Problem Statement

The problem statement "Breast cancer detection using ML" revolves around the application of machine learning techniques to assist in the early diagnosis of breast cancer. Breast cancer is a significant health concern worldwide, and early detection plays a critical role in improving treatment outcomes and reducing mortality rates. In this context, the problem statement can be broken down into several key components:

Classification Task: The primary goal is to develop a machine learning model capable of classifying breast tumors into two categories:

Malignant: Tumors that are cancerous and pose a health risk. Benign: Tumors that are non-cancerous and generally considered harmless.

Steps Involved For Breast Cancer Detection

1. Imported the Necessary Libraries: The first step is importing the important libraries
2. Loaded the Dataset: Then the cancer dataset was loaded from the file using pd.read() function
3. Performed Exploratory Data Analysis: Checked for missing value. Checked the shape Chevked the stastics
4. Preprocessed the Data:

- To improve the dataset for analysis, you removed the 'id' column as it was irrelevant for your purposes. This was done with `df.drop(labels=['id'], axis=1, inplace=True)` . checked for missing values in the dataset with `df.isnull().values.any()` . looked for duplicate rows in the dataset with `df.duplicated().any()` . visualized the correlations between the dataset's features.

5. Scaled the Features: performed feature scaling using Min-Max scaling on specific columns to ensure that all features had the same scale.
6. **Split the Data**: divided the dataset into two parts: the feature matrix 'X' (containing independent variables) and the target vector 'y' (containing the dependent variable, which is the diagnosis). split the data into training and testing sets using `train_test_split()` .
7. Trained and Evaluated the Model: initiated a Random Forest Classifier using `RandomForestClassifier()` . trained the Random Forest model on the training data by executing `rf_model.fit(X_train, y_train)` . Predictions were made on the testing data using `rf_model.predict(X_test)` . assessed the model's performance by calculating and visualizing the top 10 most important features. The model's performance was comprehensively evaluated using metrics such as accuracy, F1-score, recall, and precision. computed and displayed the confusion matrix to understand the model's predictive performance. A classification report was generated, offering additional performance metrics. ROC curve and ROC AUC score were calculated to assess the model's classification performance. Finally, generated a Precision-Recall curve, helping to understand the trade-off between precision and recall.

About the Dataset

This dataset contains a comprehensive set of features related to breast cancer characteristics, including size, shape, and texture. It is commonly used in machine learning and medical research for the classification of breast tumors as malignant or benign based on these features. It has 32 columns and 569 rows. Below are the details of each of the column headers

1. ID: This is a unique identifier for each data point or patient in the dataset.
2. Diagnosis: This column likely contains the outcome variable, indicating whether a patients breast cancer diagnosis is malignant (cancerous) or benign (non-cancerous).
3. Radius Mean: This feature likely represents the average radius of the detected tumors in millimeters.
4. Texture Mean: This feature may represent the average texture (variation in gray-scale intensity) of the tumors.
5. Perimeter Mean: This column likely contains the average perimeter (the length of the boundary) of the tumors.

6. Area Mean: This feature probably represents the average area (size) of the tumors in square millimeters.
7. Smoothness Mean: This feature might indicate the smoothness of the tumor boundary.
8. Compactness Mean: This column could represent a measure of compactness or the extent to which the tumor occupies space.
9. Concavity Mean: This feature may quantify the degree of concavity or irregularity in the tumor shape.
10. Concave Points Mean: This feature likely measures the number or characteristics of concave points in the tumor boundary.
11. Symmetry Mean: This column may represent a measure of symmetry in the tumor shape.
12. Fractal Dimension Mean: This feature could indicate the fractal dimension of the tumor boundary, which characterizes its complexity.
13. Radius SE: This may represent the standard error of the radius measurement.
14. Texture SE: This could be the standard error of the texture measurement.
15. Perimeter SE: This feature may represent the standard error of the perimeter measurement.
16. Area SE: This column likely contains the standard error of the area measurement.
17. Smoothness SE: This feature might represent the standard error of the smoothness measurement.
18. Compactness SE: This could be the standard error of the compactness measurement.
19. Concavity SE: This feature likely contains the standard error of the concavity measurement.
20. Concave Points SE: This may represent the standard error of the concave points measurement.
21. Symmetry SE: This feature could indicate the standard error of the symmetry measurement.
22. Fractal Dimension SE: This likely represents the standard error of the fractal dimension measurement.
23. Radius Worst: This may represent the worst (largest) radius measurement among the tumors.
24. Texture Worst: This column could represent the worst (largest) texture measurement among the tumors.
25. Perimeter Worst: This feature likely represents the worst (largest) perimeter measurement among the tumors.
26. Area Worst: This feature probably represents the worst (largest) area measurement among the tumors.
27. Smoothness Worst: This column may contain the worst (largest) smoothness measurement among the tumors.
28. Compactness Worst: This feature might represent the worst (largest) compactness measurement among the tumors.
29. Concavity Worst: This likely contains the worst (largest) concavity measurement among the tumors.
30. Concave Points Worst: This may represent the worst (largest) number or characteristics of concave points among the tumors.
31. Symmetry Worst: This feature could indicate the worst (largest) symmetry measurement among the tumors.
32. Fractal Dimension Worst: This likely represents the worst (largest) fractal dimension

In [10]:
```
#Importing the important libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#Importing metrices for evaluation
from sklearn.metrics import accuracy_score, precision_score, recall_score, conf
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import warnings
from sklearn.metrics import classification_report, roc_curve, roc_auc_score, pr
```

In [11]:
```
#filtering for future warnings
warnings.filterwarnings('ignore')
```

In [12]:
```
#Creating a dataframe

df = pd.read_csv('breast-cancer.csv') #Reading the dataset
df.head()
```

Out[12]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 32 columns

In [13]:
```
#Checking the shape of the dataset
df.shape
```

Out[13]:
```
(569, 32)
```

In [14]:
```
## Concise statistical description of numeric features
df.describe()
```

Out[14]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.09636
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.01406
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.05263
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.08637
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.09587
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.10530
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.16340

8 rows × 31 columns

In [15]:
```
# Remove the 'id' column as it is not needed for the analysis
df.drop(labels=['id'], axis=1, inplace=True)
transformer = MinMaxScaler()
```

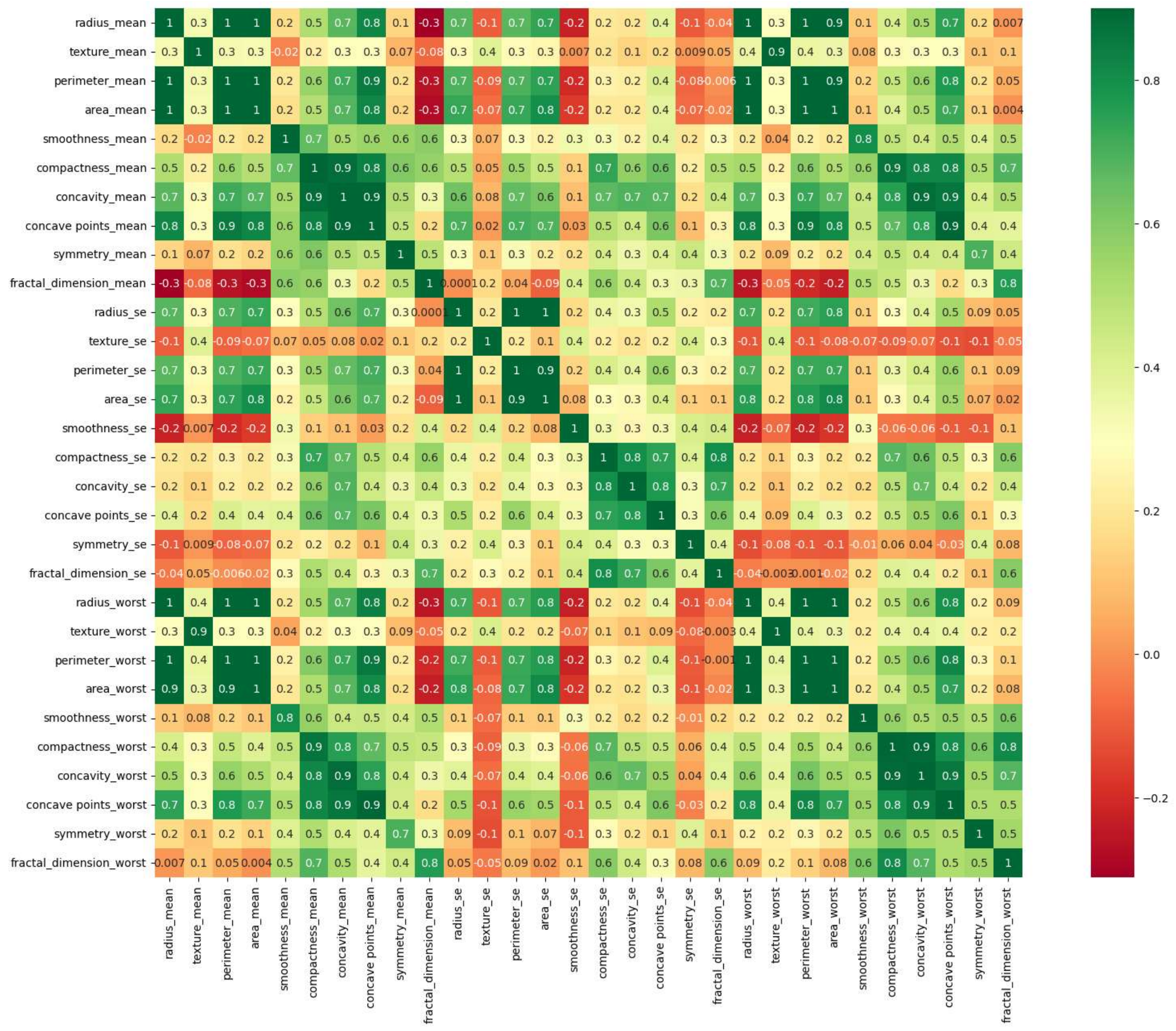
In [16]:
```
#Check for missing value
df.isnull().values.any()
```

Out[16]:
```
False
```

In [17]:
```
#Check for duplicate values
df.duplicated().any()
```

Out[17]:
```
False
```

```
In [18]: corrmat = df.corr() # Calculating the correlation matrix for the DataFrame 'df
         plt.subplots(figsize=(22,14))  #creating a subplot
         sns.heatmap(corrmat, annot=True, vmax=0.9, square=True,cmap='RdYlGn',fmt='.0g')
         plt.show();
```



In [19]:
```
def scaling(columns):
    return transformer.fit_transform(df[columns].values.reshape(-1, 1))
# List of columns to be scaled using MinMaxScaler.
columns_to_be_scaled = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_
                        'compactness_mean', 'concavity_mean', 'concave points_m
                        'fractal_dimension_mean', 'radius_se', 'texture_se', 'p
                        'smoothness_se', 'compactness_se', 'concavity_se', 'cor
                        'fractal_dimension_se', 'radius_worst', 'texture_worst'
                        'area_worst', 'smoothness_worst', 'compactness_worst',
                        'concave points_worst', 'symmetry_worst', 'fractal_dime
for i in columns_to_be_scaled:
    df[i] = scaling(i)

df.head() #Visualizing the header
```

Out[19]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compa
0	M	0.521037	0.022658	0.545989	0.363733	0.593753	
1	M	0.643144	0.272574	0.615783	0.501591	0.289880	
2	M	0.601496	0.390260	0.595743	0.449417	0.514309	
3	M	0.210090	0.360839	0.233501	0.102906	0.811321	
4	M	0.629893	0.156578	0.630986	0.489290	0.430351	

5 rows × 31 columns

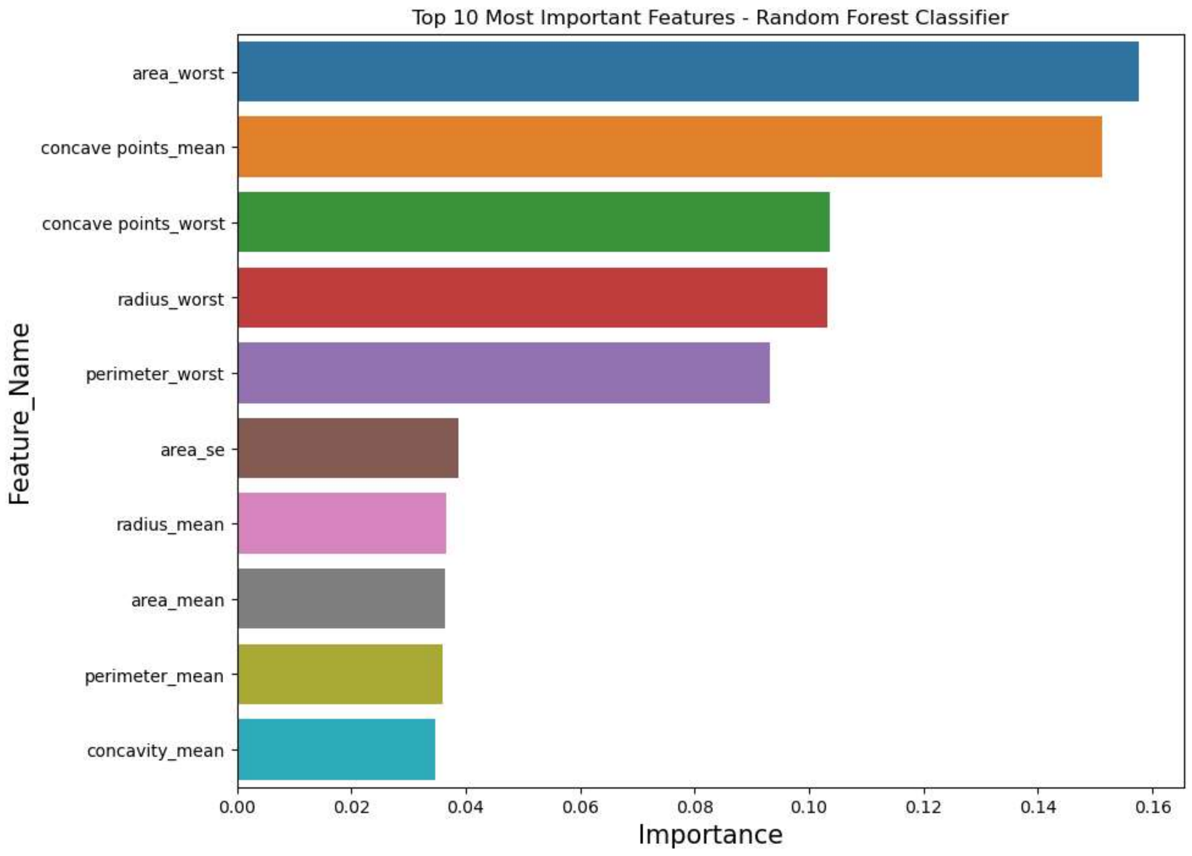
In [20]:
```
# Splitting the DataFrame 'df' into feature matrix 'X' and target vector 'y'.
X = df.drop(['diagnosis'], axis=1)
y = df['diagnosis']
# Split the dataset into training and testing sets.
# X_train: Training features
# X_test: Testing features
# y_train: Training target values
# y_test: Testing target values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, rando
```

In [21]:
```
#Getting the predictions

rf_model = RandomForestClassifier()  # Object creation
rf_model.fit(X_train, y_train)     # Fitting the data into the algorithm
y_pred_rf = rf_model.predict(X_test) # Getting the predictions
```

In [22]:
```
importance_df = pd.DataFrame({"Feature_Name": X.columns, "Importance": rf_model
# Sort the 'importance_df' DataFrame by the "Importance" column in descending o
# to identify the top 10 most important features.
sorted_importance_df = importance_df.sort_values(by="Importance", ascending=Fal
```

In [23]:
```
plt.figure(figsize=(10, 8))
sns.barplot(data=sorted_importance_df, x='Importance', y="Feature_Name")
plt.title("Top 10 Most Important Features - Random Forest Classifier")
plt.xlabel('Importance', fontsize=15)
plt.ylabel('Feature_Name', fontsize=15)
plt.show()
```



In [24]:
```
print("RANDOM FOREST MODEL RESULTS")
# Calculate and print the accuracy score for the testing set
print('Accuracy score for testing set: ', round(accuracy_score(y_test, y_pred_r
# Calculate and print the F1 score for the testing set
print('F1 score for testing set: ', round(f1_score(y_test, y_pred_rf, average='
# Calculate and print the recall score for the testing set
print('Recall score for testing set: ', round(recall_score(y_test, y_pred_rf, a
# Calculate and print the precision score for the testing set
print('Precision score for testing set: ', round(precision_score(y_test, y_pred

# Calculate the confusion matrix for the Random Forest model's predictions on t
cm_rf = confusion_matrix(y_test, y_pred_rf)
print(cm_rf)
```

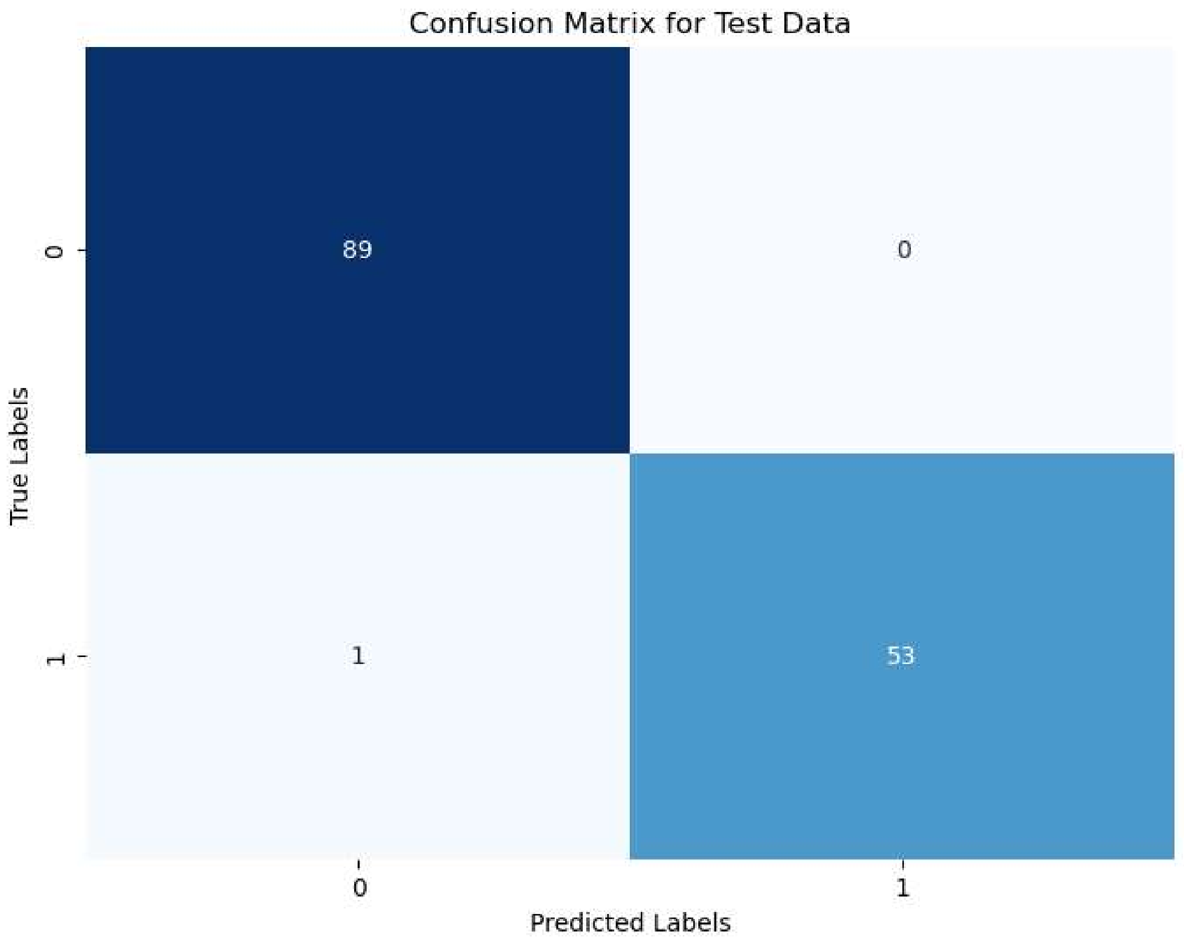
```
RANDOM FOREST MODEL RESULTS
Accuracy score for testing set:  0.993
F1 score for testing set:  0.993
Recall score for testing set:  0.993
Precision score for testing set:  0.993
[[89  0]
 [ 1 53]]
```

In [25]:
```
# Make predictions using the trained Random Forest model on the testing data
y_pred_rf_test = rf_model.predict(X_test)

# Calculate the confusion matrix for the Random Forest model's predictions on t
cm_rf_test = confusion_matrix(y_test, y_pred_rf_test)
```

In [26]:
```
# Visualizing the confusion matrix for the testing set

plt.figure(figsize=(8, 6))
sns.heatmap(cm_rf_test, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix for Test Data')
plt.show()
```



In [27]:
```

classification_rep = classification_report(y_test, y_pred_rf_test)
print("Classification Report:\n", classification_rep)

```

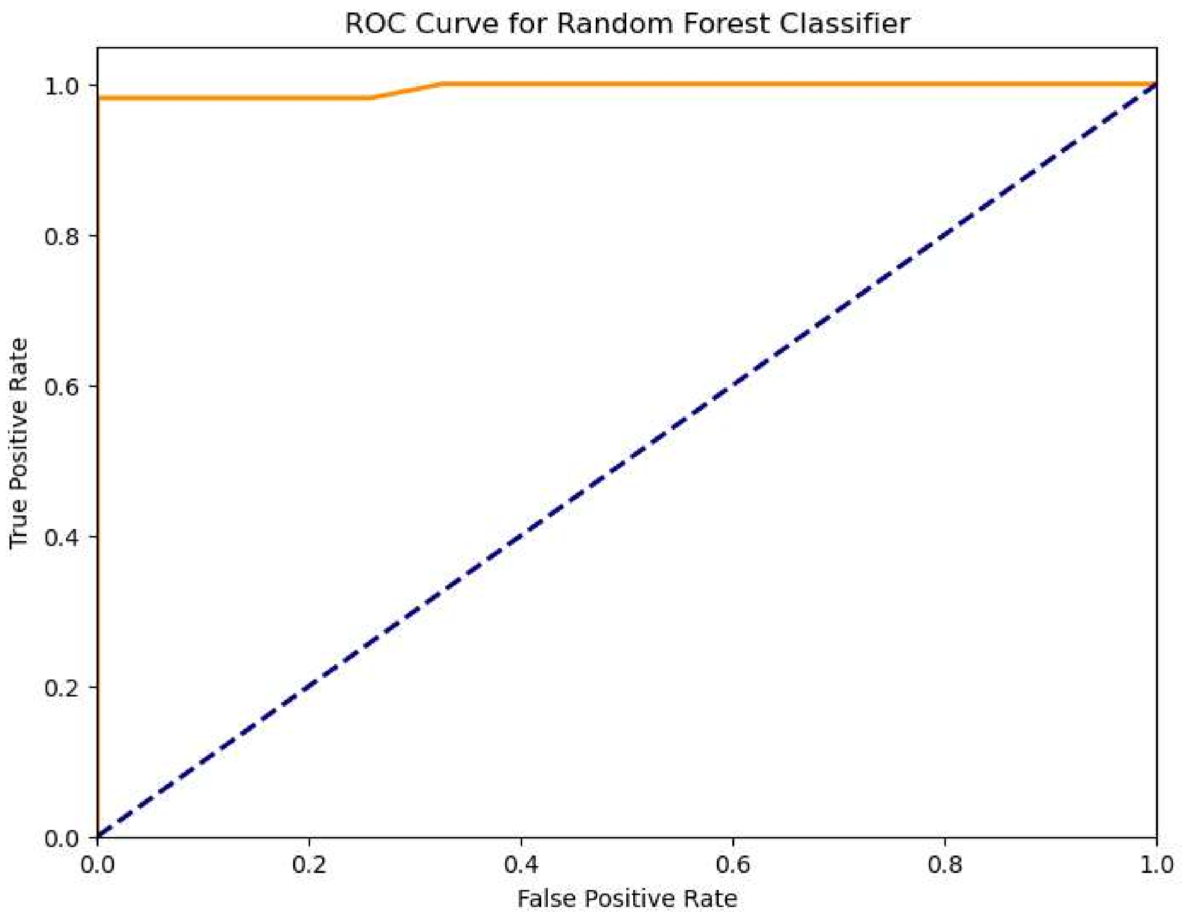
```
Classification Report:
               precision    recall  f1-score   support

           B       0.99      1.00      0.99        89
           M       1.00      0.98      0.99        54

    accuracy                           0.99       143
   macro avg       0.99      0.99      0.99       143
weighted avg       0.99      0.99      0.99       143
```

In [28]:
```
y_pred_prob_rf = rf_model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test.map({'B': 0, 'M': 1}), y_pred_prob_rf)
```

In [29]:
```
#visualizing the ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Random Forest Classifier')
plt.show()
```

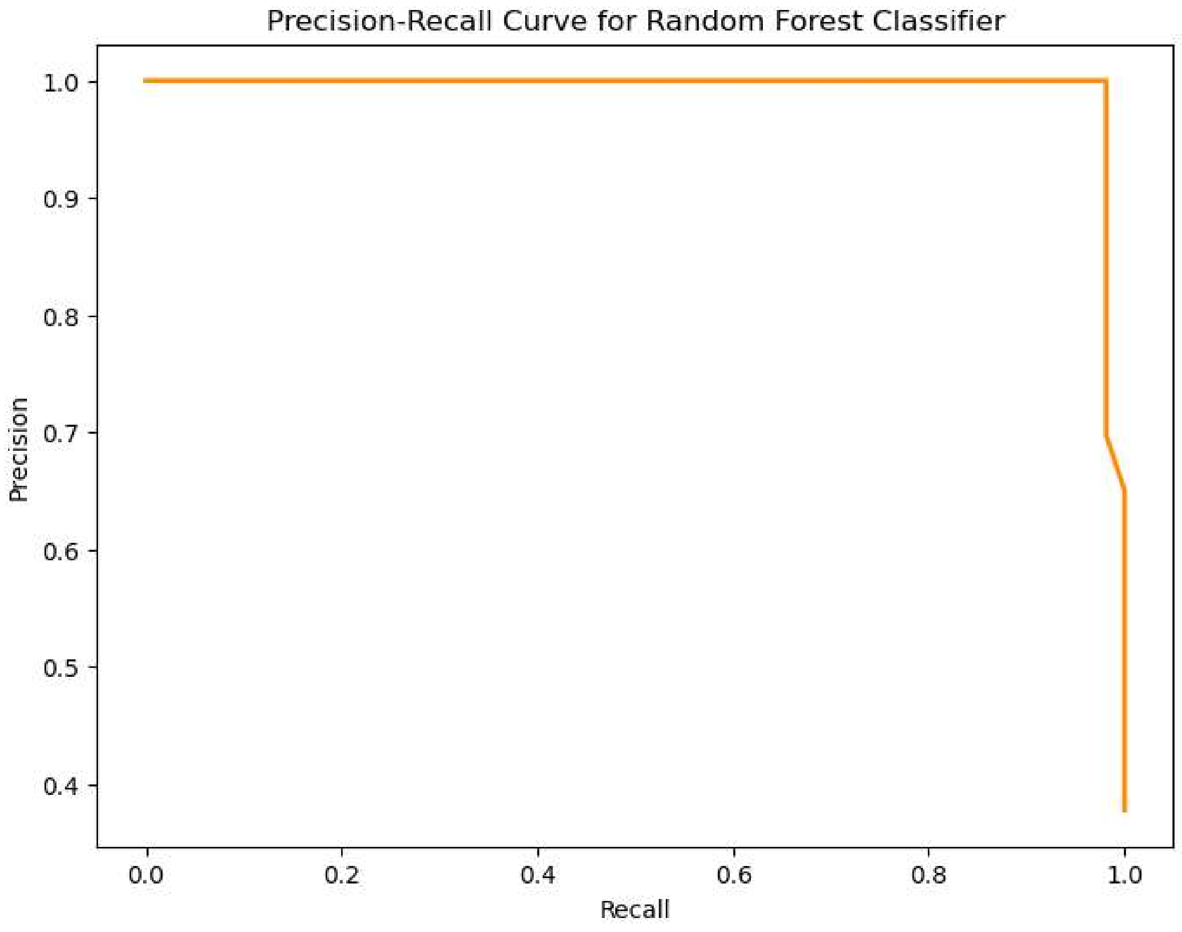


In [30]:
```
roc_auc = roc_auc_score(y_test.map({'B': 0, 'M': 1}), y_pred_prob_rf)
print("ROC AUC Score:", roc_auc)
```

```
ROC AUC Score: 0.9945900957136912
```

In [31]:
```
precision, recall, thresholds = precision_recall_curve(y_test.map({'B': 0, 'M'
```

In [32]:
```python
#visualizing the precision-recall curve for Random Forest Classifier
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='darkorange', lw=2)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve for Random Forest Classifier')
plt.show()
```



In [ ]:

In [ ]:

Dataset References

https://www.kaggle.com/datasets/nancyalaswad90/breast-cancer-dataset