Start coding or generate with AI.

```
# Import necessary libraries
import pandas as pd  # For handling data
import numpy as np  # For numerical operations
import matplotlib.pyplot as plt  # For visualization
import seaborn as sns  # For better plots
from sklearn.datasets import load_breast_cancer  # To get the dataset



# Load the breast cancer dataset
data = load_breast_cancer()

# Convert it into a DataFrame (table format)
df = pd.DataFrame(data.data, columns=data.feature_names)

# Add the target column (1 = benign, 0 = malignant)
df['target'] = data.target

# Show the first 5 rows
print(df.head())
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst area
0                 0.07871  ...          17.33           184.60      2019.0
1                 0.05667  ...          23.41           158.80      1956.0
2                 0.05999  ...          25.53           152.50      1709.0
3                 0.09744  ...          26.50            98.87       567.7
4                 0.05883  ...          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave poin
0            0.1622             0.6656           0.7119                0.26
1            0.1238             0.1866           0.2416                0.18
2            0.1444             0.4245           0.4504                0.24
3            0.2098             0.8663           0.6869                0.25
4            0.1374             0.2050           0.4000                0.16

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
```

```
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

```
#Explore the dataset
print(f"Dataset shape: {df.shape}")
```

```
Dataset shape: (569, 31)
```

```
print(df.isnull().sum())
```

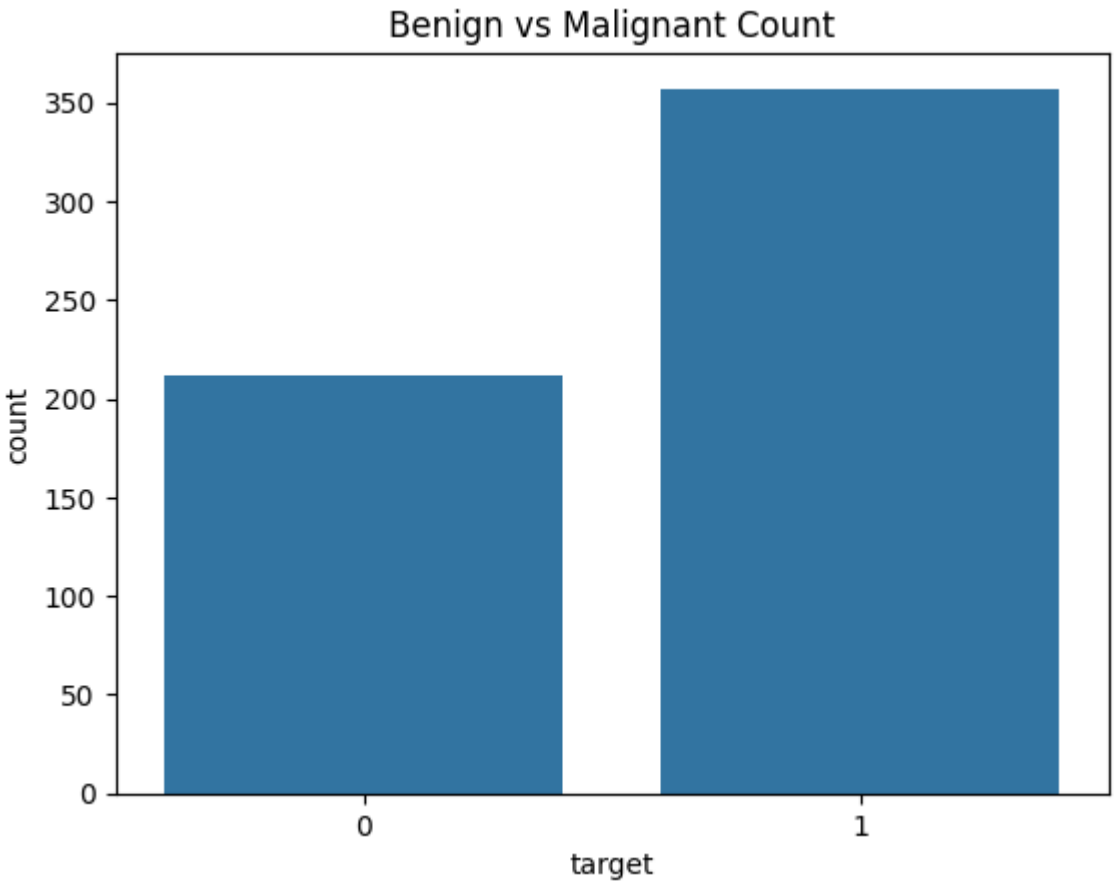
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

```
print(df.describe())
```

```
       mean radius  mean texture  mean perimeter    mean area  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
```

```
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      28.110000     39.280000      188.500000  2501.000000

       mean smoothness  mean compactness  mean concavity  mean concave poin
count       569.000000        569.000000      569.000000           569.0000
mean          0.096360          0.104341        0.088799             0.0489
std           0.014064          0.052813        0.079720             0.0388
min           0.052630          0.019380        0.000000             0.0000
25%           0.086370          0.064920        0.029560             0.0203
50%           0.095870          0.092630        0.061540             0.0335
75%           0.105300          0.130400        0.130700             0.0740
max           0.163400          0.345400        0.426800             0.2012

       mean symmetry  mean fractal dimension  ...  worst texture  \
count     569.000000              569.000000  ...     569.000000
mean        0.181162                0.062798  ...      25.677223
std         0.027414                0.007060  ...       6.146258
min         0.106000                0.049960  ...      12.020000
25%         0.161900                0.057700  ...      21.080000
50%         0.179200                0.061540  ...      25.410000
75%         0.195700                0.066120  ...      29.720000
max         0.304000                0.097440  ...      49.540000

       worst perimeter   worst area  worst smoothness  worst compactness  \
count       569.000000   569.000000        569.000000         569.000000
mean        107.261213   880.583128          0.132369           0.254265
std          33.602542   569.356993          0.022832           0.157336
min          50.410000   185.200000          0.071170           0.027290
25%          84.110000   515.300000          0.116600           0.147200
50%          97.660000   686.500000          0.131300           0.211900
75%         125.400000  1084.000000          0.146000           0.339100
max         251.200000  4254.000000          0.222600           1.058000

       worst concavity  worst concave points  worst symmetry  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
50%           0.226700              0.099930        0.282200
75%           0.382900              0.161400        0.317900
max           1.252000              0.291000        0.663800

       worst fractal dimension      target
count               569.000000  569.000000
mean                  0.083946    0.627417
std                   0.018061    0.483918
min                   0.055040    0.000000
25%                   0.071460    0.000000
50%                   0.080040    1.000000
75%                   0.092080    1.000000
```

```
sns.countplot(x=df["target"])
plt.title("Benign vs Malignant Count")
plt.show()
```




```
#Data Preprocessing
# X = all features except the target column
X = df.drop(columns=['target'])

# y = target column (1 = benign, 0 = malignant)
y = df['target']

print(f"X shape: {X.shape}")
print(f"y shape: {y.shape}")
```

```
X shape: (569, 30)
y shape: (569,)
```

```
from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform X (features)
X_scaled = scaler.fit_transform(X)

# Convert back to DataFrame for better readability
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)

print(X_scaled.head())  # Display first 5 rows after scaling
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0     1.097064     -2.073335        1.269934   0.984375         1.568466
1     1.829821     -0.353632        1.685955   1.908708        -0.826962
2     1.579888      0.456187        1.566503   1.558884         0.942210
3    -0.768909      0.253732       -0.592687  -0.764464         3.283553
4     1.750297     -1.151816        1.776573   1.826229         0.280372

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0          3.283515        2.652874             2.532475       2.217515
1         -0.487072       -0.023846             0.548144       0.001392
2          1.052926        1.363478             2.037231       0.939685
3          3.402909        1.915897             1.451707       2.867383
4          0.539340        1.371011             1.428493      -0.009560

   mean fractal dimension  ...  worst radius  worst texture  worst perimete
0                2.255747  ...      1.886690      -1.359293         2.30360
1               -0.868652  ...      1.805927      -0.369203         1.53512
2               -0.398008  ...      1.511870      -0.023974         1.34747
3                4.910919  ...     -0.281464       0.133984        -0.24993
4               -0.562450  ...      1.298575      -1.466770         1.33853

   worst area  worst smoothness  worst compactness  worst concavity  \
0    2.001237          1.307686           2.616665         2.109526
1    1.890489         -0.375612          -0.430444        -0.146749
2    1.456285          0.527407           1.082932         0.854974
3   -0.550021          3.394275           3.893397         1.989588
4    1.220724          0.220556          -0.313395         0.613179

   worst concave points  worst symmetry  worst fractal dimension
0              2.296076        2.750622                 1.937015
1              1.087084       -0.243890                 0.281190
2              1.955000        1.152255                 0.201391
3              2.175786        6.046041                 4.935010
4              0.729259       -0.868353                -0.397100

[5 rows x 30 columns]
```

```
from sklearn.model_selection import train_test_split

# Split into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,

print(f"Training set size: {X_train.shape}")
print(f"Testing set size: {X_test.shape}")
```

```
Training set size: (455, 30)
Testing set size: (114, 30)
```

```
#Train the Machine Learning Model

from sklearn.linear_model import LogisticRegression

# Initialize the model
model = LogisticRegression()
```

```
# Train (fit) the model on the training data
model.fit(X_train, y_train)

print("Model training complete!")
```

```
Model training complete!
```

```
# Predict on the test set
y_pred = model.predict(X_test)

# Show first 10 predictions
print("Predictions:", y_pred[:10])
print("Actual Values:", y_test[:10].values)
```

```
Predictions: [1 0 0 1 1 0 0 0 1 1]
Actual Values: [1 0 0 1 1 0 0 0 1 1]
```

```
#Evaluate the model
from sklearn.metrics import accuracy_score, classification_report

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

# Show detailed performance metrics
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Model Accuracy: 0.97

Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.95      0.96        43
           1       0.97      0.99      0.98        71

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

```
# Train the Random Forest Model
from sklearn.ensemble import RandomForestClassifier

# Initialize the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

print("Random Forest model training complete!")
```

```
print("Random Forest model training complete!")
```

```
Random Forest model training complete!
```

```
# Predict on the test set
rf_y_pred = rf_model.predict(X_test)

# Show first 10 predictions
print("Random Forest Predictions:", rf_y_pred[:10])
print("Actual Values:", y_test[:10].values)
```

```
Random Forest Predictions: [1 0 0 1 1 0 0 0 0 1]
Actual Values: [1 0 0 1 1 0 0 0 1 1]
```

```
# Calculate accuracy
rf_accuracy = accuracy_score(y_test, rf_y_pred)
print(f"Random Forest Model Accuracy: {rf_accuracy:.2f}")

# Show detailed performance metrics
print("\nRandom Forest Classification Report:\n", classification_report(y_test,
```

```
Random Forest Model Accuracy: 0.96

Random Forest Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

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

```
!pip install shap
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/di
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/d
Requirement already satisfied: numba in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dis
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

```
# Convert X_test to a NumPy array
X_test_array = X_test.values

# Plot the SHAP summary plot again
shap.summary_plot(rf_shap_values, X_test_array)
```

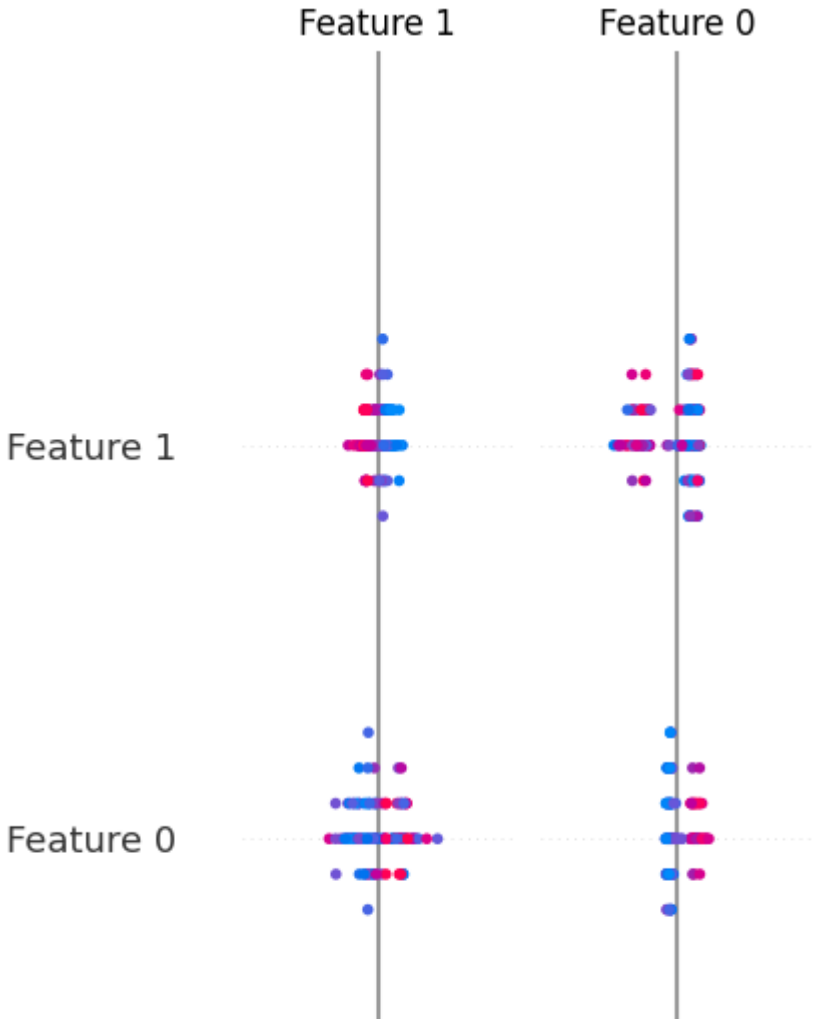
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
<ipython-input-19-29908f7d7862> in <cell line: 0>()
     11
     12 # Plot the SHAP summary plot again
---> 13 shap.summary_plot(rf_shap_values, X_test_array)

NameError: name 'rf_shap_values' is not defined
```

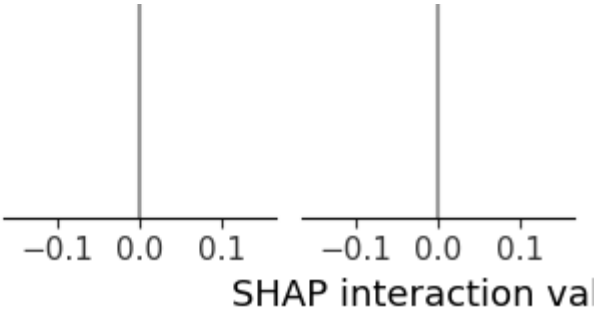
Next steps: Explain error

```
# Import necessary libraries
import shap
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Load the dataset (make sure you've loaded the dataset into 'df' previously)
# df = pd.read_csv('your_dataset.csv')  # if you're reading from a file, uncomm

# Separate features (X) and target (y)
X = df.drop(columns=['target'])
y = df['target']

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,

# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict on the test set
rf_y_pred = rf_model.predict(X_test)

# Evaluate the model
rf_accuracy = accuracy_score(y_test, rf_y_pred)
print(f"Random Forest Model Accuracy: {rf_accuracy:.2f}")
```

```
print("\nRandom Forest Classification Report:\n", classification_report(y_test,

# SHAP explanation
# Convert X_test to NumPy array for SHAP
X_test_array = X_test

# Create SHAP explainer
rf_explainer = shap.Explainer(rf_model, X_train)

# Calculate SHAP values for the test set
rf_shap_values = rf_explainer(X_test_array)

# Plot the SHAP summary plot
shap.summary_plot(rf_shap_values, X_test_array)
```

```
Random Forest Model Accuracy: 0.96

Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

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






```
import joblib

# Save the model
joblib.dump(rf_model, 'random_forest_model.pkl')

# Save the scaler
joblib.dump(scaler, 'scaler.pkl')

print("Model and scaler saved successfully!")
```

```
Model and scaler saved successfully!
```

```
!pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.43.2-py2.py3-none-any.whl.metadata (8.9 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/local/lib/python3.
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.1
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/li
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 2.3 MB/s eta 0:0
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/l
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.1
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
```

```
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Downloading streamlit-1.43.2-py2.py3-none-any.whl (9.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.7/9.7 MB 69.1 MB/s eta 0:00:0
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 81.6 MB/s eta 0:00:0
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 5.6 MB/s eta 0:00:
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.43.2 watchdog-6.0.0
```

```
import streamlit as st
import numpy as np
import joblib
from sklearn.preprocessing import StandardScaler

# Load your model and scaler
rf_model = joblib.load("random_forest_model.pkl")
scaler = joblib.load("scaler.pkl")

# Function to get user input
def get_user_input():
    # User inputs for all 30 features
    mean_radius = st.number_input("Mean Radius", min_value=0.0, max_value=50.0,
    mean_texture = st.number_input("Mean Texture", min_value=0.0, max_value=50.
    mean_perimeter = st.number_input("Mean Perimeter", min_value=0.0, max_value
    mean_area = st.number_input("Mean Area", min_value=0.0, max_value=5000.0, v
    mean_smoothness = st.number_input("Mean Smoothness", min_value=0.0, max_val
    mean_compactness = st.number_input("Mean Compactness", min_value=0.0, max_v
    mean_concavity = st.number_input("Mean Concavity", min_value=0.0, max_value
    mean_concave_points = st.number_input("Mean Concave Points", min_value=0.0,
    mean_symmetry = st.number_input("Mean Symmetry", min_value=0.0, max_value=0
    mean_fractal_dimension = st.number_input("Mean Fractal Dimension", min_valu

    radius_error = st.number_input("Radius Error", min_value=0.0, max_value=50.
    texture_error = st.number_input("Texture Error", min_value=0.0, max_value=5
    perimeter_error = st.number_input("Perimeter Error", min_value=0.0, max_val
    area_error = st.number_input("Area Error", min_value=0.0, max_value=2000.0,
    smoothness_error = st.number_input("Smoothness Error", min_value=0.0, max_v
    compactness_error = st.number_input("Compactness Error", min_value=0.0, max
    concavity_error = st.number_input("Concavity Error", min_value=0.0, max_val
    concave_points_error = st.number_input("Concave Points Error", min_value=0.
    symmetry_error = st.number_input("Symmetry Error", min_value=0.0, max_value
    fractal_dimension_error = st.number_input("Fractal Dimension Error", min_va

    worst_radius = st.number_input("Worst Radius", min_value=0.0, max_value=100
```

```
        worst_texture = st.number_input("Worst Texture", min_value=0.0, max_value=5
        worst_perimeter = st.number_input("Worst Perimeter", min_value=0.0, max_val
        worst_area = st.number_input("Worst Area", min_value=0.0, max_value=10000.0
        worst_smoothness = st.number_input("Worst Smoothness", min_value=0.0, max_v
        worst_compactness = st.number_input("Worst Compactness", min_value=0.0, max
        worst_concavity = st.number_input("Worst Concavity", min_value=0.0, max_val
        worst_concave_points = st.number_input("Worst Concave Points", min_value=0.
        worst_symmetry = st.number_input("Worst Symmetry", min_value=0.0, max_value
        worst_fractal_dimension = st.number_input("Worst Fractal Dimension", min_va

        # Create the user input array
        user_input = np.array([[mean_radius, mean_texture, mean_perimeter, mean_are
                                mean_compactness, mean_concavity, mean_concave_poir
                                mean_fractal_dimension, radius_error, texture_error
                                smoothness_error, compactness_error, concavity_erro
                                symmetry_error, fractal_dimension_error, worst_radi
                                worst_area, worst_smoothness, worst_compactness, wo
                                worst_symmetry, worst_fractal_dimension]])

        return user_input

    # Main Streamlit app function
    def main():
        st.title("Breast Cancer Prediction")

        # Get user input
        user_input = get_user_input()

        # Add a **Predict** button
        if st.button("Predict"):
            # Scale the user input using the scaler fitted on the training data
            user_input_scaled = scaler.transform(user_input)

            # Make the prediction using the trained model
            prediction = rf_model.predict(user_input_scaled)

            # Display the result
            if prediction == 0:
                st.subheader("🔴 The model predicts the tumor is **Malignant**")
            else:
                st.subheader("🟢 The model predicts the tumor is **Benign**")

    # Run the app
    if __name__ == '__main__':
        main()
```

```
2025-03-11 20:20:29.019 WARNING streamlit.runtime.scriptrunner_utils.script
2025-03-11 20:20:29.260
  Warning: to view this Streamlit app on a browser, run it with the followi
  command:

    streamlit run /usr/local/lib/python3.11/dist-packages/colab_kernel_laun
2025-03-11 20:20:29.264 Thread 'MainThread': missing ScriptRunContext! This
```

```
2025-03-11 20:20:29.267 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.269 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.272 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.274 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.277 Session state does not function when running a scri
2025-03-11 20:20:29.279 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.282 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.284 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.286 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.287 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.290 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.292 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.294 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.296 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.298 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.300 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.302 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.303 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.305 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.307 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.309 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.311 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.314 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.316 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.317 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.319 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.321 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.323 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.325 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.326 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.330 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.331 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.331 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.334 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.338 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.339 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.340 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.340 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.344 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.344 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.345 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.346 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.351 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.352 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.352 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.356 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.357 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.357 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.358 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.361 Thread 'MainThread': missing ScriptRunContext! This
2025-03-11 20:20:29.362 Thread 'MainThread': missing ScriptRunContext! This
```

```
!pip install streamlit
!pip install pyngrok
```

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.11/dist-
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/local/lib/python3.
```

```
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.1
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/li
Requirement already satisfied: watchdog<7,>=2.1.5 in /usr/local/lib/python3
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/l
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.1
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Collecting pyngrok
  Downloading pyngrok-7.2.3-py3-none-any.whl.metadata (8.7 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.11/dis
Downloading pyngrok-7.2.3-py3-none-any.whl (23 kB)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.2.3
```

```
code = """
import streamlit as st
import numpy as np
import joblib
from sklearn.preprocessing import StandardScaler

# Load your model and scaler
rf_model = joblib.load("random_forest_model.pkl")
scaler = joblib.load("scaler.pkl")

# Function to get user input
def get_user_input():
    # User inputs for all 30 features
    mean_radius = st.number_input("Mean Radius", min_value=0.0, max_value=50.0,
    mean_texture = st.number_input("Mean Texture", min_value=0.0, max_value=50.
    mean_perimeter = st.number_input("Mean Perimeter", min_value=0.0, max_value
```

```
    mean_perimeter = st.number_input("Mean Perimeter", min_value=0.0, max_value
    mean_area = st.number_input("Mean Area", min_value=0.0, max_value=5000.0, v
    mean_smoothness = st.number_input("Mean Smoothness", min_value=0.0, max_val
    mean_compactness = st.number_input("Mean Compactness", min_value=0.0, max_v
    mean_concavity = st.number_input("Mean Concavity", min_value=0.0, max_value
    mean_concave_points = st.number_input("Mean Concave Points", min_value=0.0,
    mean_symmetry = st.number_input("Mean Symmetry", min_value=0.0, max_value=0
    mean_fractal_dimension = st.number_input("Mean Fractal Dimension", min_valu

    radius_error = st.number_input("Radius Error", min_value=0.0, max_value=50.
    texture_error = st.number_input("Texture Error", min_value=0.0, max_value=5
    perimeter_error = st.number_input("Perimeter Error", min_value=0.0, max_val
    area_error = st.number_input("Area Error", min_value=0.0, max_value=2000.0,
    smoothness_error = st.number_input("Smoothness Error", min_value=0.0, max_v
    compactness_error = st.number_input("Compactness Error", min_value=0.0, max
    concavity_error = st.number_input("Concavity Error", min_value=0.0, max_val
    concave_points_error = st.number_input("Concave Points Error", min_value=0.
    symmetry_error = st.number_input("Symmetry Error", min_value=0.0, max_value
    fractal_dimension_error = st.number_input("Fractal Dimension Error", min_va

    worst_radius = st.number_input("Worst Radius", min_value=0.0, max_value=100
    worst_texture = st.number_input("Worst Texture", min_value=0.0, max_value=5
    worst_perimeter = st.number_input("Worst Perimeter", min_value=0.0, max_val
    worst_area = st.number_input("Worst Area", min_value=0.0, max_value=10000.0
    worst_smoothness = st.number_input("Worst Smoothness", min_value=0.0, max_v
    worst_compactness = st.number_input("Worst Compactness", min_value=0.0, max
    worst_concavity = st.number_input("Worst Concavity", min_value=0.0, max_val
    worst_concave_points = st.number_input("Worst Concave Points", min_value=0.
    worst_symmetry = st.number_input("Worst Symmetry", min_value=0.0, max_value
    worst_fractal_dimension = st.number_input("Worst Fractal Dimension", min_va

    # Create the user input array
    user_input = np.array([[mean_radius, mean_texture, mean_perimeter, mean_are
                            mean_compactness, mean_concavity, mean_concave_poir
                            mean_fractal_dimension, radius_error, texture_error
                            smoothness_error, compactness_error, concavity_erro
                            symmetry_error, fractal_dimension_error, worst_radi
                            worst_area, worst_smoothness, worst_compactness, wo
                            worst_symmetry, worst_fractal_dimension]])

    return user_input

# Main Streamlit app function
def main():
    st.title("Breast Cancer Prediction")

    # Get user input
    user_input = get_user_input()

    # Scale the user input using the scaler fitted on the training data
    user_input_scaled = scaler.transform(user_input)

    # Make the prediction using the trained model
    prediction = rf_model.predict(user_input_scaled)
```

```
    # Display the result
    if prediction == 0:
        st.write("The model predicts the tumor is **Malignant**")
    else:
        st.write("The model predicts the tumor is **Benign**")

# Run the app
if __name__ == '__main__':
    main()
"""

with open("/content/breast_cancer_app.py", "w") as f:
    f.write(code)
```

```
!pip install pyngrok
```

```
Requirement already satisfied: pyngrok in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.11/dis
```

```
!ngrok authtoken 2tVI0bSRvJLtBQ2T3GE7keZDe5t_52p88WuXttaxEMC44afT1
```

```
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
```

```
!pip install streamlit pyngrok --quiet
```

```
!streamlit run breast_cancer_app.py --server.enableCORS false --server.enableXsr
```

```
...
    Collecting usage statistics. To deactivate, set browser.gatherUsageStats to

      You can now view your Streamlit app in your browser.

      Local URL: http://localhost:8501
      Network URL: http://172.28.0.12:8501
      External URL: http://34.125.234.42:8501
```

```
from pyngrok import ngrok

# Kill any old tunnels
ngrok.kill()

# Create a new tunnel for Streamlit (port 8501)
public_url = ngrok.connect(port='8501')
```

```
print(f"Public URL: {public_url}")
```

```
WARNING:pyngrok.process.ngrok:t=2025-02-24T21:25:54+0000 lvl=warn msg="inva
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call
last)
/usr/local/lib/python3.11/dist-packages/pyngrok/ngrok.py in
api_request(url, method, data, params, timeout, auth)
    556     try:
--> 557         response = urlopen(request, encoded_data, timeout)
    558         response_data = response.read().decode("utf-8")

                                 8 frames
HTTPError: HTTP Error 400: Bad Request

During handling of the above exception, another exception occurred:

PyngrokNgrokHTTPError                     Traceback (most recent call
last)
/usr/local/lib/python3.11/dist-packages/pyngrok/ngrok.py in
api_request(url, method, data, params, timeout, auth)
    576         logger.debug(f"Response {status_code}:
{response_data.strip()}")
    577
--> 578         raise PyngrokNgrokHTTPError(f"ngrok client exception, API
returned {status_code}: {response_data}",
    579                                     e.url,
    580                                     status_code, e.reason,
```

```
!git config --global user.name davisheela506
!git config --global user.email sheeladavi506@gmail.com
```

```
!git init
!git remote add origin https://github.com/davisheela506/breastcancerproject
```

```
hint: Using 'master' as the name for the initial branch. This default branc
hint: is subject to change. To configure the initial branch name to use in
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this comman
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /content/.git/
```

```
!mv app.py your-repo/
!mv random_forest_model.pkl your-repo/
!mv scaler.pkl your-repo/
```

```
mv: cannot stat 'app.py': No such file or directory
```

```
mv: cannot move 'random_forest_model.pkl' to 'your-repo/': Not a directory
mv: cannot move 'scaler.pkl' to 'your-repo/': Not a directory
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call
last)
<ipython-input-81-d5df0069828e> in <cell line: 0>()
      1 from google.colab import drive
----> 2 drive.mount('/content/drive')

 1 frames
/usr/local/lib/python3.11/dist-packages/google/colab/drive.py in
_mount(mountpoint, force_remount, timeout_ms, ephemeral, readonly)
    275             'https://research.google.com/colaboratory/
faq.html#drive-timeout'
    276         )
--> 277       raise ValueError('mount failed' + extra_reason)
    278     elif case == 4:
    279       # Terminate the DriveFS binary before killing bash.
```

Start coding or generate with AI.