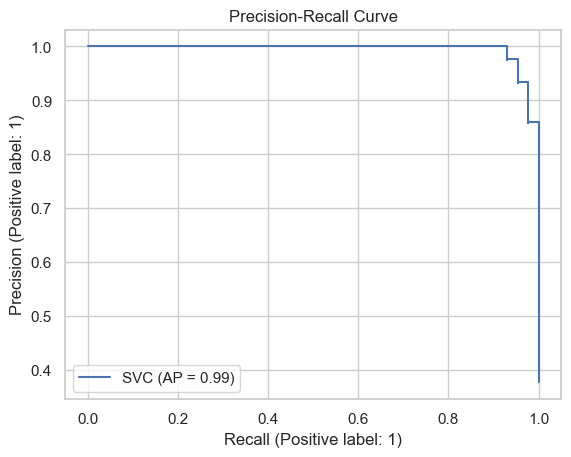












1. **Pairplot**: A pairplot that shows scatterplots for pairs of features.
2. **Correlation Matrix Heatmap**: A heatmap of the correlation matrix to visualize feature correlations.
3. **Boxplot**: Boxplots for each feature, showing the distribution of values.
4. **Violin Plot**: Violin plots for each feature, providing insights into the distribution of values.
5. **Countplot**: A countplot of the target variable to visualize the balance of classes.
6. **ROC Curve**: A Receiver Operating Characteristic (ROC) curve for the SVM classifier.
7. **Precision-Recall Curve**: A precision-recall curve for the SVM classifier.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import plot\_roc\_curve, plot\_precision\_recall\_curve

from sklearn.preprocessing import StandardScaler

# Load the breast cancer dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"

column\_names = ['ID', 'Diagnosis', 'Mean Radius', 'Mean Texture', 'Mean Perimeter', 'Mean Area', 'Mean Smoothness',

'Mean Compactness', 'Mean Concavity', 'Mean Concave Points', 'Mean Symmetry', 'Mean Fractal Dimension',

'SE Radius', 'SE Texture', 'SE Perimeter', 'SE Area', 'SE Smoothness', 'SE Compactness', 'SE Concavity',

'SE Concave Points', 'SE Symmetry', 'SE Fractal Dimension', 'Worst Radius', 'Worst Texture',

'Worst Perimeter', 'Worst Area', 'Worst Smoothness', 'Worst Compactness', 'Worst Concavity',

'Worst Concave Points', 'Worst Symmetry', 'Worst Fractal Dimension']

data = pd.read\_csv(url, header=None, names=column\_names)

# Drop unnecessary columns (e.g., ID column)

data = data.drop(['ID'], axis=1)

# Convert categorical diagnosis labels to numerical (Malignant: 1, Benign: 0)

data['Diagnosis'] = data['Diagnosis'].map({'M': 1, 'B': 0})

# Separate features (X) and target variable (y)

X = data.drop('Diagnosis', axis=1)

y = data['Diagnosis']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features (important for SVM)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Initialize the SVM classifier

svm\_classifier = SVC(kernel='linear', random\_state=42)

# Train the classifier on the scaled training data

svm\_classifier.fit(X\_train\_scaled, y\_train)

# Set style for seaborn plots

sns.set(style="whitegrid")

# 1. Pairplot

sns.pairplot(data, hue='Diagnosis', markers=["o", "s"])

plt.suptitle("Pairplot of Breast Cancer Dataset", y=1.02)

plt.show()

# 2. Correlation Matrix Heatmap

plt.figure(figsize=(15, 12))

sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Matrix Heatmap")

plt.show()

# 3. Boxplot

plt.figure(figsize=(15, 8))

sns.boxplot(data=data.drop('Diagnosis', axis=1), palette='viridis')

plt.title("Boxplot of Breast Cancer Dataset Features")

plt.xticks(rotation=90)

plt.show()

# 4. Violin Plot

plt.figure(figsize=(15, 8))

sns.violinplot(data=data.drop('Diagnosis', axis=1), palette='Set3')

plt.title("Violin Plot of Breast Cancer Dataset Features")

plt.xticks(rotation=90)

plt.show()

# 5. Countplot

plt.figure(figsize=(6, 4))

sns.countplot(x='Diagnosis', data=data, palette='pastel')

plt.title("Countplot of Diagnosis (0: Benign, 1: Malignant)")

plt.show()

# 6. ROC Curve

plt.figure(figsize=(8, 6))

plot\_roc\_curve(svm\_classifier, X\_test\_scaled, y\_test)

plt.title("Receiver Operating Characteristic (ROC) Curve")

plt.show()

# 7. Precision-Recall Curve

plt.figure(figsize=(8, 6))

plot\_precision\_recall\_curve(svm\_classifier, X\_test\_scaled, y\_test)

plt.title("Precision-Recall Curve")

plt.show()