In [1]:

```
import pickle
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sn
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [2]:

```
names = 'C:/Users/hsahn/Design of AI systems/module5/wdbc.names'
table = 'C:/Users/hsahn/Design of AI systems/module5/wdbc.pkl'
```

In [3]:

```
with open(table, 'rb') as f:
    df_table_dataset = pickle.load(f)
```

In [4]:

```
df_table_dataset
```

Out[4]:

	id	malignant	radius_0	texture_0	perimeter_0	area_0	smoothness_0	compactness_0	concavity_0	concave points_0	...	radius_2	texture_2	perimeter_2	area_2	smoothness_2	compactness_2	concavity_2	concave points_2	symmetry_2	fractal dimension_2
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	926682	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	926954	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	927241	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	92751	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

569 rows × 32 columns

In [5]:

```
df_table_dataset[df_table_dataset['malignant'] == 1]
```

Out[5]:

	id	malignant	radius_0	texture_0	perimeter_0	area_0	smoothness_0	compactness_0	concavity_0	concave points_0	...	radius_2	texture_2	perimeter_2	area_2	smoothness_2	compactness_2	concavity_2	concave points_2	symmetry_2	fractal dimension_2
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
563	926125	1	20.92	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0.14740	...	24.29	29.41	179.10	1819.0	0.1407	0.4186	0.6599	0.2542	0.2929	0.09873
564	926424	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.45	26.40	166.10	2027.0	0.1410	0.2113	0.4107	0.2216	0.2060	0.07115
565	926682	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.69	38.25	155.00	1731.0	0.1166	0.1922	0.3215	0.1628	0.2572	0.06637
566	926954	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.98	34.12	126.70	1124.0	0.1139	0.3094	0.3403	0.1418	0.2218	0.07820
567	927241	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.74	39.42	184.60	1821.0	0.1650	0.8681	0.9387	0.2650	0.4087	0.12400

212 rows × 32 columns

In [6]:

```
df_table_dataset[df_table_dataset['malignant'] == 0]
```

Out[6]:

	id	malignant	radius_0	texture_0	perimeter_0	area_0	smoothness_0	compactness_0	concavity_0	concave points_0	...	radius_2	texture_2	perimeter_2	area_2	smoothness_2	compactness_2	concavity_2	concave points_2	symmetry_2	fractal dimension_2
19	8510426	0	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.047810	...	15.110	19.26	99.70	711.2	0.14400	0.17730	0.23900	0.12880	0.2977	0.07259
20	8510653	0	13.080	15.71	85.63	520.0	0.10750	0.12700	0.04568	0.031100	...	14.500	20.49	96.09	630.5	0.13120	0.27760	0.18900	0.07283	0.3184	0.08183
21	8510824	0	9.504	12.44	60.34	273.9	0.10240	0.06492	0.02956	0.020760	...	10.230	15.66	65.13	314.9	0.13240	0.11480	0.08867	0.06227	0.2450	0.07773
37	854941	0	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.029230	...	13.300	22.81	84.46	545.9	0.09701	0.04619	0.04833	0.05013	0.1987	0.06169
46	85713702	0	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005917	...	8.964	21.96	57.26	242.2	0.12970	0.13570	0.06880	0.02564	0.3105	0.07409
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
558	925277	0	14.590	22.68	96.39	657.1	0.08473	0.13300	0.10290	0.037360	...	15.480	27.27	105.90	733.5	0.10260	0.31710	0.36620	0.11050	0.2258	0.08004
559	925291	0	11.510	23.93	74.52	403.5	0.09261	0.10210	0.11120	0.041050	...	12.480	37.16	82.28	474.2	0.12980	0.25170	0.36300	0.09653	0.2112	0.08732
560	925292	0	14.050	27.15	91.38	600.4	0.09929	0.11260	0.04462	0.043040	...	15.300	33.17	100.20	706.7	0.12410	0.22640	0.13260	0.10480	0.2250	0.08321
561	925311	0	11.200	29.37	70.67	386.0	0.07449	0.03558	0.00000	0.000000	...	11.920	38.30	75.19	439.6	0.09267	0.05494	0.00000	0.00000	0.1566	0.05905
568	92751	0	7.760	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.000000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.00000	0.00000	0.2871	0.07039

357 rows × 32 columns

Rule Based Classifier

This classifier uses the following features from the data set:

1. for cell size - perimeter_0
2. for cell shape - compactness_2
3. for cell homogenity - concave points_2
4. for cell texture - texture_2

In [7]:

```
df_table_dataset
```

Out[7]:

	id	malignant	radius_0	texture_0	perimeter_0	area_0	smoothness_0	compactness_0	concavity_0	concave points_0	...	radius_2	texture_2	perimeter_2	area_2	smoothness_2	compactness_2	concavity_2	concave points_2	symmetry_2	fractal dimension_2
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	926682	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	926954	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	927241	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	92751	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

569 rows × 32 columns

In [8]:

```
df_rule_based_classifier = df_table_dataset[['perimeter_0','compactness_2','concave points_2', 'texture_2']]
```

In [9]:

```
df_rule_based_classifier
```

Out[9]:

	perimeter_0	compactness_2	concave points_2	texture_2
0	122.80	0.66560	0.2654	17.33
1	132.90	0.18660	0.1860	23.41
2	130.00	0.42450	0.2430	25.53
3	77.58	0.86630	0.2575	26.50
4	135.10	0.20500	0.1625	16.67
...	...	...	...	...
564	142.00	0.21130	0.2216	26.40
565	131.20	0.19220	0.1628	38.25
566	108.30	0.30940	0.1418	34.12
567	140.10	0.86810	0.2650	39.42
568	47.92	0.06444	0.0000	30.37

569 rows × 4 columns

In [10]:

```
X_our_model = df_rule_based_classifier.to_numpy()
```

In [11]:

```
X_our_model
```

Out[11]:

```
array([[1.228e+02, 6.656e-01, 2.654e-01, 1.733e+01],
       [1.329e+02, 1.866e-01, 1.860e-01, 2.341e+01],
       [1.300e+02, 4.245e-01, 2.430e-01, 2.553e+01],
       ...,
       [1.083e+02, 3.094e-01, 1.418e-01, 3.412e+01],
       [1.401e+02, 8.681e-01, 2.650e-01, 3.942e+01],
       [4.792e+01, 6.444e-02, 0.000e+00, 3.037e+01]])
```

In [12]:

```
y_our_model = df_table_dataset['malignant'].to_numpy()
```

In [13]:

```
y_our_model
```

Out[13]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0])
```

In [14]:

```
X_our_model.shape
```

Out[14]:

```
(569, 4)
```

In [15]:

```
def PredictionOurModel(values, X_our_model):
    labels = []
    for i in range(X_our_model.shape[0]):
        cell_size = X_our_model[i,0]
        cell_shape = X_our_model[i,1]
        cell_homogenity = X_our_model[i,2]
        cell_texture = X_our_model[i,3]
        if (cell_size >= values[0] or cell_shape >= values[1] or cell_homogenity >= values[2] or cell_texture >= values[3]):
            label = 1
        else:
            label = 0
        labels.append(label)
    return labels
```

In [16]:

```
values = [114.5, 0.4, 0.1747 , 35]
```

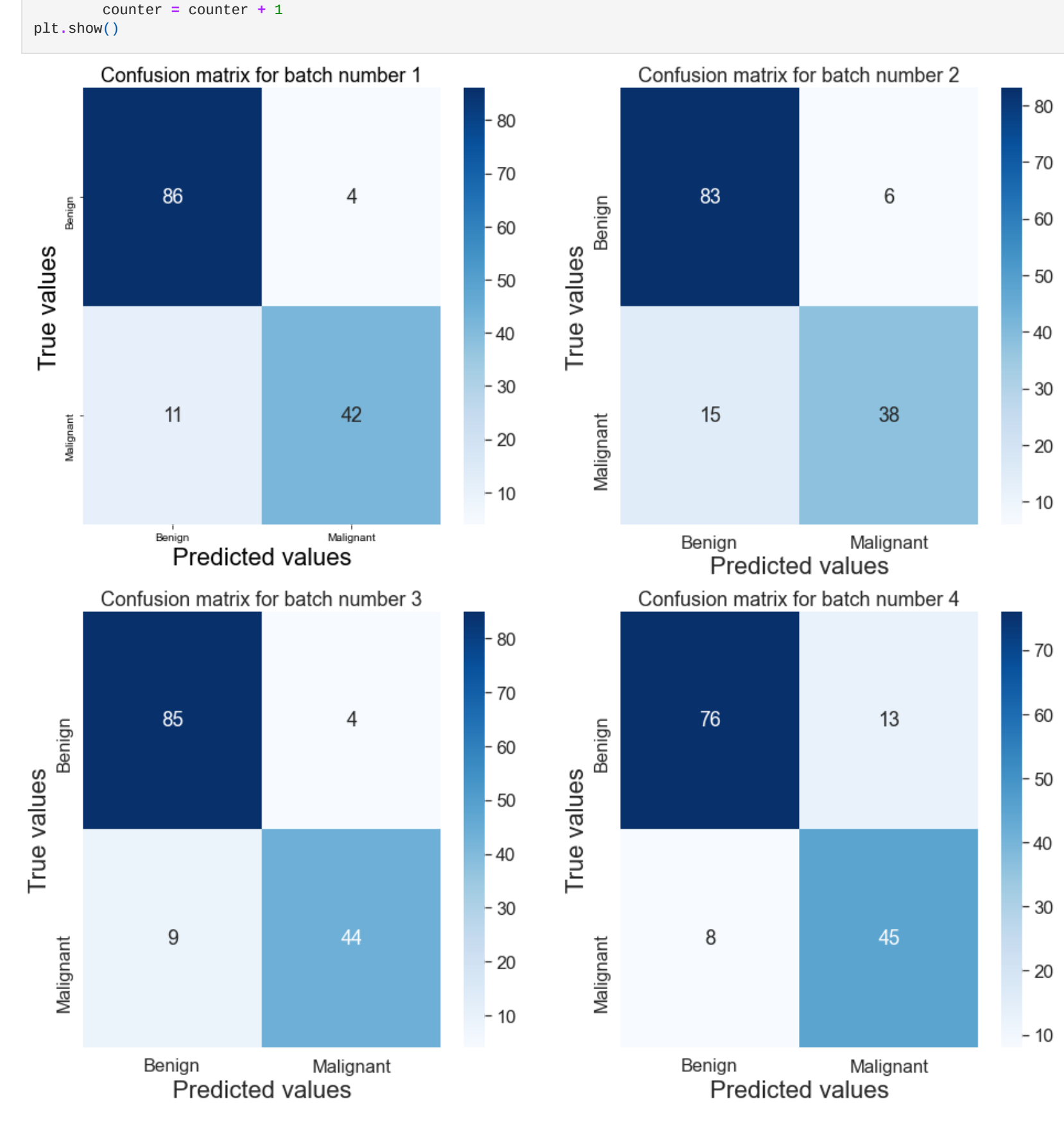
In [17]:

```
X = X_our_model
y = y_our_model

skf = StratifiedKFold(n_splits = 4)


accuracies = []
cf_score = []

for train_index , test_index in skf.split(X,y):
    X_train , X_test = X[train_index], X[test_index]
    y_train , y_test = y[train_index] , y[test_index]

    y_pred = PredictionOurModel(values, X_test)
    accuracy = accuracy_score(y_pred, y_test)
    accuracies.append(accuracy)

    cf_array = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cf_array, ['Benign', 'Malignant'], ['Benign', 'Malignant'])
    cf_score.append(cf_array)

mean_accuracy = sum(accuracies)/len(accuracies)

print('accuracy for each fold - {}'.format(accuracies))
print('Mean accuracy : {}'.format(mean_accuracy))
print("Confusion matrix:")
# print(cf_array)
print('Confusion matrix of each fold - {}'.format(cf_score))
```

```
accuracy for each fold - [0.8951048951048951, 0.852112676056338, 0.9084507042253521, 0.852112676056338]
Mean accuracy : 0.8769452378607308
Confusion matrix:
Confusion matrix of each fold - [array([[86,  4],
       [11, 42]], dtype=int64), array([[83,  6],
       [15, 38]], dtype=int64), array([[85,  4],
       [ 9, 44]], dtype=int64), array([[76, 13],
       [ 8, 45]], dtype=int64)]
```

In [18]:

```
plt.figure(figsize=(16,16))
counter = 0
for i in range(2):
    for j in range(2):
        df_cm = pd.DataFrame(cf_score[counter], ['Benign', 'Malignant'], ['Benign', 'Malignant'])
        plt.subplot(2, 2, counter + 1)
        sn.set(font_scale=1.5,) # Adjust font size
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 18},cmap='Blues')
        plt.xlabel('Predicted values', fontsize=22)
        plt.ylabel('True values', fontsize=22)
        plt.title(('Confusion matrix for batch number ' + str(counter + 1)), fontsize=18)
        counter = counter + 1
plt.show()
```



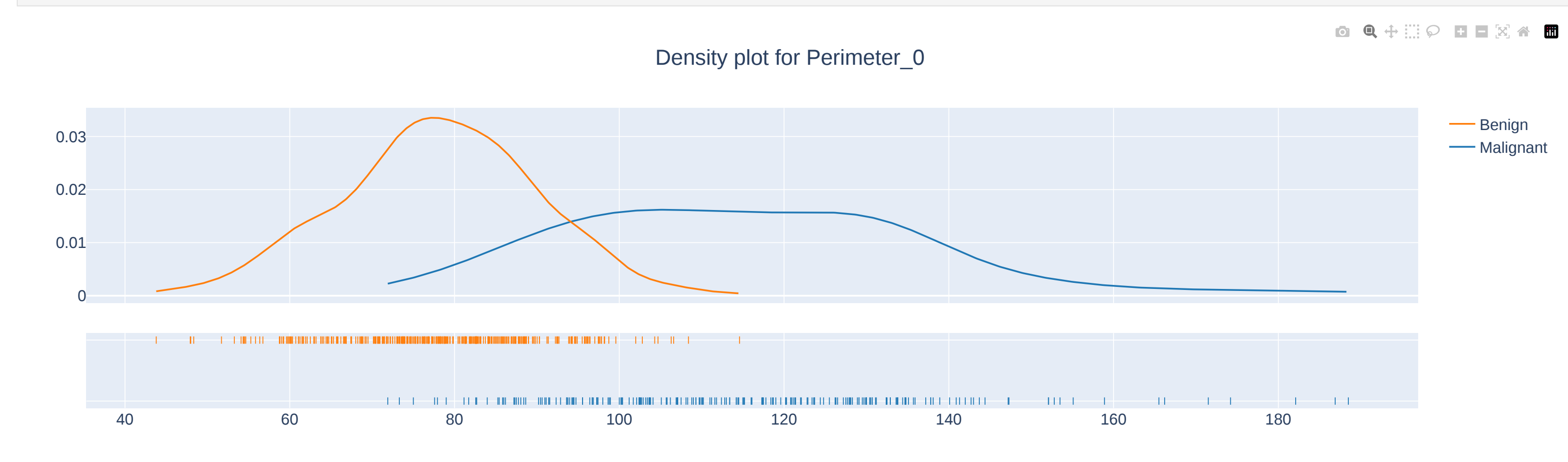
Justification for chossing the above mentioned features in our classifier

In [19]:

```
import plotly.figure_factory as ff

def plot_density(df, column):
    # Separate data by malignant status
    malignant = df_table_dataset.loc[df_table_dataset['malignant'] == 1, column]
    benign = df_table_dataset.loc[df_table_dataset['malignant'] == 0, column]

    # Create distplot with custom bin_size
    fig = ff.create_distplot([malignant, benign], ['Malignant', 'Benign'], bin_size=1, show_hist=False)
    fig.update_layout(title_text=f'Density plot for {column.capitalize()}', title_x=0.5, font_size=18)
    fig.show()
```

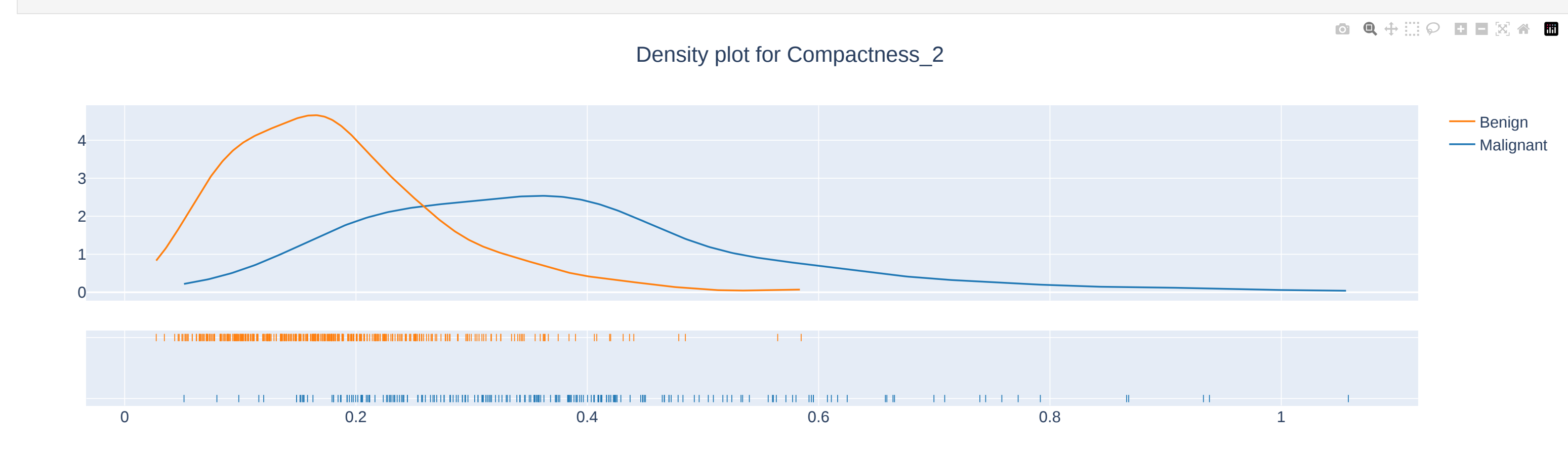
In [20]:

```
plot_density(df_table_dataset, 'perimeter_0')
```



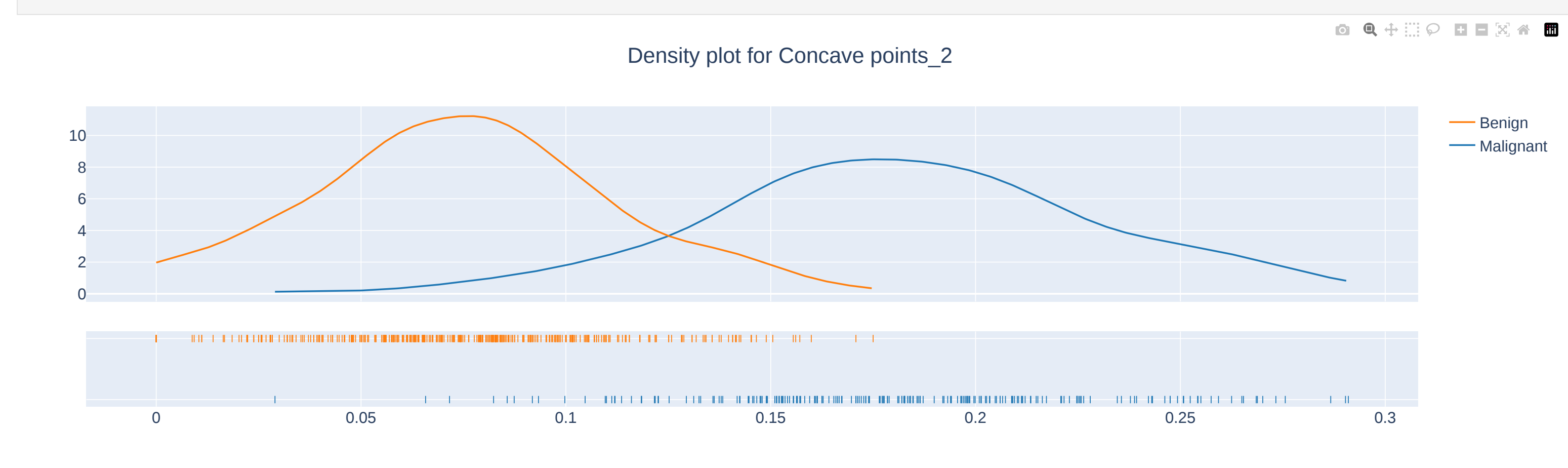
In [21]:

```
plot_density(df_table_dataset, 'compactness_2')
```



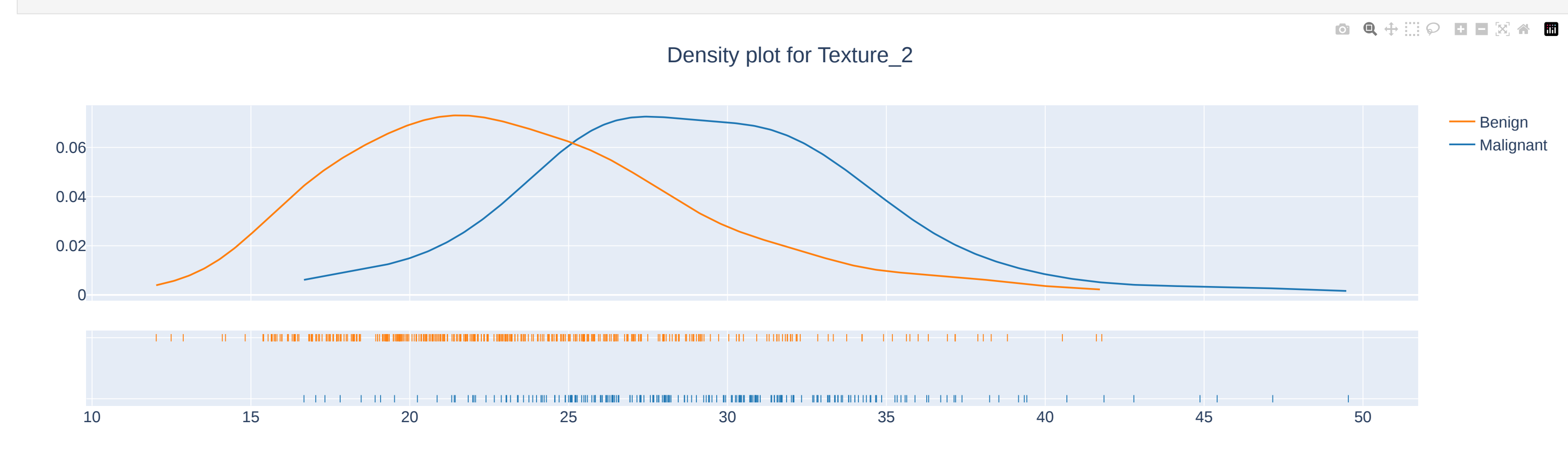
In [22]:

```
plot_density(df_table_dataset, 'concave points_2')
```



In [23]:

```
plot_density(df_table_dataset, 'texture_2')
```



Random Forest Classifier

In [24]:

```
df_table_dataset
```

Out[24]:

	id	malignant	radius_0	texture_0	perimeter_0	area_0	smoothness_0	compactness_0	concavity_0	concave points_0	...	radius_2	texture_2	perimeter_2	area_2	smoothness_2	compactness_2	concavity_2	concave points_2	symmetry_2	fractal dimension_2
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	926682	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	926954	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	927241	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	92751	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

569 rows × 32 columns

In [25]:

```
X_random_forest_model = df_table_dataset.drop(['id', 'malignant'], axis = 'columns')
X_random_forest_model = X_random_forest_model.to_numpy()
```

In [26]:

```
X_random_forest_model
```

Out[26]:

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

In [27]:

```
y_random_forest_model = df_table_dataset['malignant'].to_numpy()
```

In [28]:

```
y_random_forest_model
```

Out[28]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0])
```

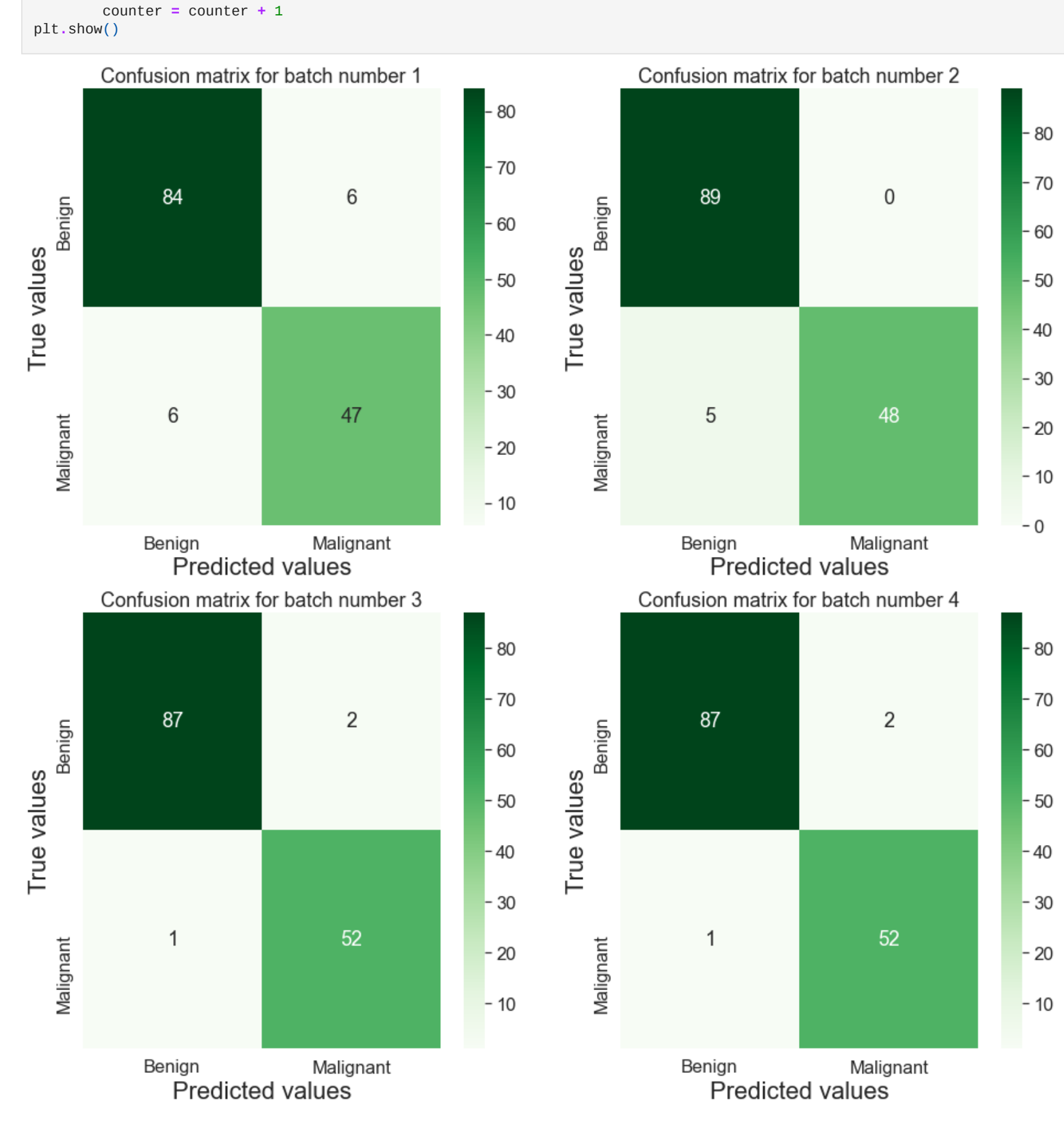
In [29]:

```
X = X_random_forest_model
y = y_random_forest_model

skf = StratifiedKFold(n_splits = 4)

accuracies = []
cf_score = []

for train_index , test_index in skf.split(X,y):
    X_train , X_test = X[train_index], X[test_index]
    y_train , y_test = y[train_index] , y[test_index]

    clf = RandomForestClassifier(n_estimators= 100, max_depth=10)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_pred, y_test)
    accuracies.append(accuracy)

    cf_array = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cf_array, ['Benign', 'Malignant'], ['Benign', 'Malignant'])
    cf_score.append(cf_array)

mean_accuracy = sum(accuracies)/len(accuracies)

print('accuracy for each fold - {}'.format(accuracies))
print('Mean accuracy : {}'.format(mean_accuracy))
print("Confusion matrix:")
print('Confusion matrix of each fold - {}'.format(cf_score))
```

```
accuracy for each fold - [0.916083916083916, 0.9647887323943662, 0.9788732394366197, 0.9788732394366197]
Mean accuracy : 0.9596547818378804
Confusion matrix:
Confusion matrix of each fold - [array([[84,  6],
       [ 6, 47]], dtype=int64), array([[89,  0],
       [ 5, 48]], dtype=int64), array([[87,  2],
       [ 1, 52]], dtype=int64), array([[87,  2],
       [ 1, 52]], dtype=int64)]
```

In [30]:

```
plt.figure(figsize=(16,16))
counter = 0
for i in range(2):
    for j in range(2):
        df_cm = pd.DataFrame(cf_score[counter], ['Benign', 'Malignant'], ['Benign', 'Malignant'])
        plt.subplot(2, 2, counter + 1)
        sn.set(font_scale=1.5,) # Adjust font size
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 18},cmap='Greens')
        plt.xlabel('Predicted values', fontsize=22)
        plt.ylabel('True values', fontsize=22)
        plt.title(('Confusion matrix for batch number ' + str(counter + 1)), fontsize=18)
        counter = counter + 1
plt.show()
```



Decision Tree

In [31]:

```
df_table_dataset
```

Out[31]:

	id	malignant	radius_0	texture_0	perimeter_0	area_0	smoothness_0	compactness_0	concavity_0	concave points_0	...	radius_2	texture_2	perimeter_2	area_2	smoothness_2	compactness_2	concavity_2	concave points_2	symmetry_2	fractal dimension_2
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	926682	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	926954	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	927241	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	92751	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

569 rows × 32 columns

In [32]:

```
X_decision_tree_model = df_table_dataset.drop(['id', 'malignant'], axis = 'columns')
X_decision_tree_model = X_decision_tree_model.to_numpy()
```

In [33]:

```
y_decision_tree_model = df_table_dataset['malignant'].to_numpy()
```

In [34]:

```
X = X_decision_tree_model
y = y_decision_tree_model

skf = StratifiedKFold(n_splits = 4)

accuracies = []
cf_score = []

for train_index , test_index in skf.split(X,y):
    X_train , X_test = X[train_index], X[test_index]
    y_train , y_test = y[train_index] , y[test_index]

    clf = DecisionTreeClassifier(max_depth=3)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_pred, y_test)
    accuracies.append(accuracy)

    cf_array = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cf_array, ['Benign', 'Malignant'], ['Benign', 'Malignant'])
    cf_score.append(cf_array)

mean_accuracy = sum(accuracies)/len(accuracies)

print('accuracy for each fold - {}'.format(accuracies))
print('Mean accuracy : {}'.format(mean_accuracy))
print("Confusion matrix:")
print('Confusion matrix of each fold - {}'.format(cf_score))
```

```
accuracy for each fold - [0.9090909090909091, 0.9507042253521126, 0.9647887323943662, 0.9014084507042254]
Mean accuracy : 0.9314980793854034
Confusion matrix:
Confusion matrix of each fold - [array([[85,  5],
       [ 8, 45]], dtype=int64), array([[89,  0],
       [ 7, 46]], dtype=int64), array([[87,  2],
       [ 3, 50]], dtype=int64), array([[79, 10],
       [ 4, 49]], dtype=int64)]
```

In [35]:

```
plt.figure(figsize=(16,16))
counter = 0
for i in range(2):
    for j in range(2):
        df_cm = pd.DataFrame(cf_score[counter], ['Benign', 'Malignant'], ['Benign', 'Malignant'])
        plt.subplot(2, 2, counter + 1)
        sn.set(font_scale=1.5,) # Adjust font size
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 18},cmap='Reds')
        plt.xlabel('Predicted values', fontsize=22)
        plt.ylabel('True values', fontsize=22)
        plt.title(('Confusion matrix for batch number ' + str(counter + 1)), fontsize=18)
        counter = counter + 1
plt.show()
```