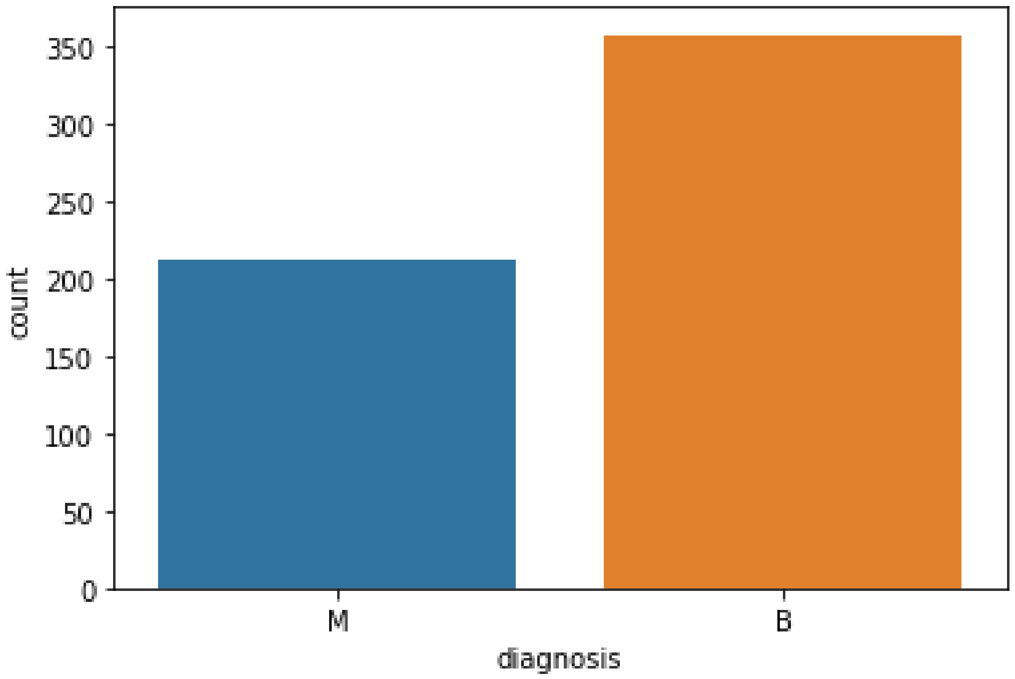
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv("data_cancer.csv")

# Display basic statistics of the dataset
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Visualize the target variable distribution
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='diagnosis', data=data)
plt.show()
```

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

       smoothness_mean  compactness_mean  concavity_mean  concave 
points_mean  \
count       569.000000        569.000000      569.000000          
569.000000
mean          0.096360          0.104341        0.088799          
0.048919
std           0.014064          0.052813        0.079720          
0.038803
```

```
min           0.052630          0.019380        0.000000
0.000000
25%           0.086370          0.064920        0.029560
0.020310
50%           0.095870          0.092630        0.061540
0.033500
75%           0.105300          0.130400        0.130700
0.074000
max           0.163400          0.345400        0.426800
0.201200

       symmetry_mean  ...  texture_worst  perimeter_worst   area_worst
\
count     569.000000  ...     569.000000       569.000000   569.000000

mean        0.181162  ...      25.677223       107.261213   880.583128

std         0.027414  ...       6.146258        33.602542   569.356993

min         0.106000  ...      12.020000        50.410000   185.200000

25%         0.161900  ...      21.080000        84.110000   515.300000

50%         0.179200  ...      25.410000        97.660000   686.500000

75%         0.195700  ...      29.720000       125.400000  1084.000000

max         0.304000  ...      49.540000       251.200000  4254.000000


       smoothness_worst  compactness_worst  concavity_worst  \
count        569.000000         569.000000       569.000000
mean           0.132369           0.254265         0.272188
std            0.022832           0.157336         0.208624
min            0.071170           0.027290         0.000000
25%            0.116600           0.147200         0.114500
50%            0.131300           0.211900         0.226700
75%            0.146000           0.339100         0.382900
max            0.222600           1.058000         1.252000

       concave points_worst  symmetry_worst
fractal_dimension_worst  \
count            569.000000      569.000000               569.000000

mean               0.114606        0.290076                 0.083946

std                0.065732        0.061867                 0.018061

min                0.000000        0.156500                 0.055040

25%                0.064930        0.250400                 0.071460
```

```
50%                  0.099930        0.282200                 0.080040

75%                  0.161400        0.317900                 0.092080

max                  0.291000        0.663800                 0.207500


       Unnamed: 32
count          0.0
mean           NaN
std            NaN
min            NaN
25%            NaN
50%            NaN
75%            NaN
max            NaN

[8 rows x 32 columns]
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
```

```
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```



```python
# Handle missing values (if any)
data.dropna(inplace=True)

# Encode categorical variable 'diagnosis' (Malignant (M) and Benign
(B))
data['diagnosis'] = data['diagnosis'].map({'M': 1, 'B': 0})

# Split the dataset into features (X) and target variable (y)
X = data.drop(['id', 'diagnosis'], axis=1)
y = data['diagnosis']



# Load the dataset
data = pd.read_csv("data_cancer.csv")

# Display basic statistics of the dataset
print(data.describe())

# Check the number of samples before handling missing values
print("Number of samples before handling missing values:", len(data))

# Handle missing values (if any)
data.dropna(inplace=True)
```

```
# Check the number of samples after handling missing values
print("Number of samples after handling missing values:", len(data))

# Check if the dataset is not empty before splitting
if len(data) > 0:
    # Encode categorical variable 'diagnosis' (Malignant (M) and
Benign (B))
    data['diagnosis'] = data['diagnosis'].map({'M': 1, 'B': 0})

    # Visualize the target variable distribution
    sns.countplot(x='diagnosis', data=data)
    plt.show()

    # Split the dataset into features (X) and target variable (y)
    X = data.drop(['id', '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)

    # Build and train a Random Forest model
    model = RandomForestClassifier(random_state=42)
    model.fit(X_train, y_train)

    # Evaluate the model on the testing set
    y_pred = model.predict(X_test)

    # Display performance metrics
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test,
y_pred))
else:
    print("Dataset is empty after handling missing values.")
```

```
                 id  radius_mean  texture_mean  perimeter_mean
area_mean  \
count  5.690000e+02   569.000000    569.000000      569.000000
569.000000
mean   3.037183e+07    14.127292     19.289649       91.969033
654.889104
std    1.250206e+08     3.524049      4.301036       24.298981
351.914129
min    8.670000e+03     6.981000      9.710000       43.790000
143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000
420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000
```

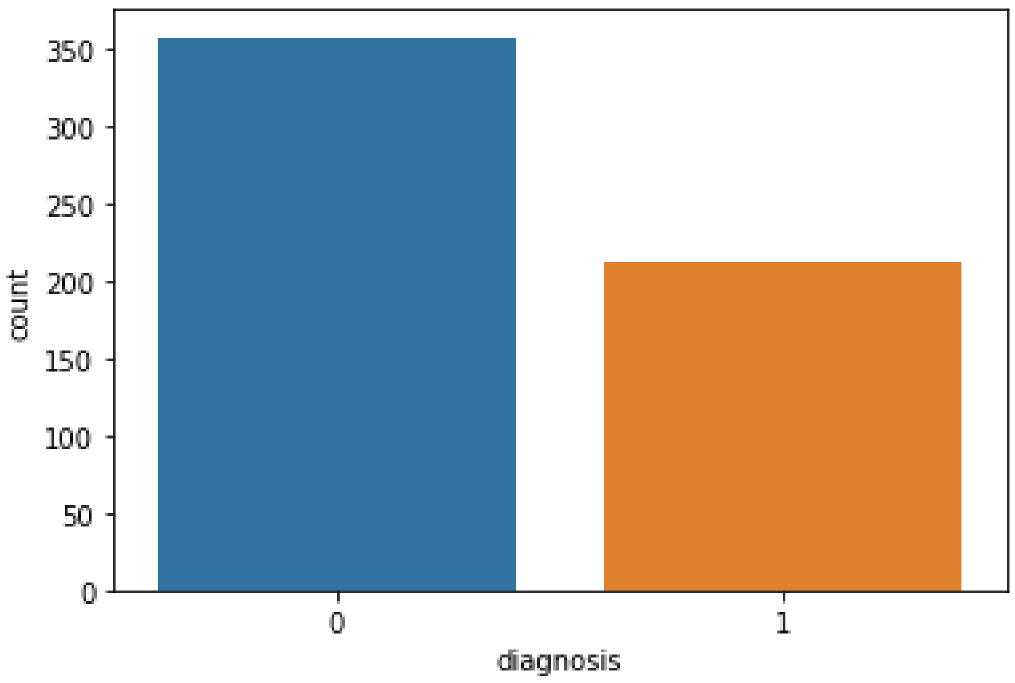
```
551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000
782.700000
max    9.113205e+08    28.110000     39.280000      188.500000
2501.000000

       smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
count       569.000000        569.000000      569.000000
569.000000
mean          0.096360          0.104341        0.088799
0.048919
std           0.014064          0.052813        0.079720
0.038803
min           0.052630          0.019380        0.000000
0.000000
25%           0.086370          0.064920        0.029560
0.020310
50%           0.095870          0.092630        0.061540
0.033500
75%           0.105300          0.130400        0.130700
0.074000
max           0.163400          0.345400        0.426800
0.201200

       symmetry_mean  ...  texture_worst  perimeter_worst   area_worst
\
count     569.000000  ...     569.000000       569.000000   569.000000

mean        0.181162  ...      25.677223       107.261213   880.583128

std         0.027414  ...       6.146258        33.602542   569.356993

min         0.106000  ...      12.020000        50.410000   185.200000

25%         0.161900  ...      21.080000        84.110000   515.300000

50%         0.179200  ...      25.410000        97.660000   686.500000

75%         0.195700  ...      29.720000       125.400000  1084.000000

max         0.304000  ...      49.540000       251.200000  4254.000000


       smoothness_worst  compactness_worst  concavity_worst  \
count        569.000000         569.000000       569.000000
mean           0.132369           0.254265         0.272188
std            0.022832           0.157336         0.208624
min            0.071170           0.027290         0.000000
25%            0.116600           0.147200         0.114500
```

```
50%              0.131300            0.211900          0.226700
75%              0.146000            0.339100          0.382900
max              0.222600            1.058000          1.252000

       concave points_worst  symmetry_worst
fractal_dimension_worst  \
count            569.000000      569.000000                569.000000

mean               0.114606        0.290076                  0.083946

std                0.065732        0.061867                  0.018061

min                0.000000        0.156500                  0.055040

25%                0.064930        0.250400                  0.071460

50%                0.099930        0.282200                  0.080040

75%                0.161400        0.317900                  0.092080

max                0.291000        0.663800                  0.207500


       Unnamed: 32
count          0.0
mean           NaN
std            NaN
min            NaN
25%            NaN
50%            NaN
75%            NaN
max            NaN

[8 rows x 32 columns]
Number of samples before handling missing values: 569
Number of samples after handling missing values: 0
Dataset is empty after handling missing values.
```

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('data_cancer.csv')

# Drop unnecessary columns (e.g., 'id' and 'Unnamed: 32')
data.drop(['id', 'Unnamed: 32'], axis=1, inplace=True)
```

```
# Handle missing values using mean imputation
data.fillna(data.mean(), inplace=True)

# Map the target variable to numerical values (Malignant: 1, Benign:
0)
data['diagnosis'] = data['diagnosis'].map({'M': 1, 'B': 0})

# Visualize the target variable distribution
sns.countplot(x='diagnosis', data=data)
plt.show()

# Split the dataset into 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)

# Build and train a Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:\n', classification_report)
```

```
C:\Users\MAYURSINGH\AppData\Local\Temp\
ipykernel_11636\1998506942.py:15: FutureWarning: Dropping of nuisance
columns in DataFrame reductions (with 'numeric_only=None') is
deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  data.fillna(data.mean(), inplace=True)
```



```
Accuracy: 0.9649
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.97        71
           1       0.98      0.93      0.95        43

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```