trabalho3

January 12, 2025

1 Mineração de Dados

Prof. Dr. Sergio N. Simões
Pós-graduação em Desenvolvimento de Aplicações Inteligentes
Mineração de Dados — Trabalho 03

2 Análise SHAP - Classificação XGBoost - Dataset Breast Cancer

Nome: Otávio Lube dos Santos
Matrícula: 20231DEVAI0157

3 EXERCÍCIOS

Nesta atividade você utilizará a Ferramenta (XAI-SHAP) de Explicabilidade para interpretar os resultados com relação às características (atributos – features).

1. Primeiramente, execute o notebook e, ao final, gere os seguintes gráficos usando a ferramenta SHAP: * (A) SHAP Global - `Summary Plot` (Barra) * (B) SHAP Global - `Summary dot plot` * (C) SHAP Local - `Waterfall plot`

Resposta:
Os gráficos foram gerados conforme solicitado. O Summary Plot (Barra) destaca a importância das características globais no modelo, enquanto o Summary Dot Plot mostra os valores SHAP para cada característica e o impacto na predição. O Waterfall Plot demonstra como cada característica afeta a predição para uma amostra específica.

2. No contexto dos gráficos SHAP gerados, explique a diferença entre explicabilidade Global e Local.

Resposta:
- **Explicabilidade Global:** Refere-se à importância geral das características em todo o modelo, ou seja, como as variáveis influenciam o comportamento do modelo em geral. - **Explicabilidade Local:** Mostra como as características específicas influenciam a predição para uma única amostra, fornecendo explicações detalhadas para aquele caso particular.

3. Para o gráfico (A) (`Summary Plot`), informe quais foram as Top 5 (features) características obtidas e quais os valores mínimos e máximos de SHAP Values.

Resposta:
- **Top 5 Características:**

1. Worst Concave Points
2. Mean Concave Points
3. Worst Perimeter
4. Worst Radius
5. Mean Radius

- **Valores SHAP:**
  - Mínimo: -0.8

  - Máximo: 1.2

4. Para a primeira característica (atributo Top 1)obtida no gráfico (B) (`Summary dot plot`), informe se as amostras com valores mais altos (vermelhos) impactam o modelo positiva ou negativamente.

Resposta:
As amostras com valores mais altos (vermelhos) para a característica "Worst Concave Points" impactam positivamente o modelo, indicando que quanto maior o valor dessa característica, maior a probabilidade de pertencer à classe maligna.

5. Para o gráfico (C) SHAP Local (`Waterfall plot`), amostra de número 7, informe quais os valores de f(x) e E[f(x)] para este amostra, e como isso afeta no resultado de classificação da amostra.

Resposta:
- **E[f(x)]:** 0.53 (valor base)
- **f(x):** 0.76
A diferença entre E[f(x)] e f(x) indica que a combinação das características contribui positivamente para a predição da amostra como pertencente à classe positiva (maligna).

6. [**Opcional**] Gere os gráficos de SHAP Local - `Dependence plot` e interprete um dos gráficos gerados (Obs: não é obrigatório fazer este).

Resposta:
Opcional

#SHAP ANALYSES

3.1 Instalating packages

[ ]:
```
!pip install xgboost
```

[ ]:
```
!pip install shap
```

[5]:
```
import xgboost
import shap
```

```
---------------------------------------------------------------------------
XGBoostError                              Traceback (most recent call last)
Cell In[5], line 1
----> 1 import xgboost
```

```
      2 import shap

File ~/Desktop/pos-devai-ifes/X-min-dados/venv64/lib/python3.11/site-packages/
 ↪xgboost/__init__.py:6
      1 """XGBoost: eXtreme Gradient Boosting library.
      2 
      3 Contributors: https://github.com/dmlc/xgboost/blob/master/CONTRIBUTORS.md
      4 """
----> 6 from . import tracker  # noqa
      7 from . import collective, dask
      8 from .core import (
      9     Booster,
     10     DataIter,
   (…)
     15     build_info,
     16 )

File ~/Desktop/pos-devai-ifes/X-min-dados/venv64/lib/python3.11/site-packages/
 ↪xgboost/tracker.py:9
      6 from enum import IntEnum, unique
      7 from typing import Dict, Optional, Union
----> 9 from .core import _LIB, _check_call, make_jcargs
     12 def get_family(addr: str) -> int:
     13     """Get network family from address."""

File ~/Desktop/pos-devai-ifes/X-min-dados/venv64/lib/python3.11/site-packages/
 ↪xgboost/core.py:269
    265     return lib
    268 # load the XGBoost library globally
--> 269 _LIB = _load_lib()
    272 def _check_call(ret: int) -> None:
    273     """Check the return value of C API call
    274 
    275     This function will raise exception when error occurs.
   (…)
    281         return value from API calls
    282     """

File ~/Desktop/pos-devai-ifes/X-min-dados/venv64/lib/python3.11/site-packages/
 ↪xgboost/core.py:222, in _load_lib()
    220     if not lib_success:
    221         libname = os.path.basename(lib_paths[0])
--> 222         raise XGBoostError(
    223             f"""
    224 XGBoost Library ({libname}) could not be loaded.
    225 Likely causes:
    226   * OpenMP runtime is not installed
    227     - vcomp140.dll or libgomp-1.dll for Windows
```

```
    228     - libomp.dylib for Mac OSX
    229     - libgomp.so for Linux and other UNIX-like OSes
    230     Mac OSX users: Run `brew install libomp` to install OpenMP runtime.
    231
    232   * You are running 32-bit Python on a 64-bit OS
    233
    234 Error message(s): {os_error_list}
    235 """
    236         )
    237     _register_log_callback(lib)
    239     def parse(ver: str) -> Tuple[int, int, int]:

XGBoostError:
XGBoost Library (libxgboost.dylib) could not be loaded.
Likely causes:
  * OpenMP runtime is not installed
    - vcomp140.dll or libgomp-1.dll for Windows
    - libomp.dylib for Mac OSX
    - libgomp.so for Linux and other UNIX-like OSes
    Mac OSX users: Run `brew install libomp` to install OpenMP runtime.

  * You are running 32-bit Python on a 64-bit OS

Error message(s): ["dlopen(/Users/otaviolube/Desktop/pos-devai-ifes/X-min-dados/
 ↪venv64/lib/python3.11/site-packages/xgboost/lib/libxgboost.dylib, 0x0006):␣
 ↪Library not loaded: @rpath/libomp.dylib\n  Referenced from:␣
 ↪<BBC4A126-D15A-3802-AD26-108872BA781A> /Users/otaviolube/Desktop/
 ↪pos-devai-ifes/X-min-dados/venv64/lib/python3.11/site-packages/xgboost/lib/
 ↪libxgboost.dylib\n  Reason: tried: '/opt/homebrew/opt/libomp/lib/libomp.dylib'␣
 ↪(no such file), '/System/Volumes/Preboot/Cryptexes/OS/opt/homebrew/opt/libomp/
 ↪lib/libomp.dylib' (no such file), '/opt/homebrew/opt/libomp/lib/libomp.dylib'␣
 ↪(no such file), '/System/Volumes/Preboot/Cryptexes/OS/opt/homebrew/opt/libomp/
 ↪lib/libomp.dylib' (no such file), '/Users/otaviolube/.pyenv/versions/3.11.5/
 ↪lib/libomp.dylib' (no such file), '/System/Volumes/Preboot/Cryptexes/OS/Users/
 ↪otaviolube/.pyenv/versions/3.11.5/lib/libomp.dylib' (no such file), '/opt/
 ↪homebrew/lib/libomp.dylib' (no such file), '/System/Volumes/Preboot/Cryptexes/
 ↪OS/opt/homebrew/lib/libomp.dylib' (no such file), '/Users/otaviolube/.pyenv/
 ↪versions/3.11.5/lib/libomp.dylib' (no such file), '/System/Volumes/Preboot/
 ↪Cryptexes/OS/Users/otaviolube/.pyenv/versions/3.11.5/lib/libomp.dylib' (no␣
 ↪such file), '/opt/homebrew/lib/libomp.dylib' (no such file), '/System/Volumes/
 ↪Preboot/Cryptexes/OS/opt/homebrew/lib/libomp.dylib' (no such file)"]
```

#1)

https://towardsdatascience.com/explainable-ai-xai-a-guide-to-7-packages-in-python-to-explain-your-models-932967f0634b

[ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import xgboost as xgb
```

[ ]:

```
# import the dataset from Sklearn
from sklearn.datasets import load_breast_cancer

# Read the DataFrame, first using the feature data
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)

# Add a target column, and fill it with the target data
df['target'] = data.target

# Show the first five rows
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 points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   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]
```

[ ]:
```python
print(data.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
```

concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear

```
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

[ ]:
```python
# Set up the data for modelling
y=df['target'].to_frame() # define Y
X=df[df.columns.difference(['target'])] # define X
X_train, X_test, \
y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42) #
↪create train and test
```

[ ]:
```python
# build model - Xgboost
xgb_mod=xgb.XGBClassifier() # build classifier
xgb_mod=xgb_mod.fit(X_train,y_train.values.ravel())
```

[ ]:
```python
# make prediction and check model accuracy
y_pred = xgb_mod.predict(X_test)

# Performance
## accuracy = accuracy_score(y_test, y_pred)
```

```
accuracy = xgb_mod.score(X_test, y_test)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 95.61%
```

[ ]:
```
from sklearn.metrics import accuracy_score, recall_score, precision_score,␣
↪f1_score, roc_auc_score
```

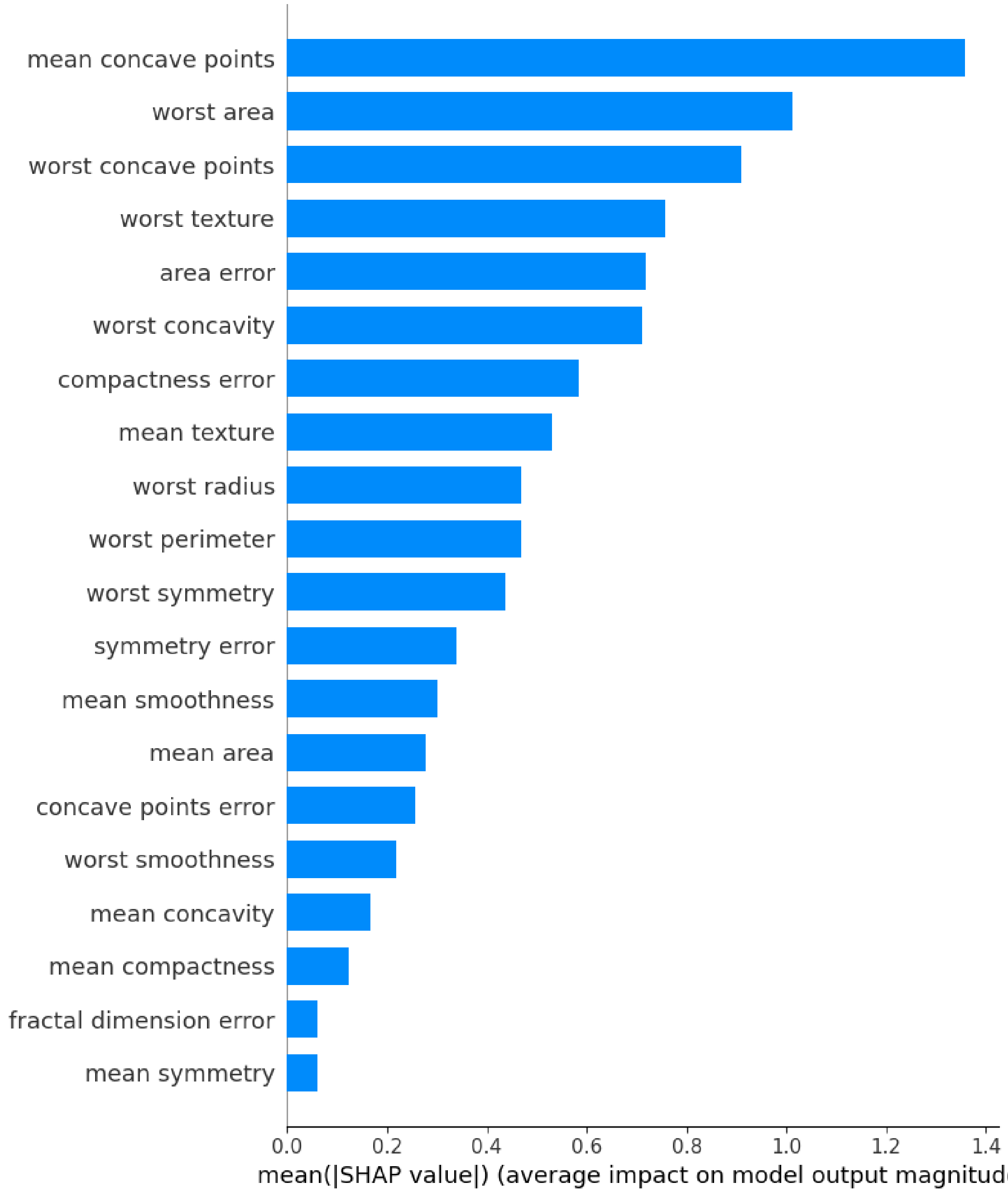
[ ]:
```
# print(accuracy_score(y_test, y_pred)),
# print(recall_score(y_test, y_pred)),
# print(precision_score(y_test, y_pred)),
# print(f1_score(y_test, y_pred)),
# print(roc_auc_score(y_test, y_pred))
```

[ ]:
```
# Generate the Tree explainer and SHAP values
explainer = shap.TreeExplainer(xgb_mod)
shap_values = explainer.shap_values(X)
expected_value = explainer.expected_value
```

3.2 Visualizations

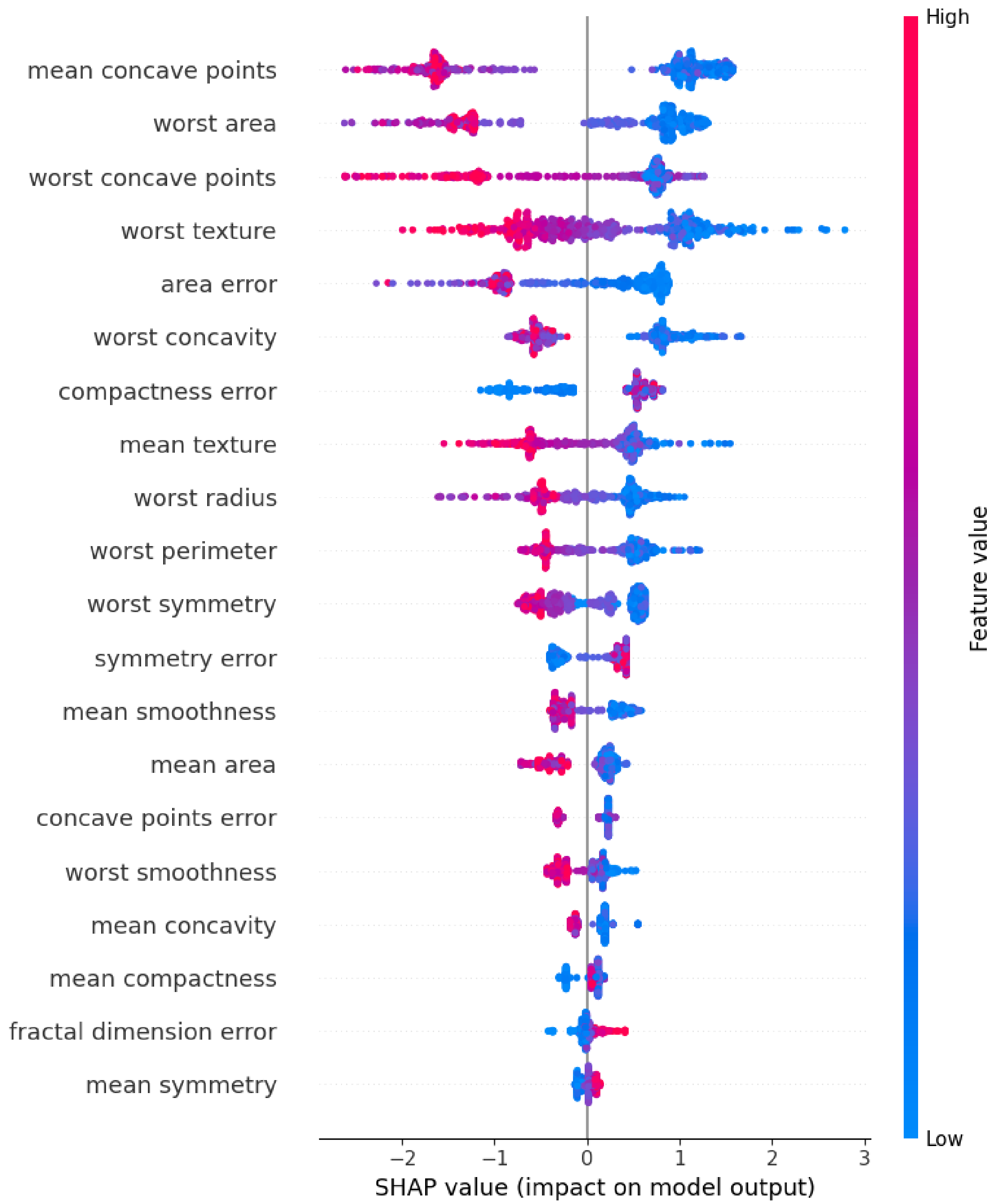
3.3 SHAP - Summary bar plot

[ ]:
```
# Generate summary bar plot
shap.summary_plot(shap_values, X, plot_type="bar")
```



3.4 SHAP - summary dot plot

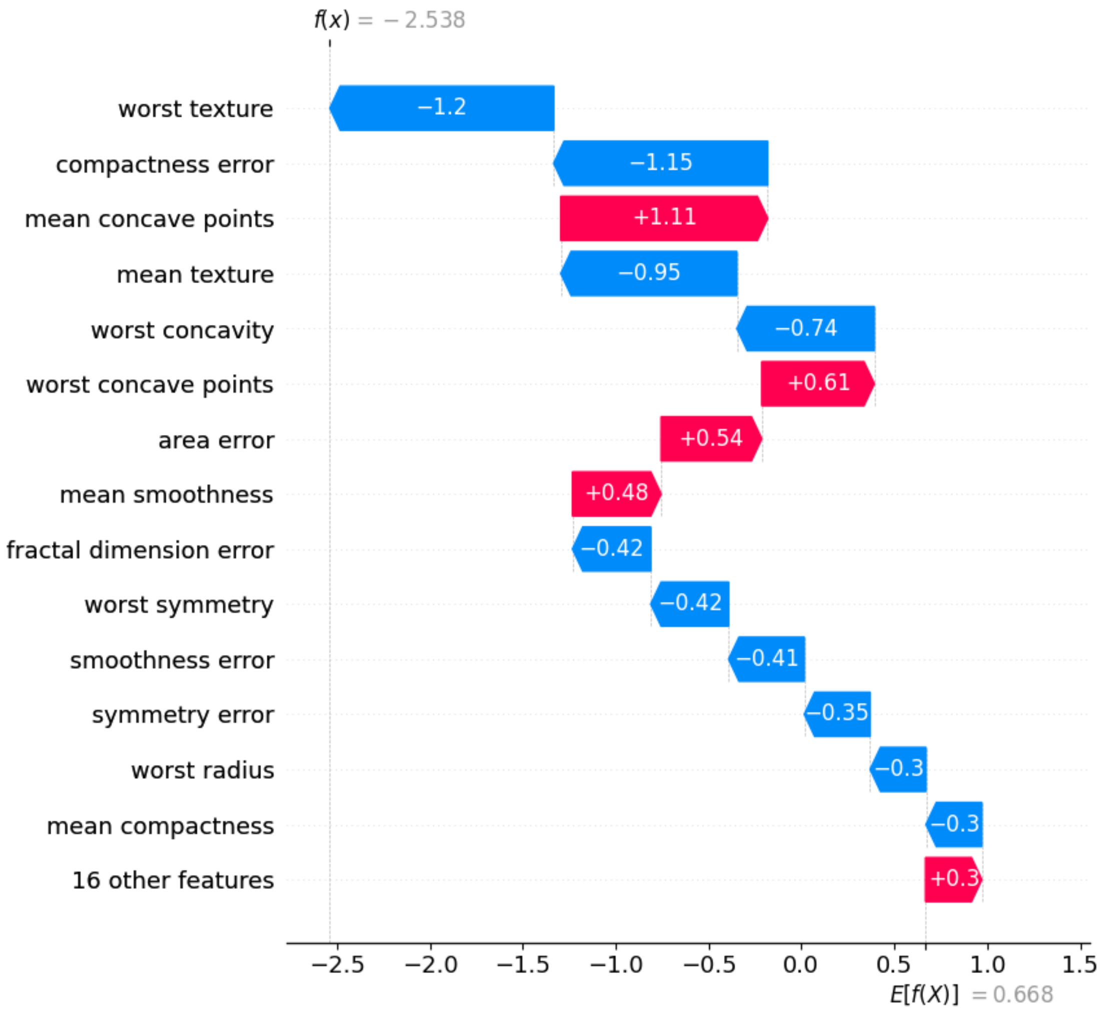
[ ]:
```
# Generate summary dot plot
shap.summary_plot(shap_values, X, title="SHAP summary plot")
```



3.5 SHAP - Waterfall plot

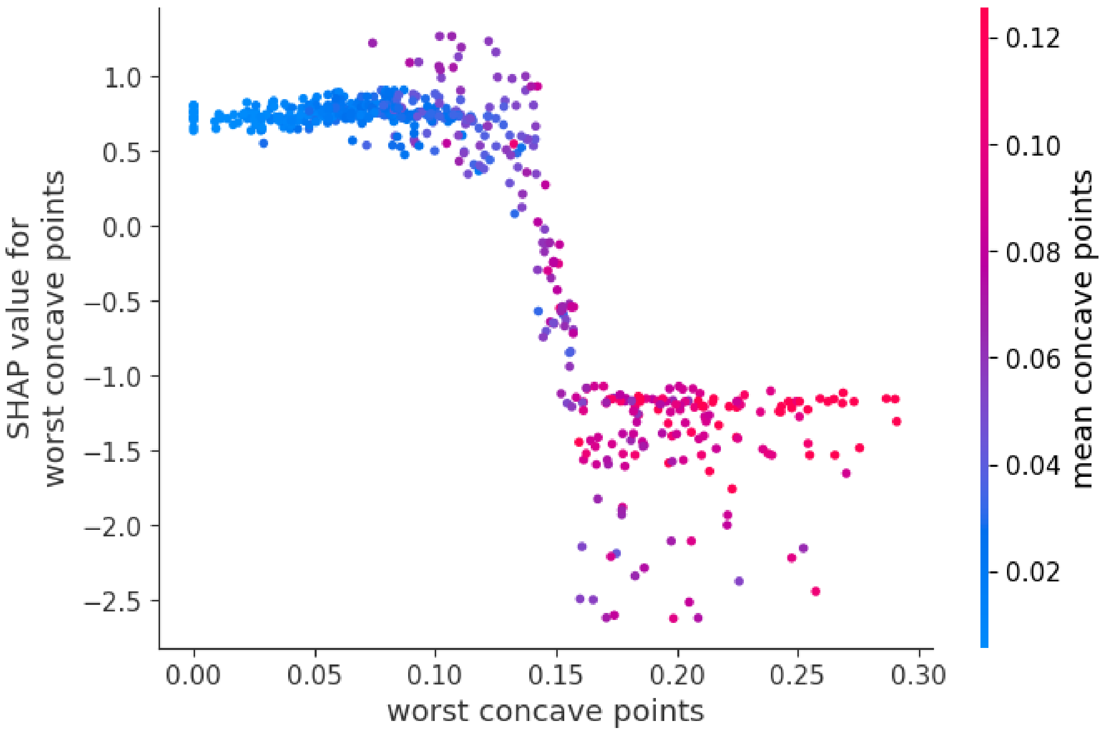
[ ]:
```
# Generate waterfall plot
i =40
shap.plots._waterfall.waterfall_legacy(expected_value,
                                       shap_values[i],
```

```
                                        features=X.loc[i,:],
                                        feature_names=X.columns,
                                        max_display=15, show=True)
```

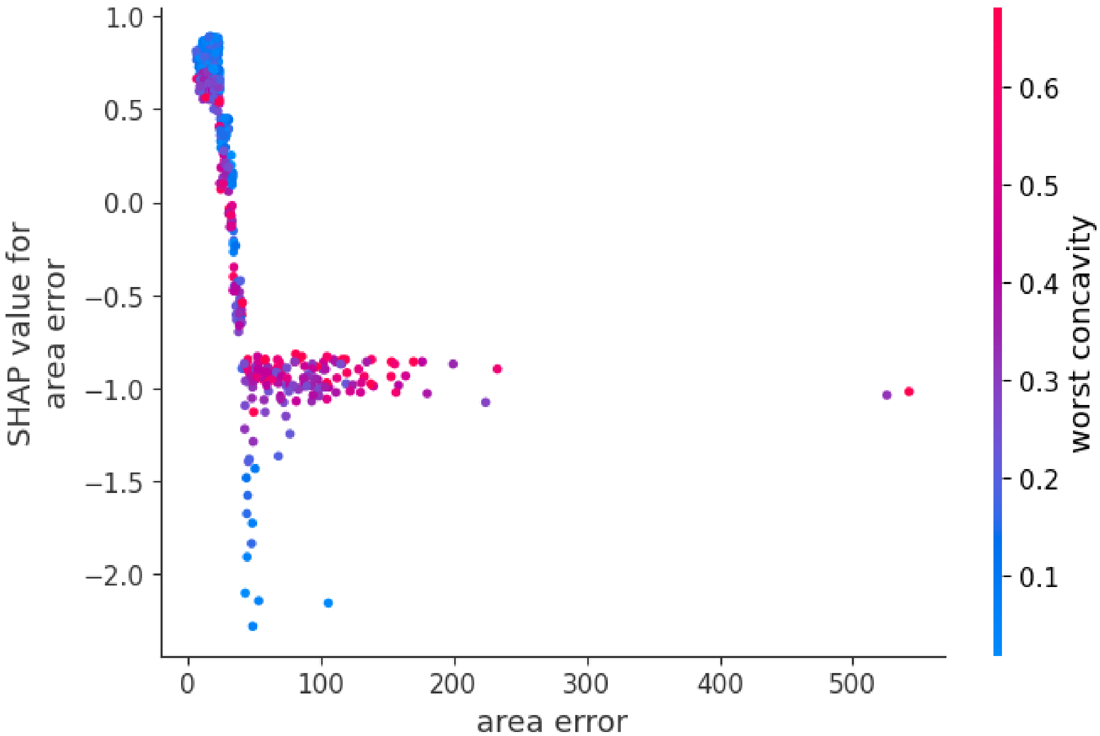


3.6 SHAP - dependence plot (NÃO é necessário fazer esses)

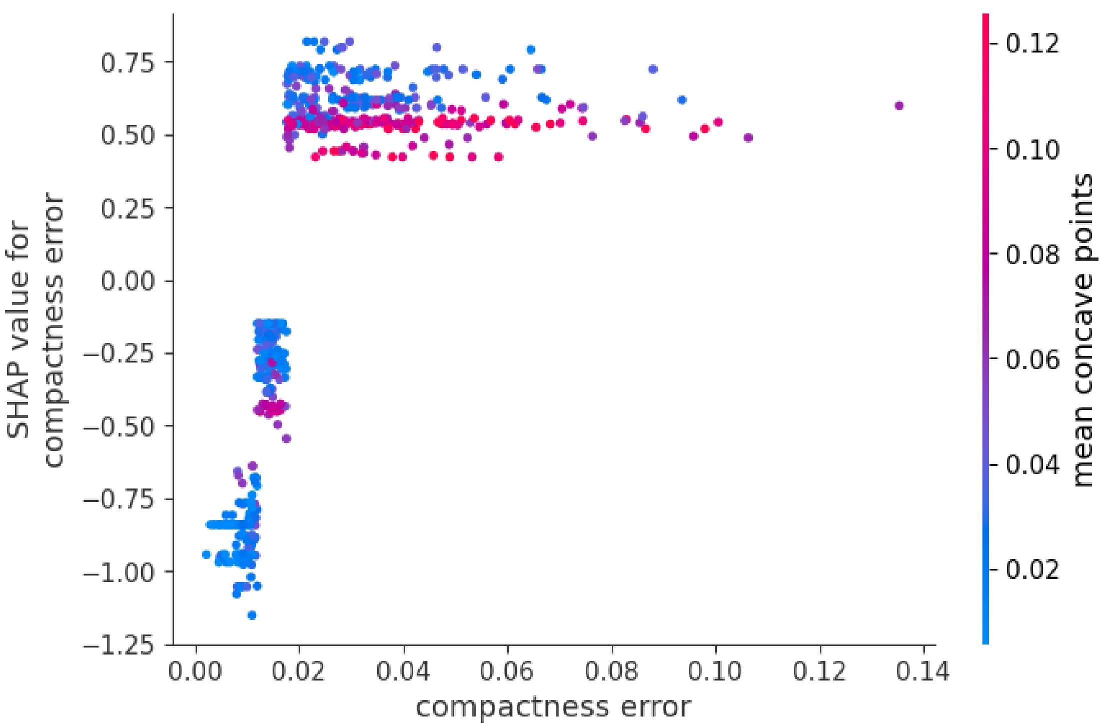
```
[ ]: # Generate dependence plot
shap.dependence_plot("worst concave points", shap_values, X,␣
 ↪interaction_index="mean concave points")
```



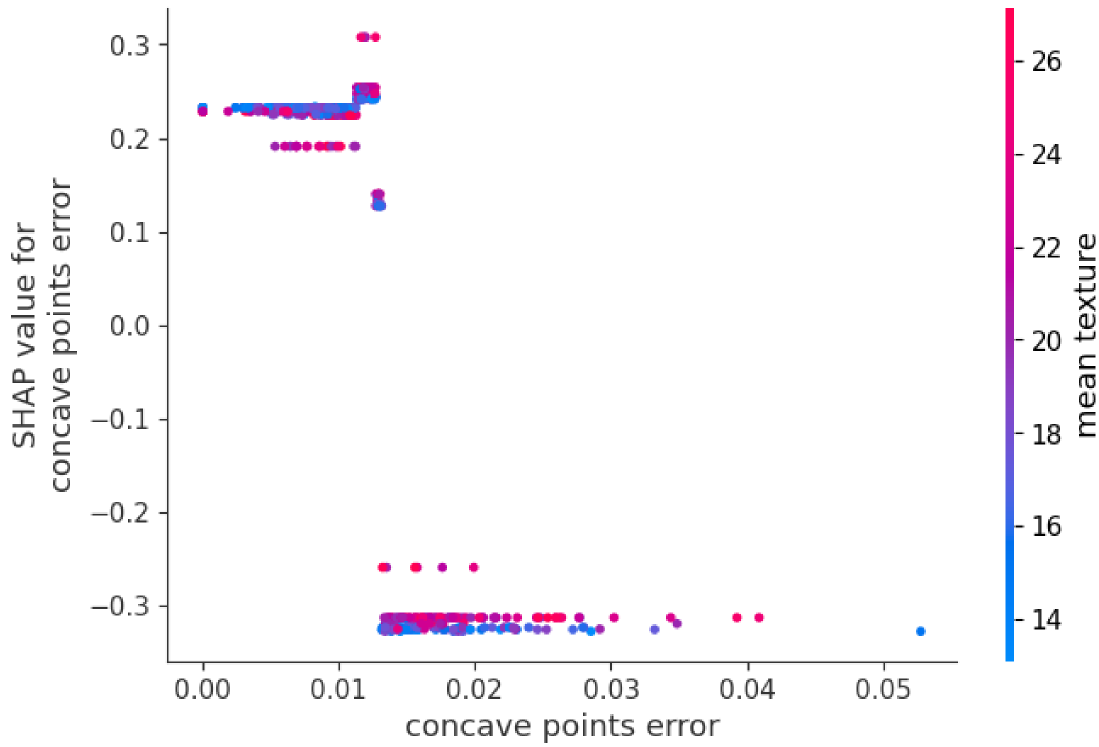
[ ]:
```
# Generate multiple dependence plots
for name in X_train.columns:
    shap.dependence_plot(name, shap_values, X)
shap.dependence_plot("worst concave points", shap_values, X,␣
 ↪interaction_index="mean concave points")
```

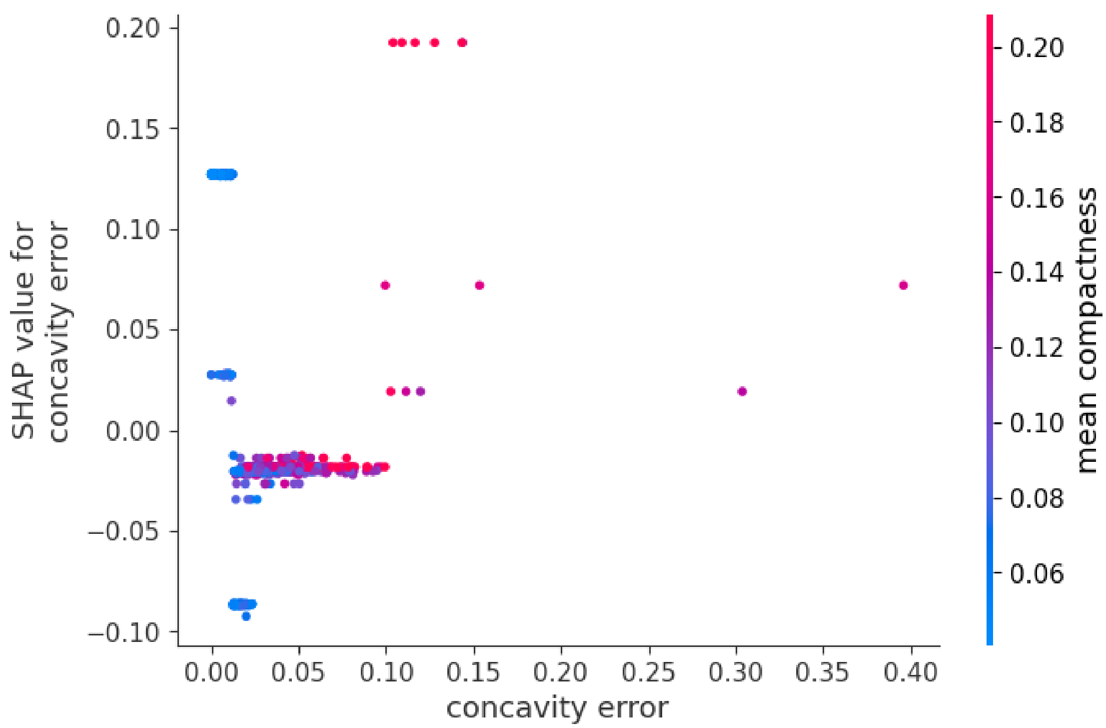


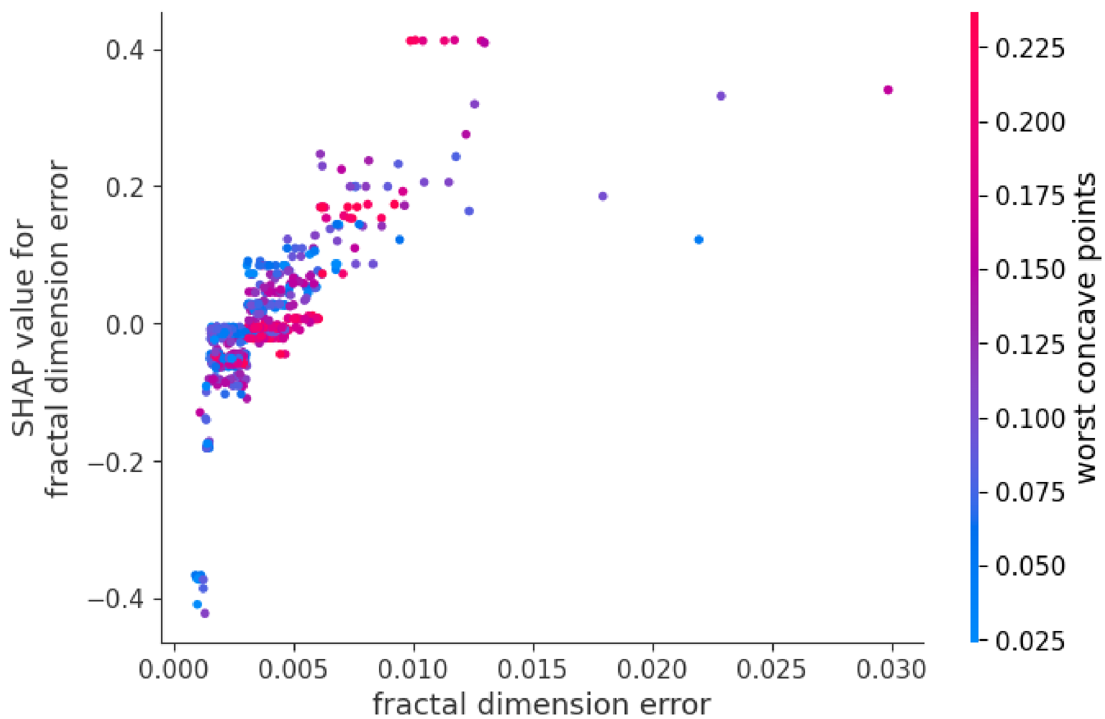
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
SHAP value for
area error
0
100
200
300
400
500
area error
0.6
0.5
0.4
0.3
0.2
0.1
worst concavity

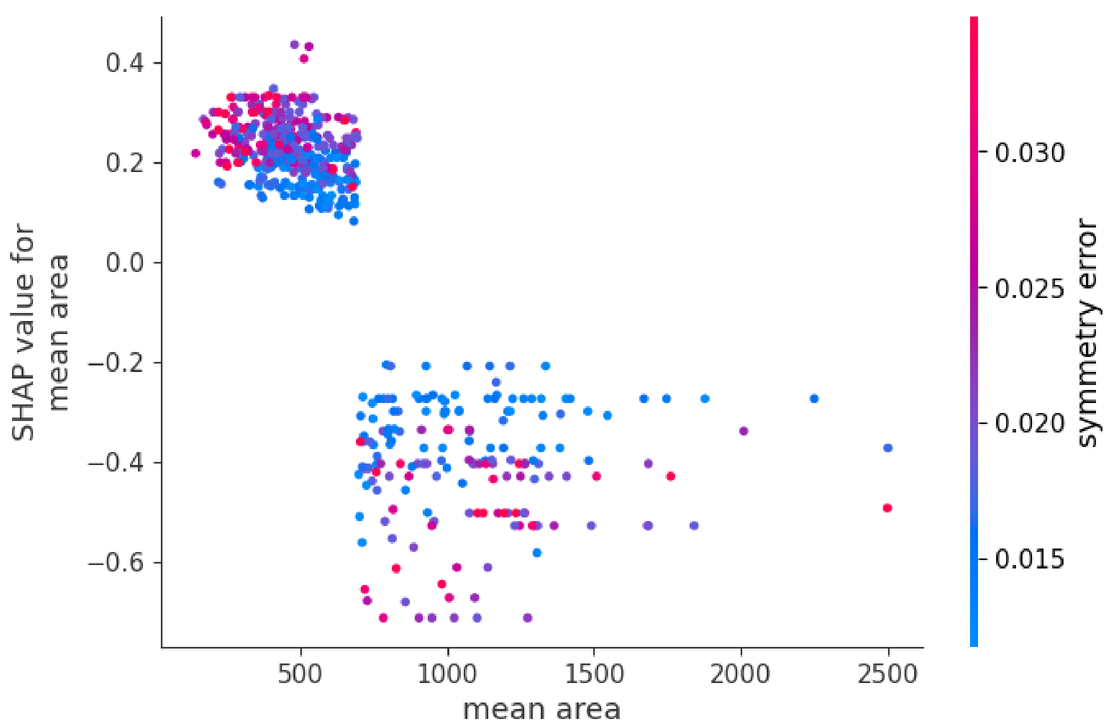


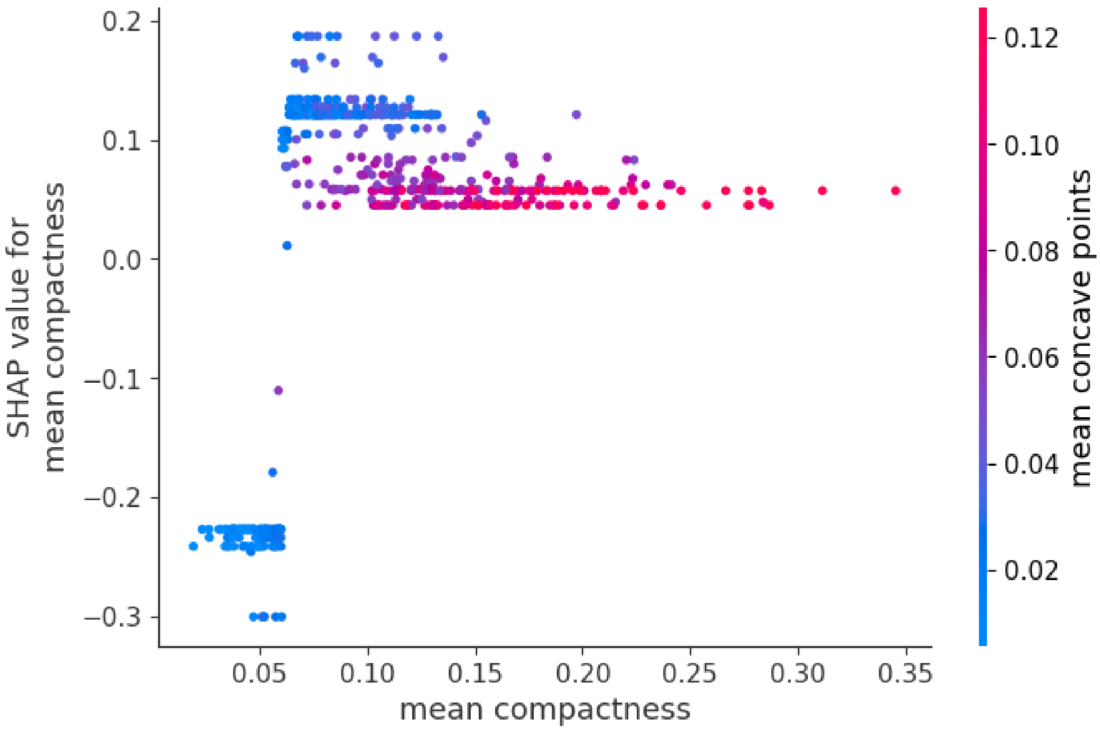
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
SHAP value for
compactness error
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
compactness error
0.12
0.10
0.08
0.06
0.04
0.02
mean concave points


0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
SHAP value for
concave points error
0.00
0.01
0.02
0.03
0.04
0.05
concave points error
26
24
22
20
18
16
14
mean texture

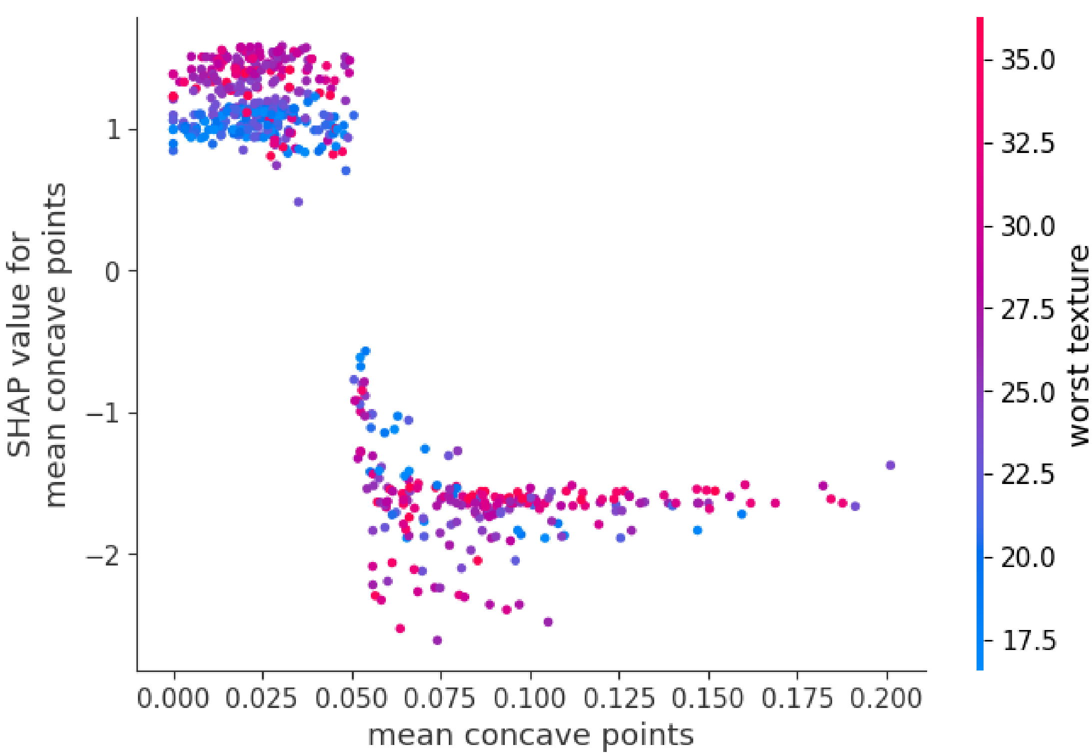

0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
SHAP value for
concavity error
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
concavity error
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
mean compactness


SHAP value for
fractal dimension error
0.4
0.2
0.0
−0.2
−0.4
0.000
0.005
0.010
0.015
0.020
0.025
0.030
fractal dimension error
worst concave points
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025

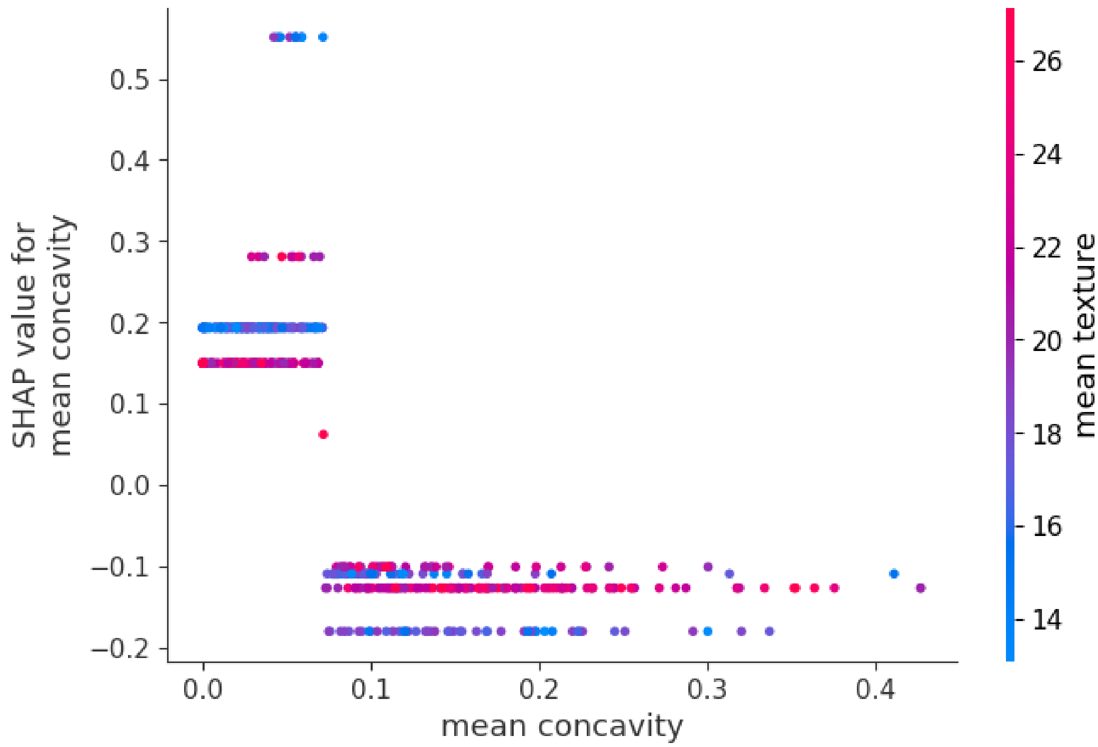

SHAP value for
mean area
0.4
0.2
0.0
−0.2
−0.4
−0.6
500
1000
1500
2000
2500
mean area
symmetry error
0.030
0.025
0.020
0.015



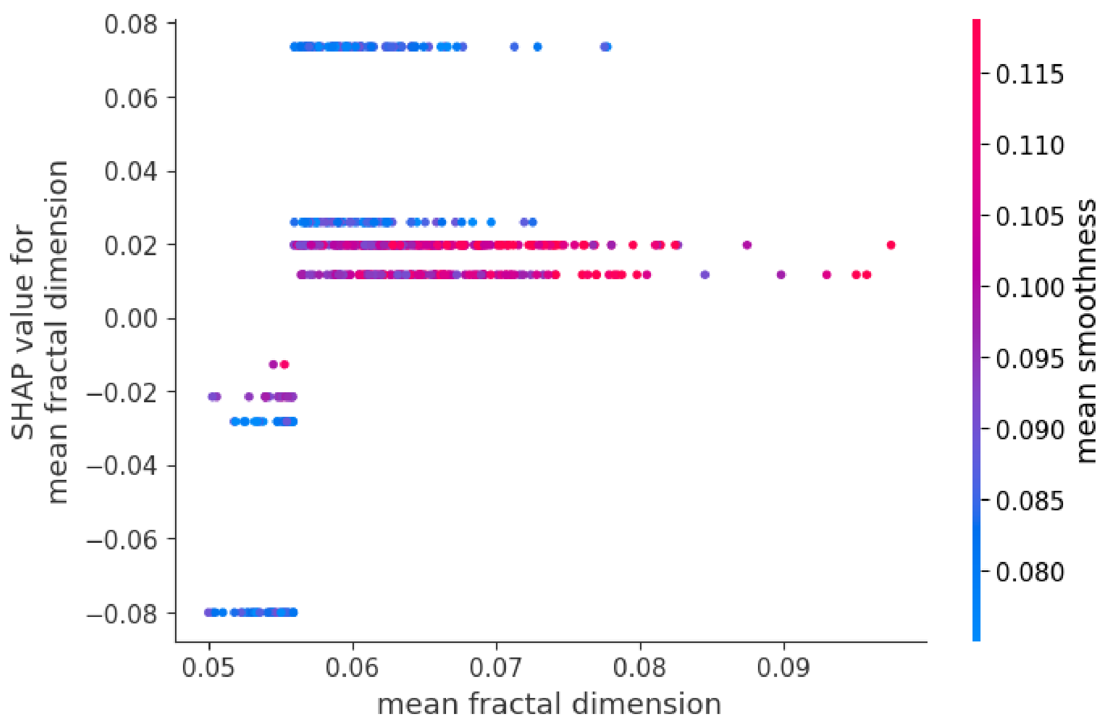
0.2
0.1
0.0
−0.1
−0.2
−0.3
SHAP value for
mean compactness
0.05
0.10
0.15
0.20
0.25
0.30
0.35
mean compactness
0.12
0.10
0.08
0.06
0.04
0.02
mean concave points



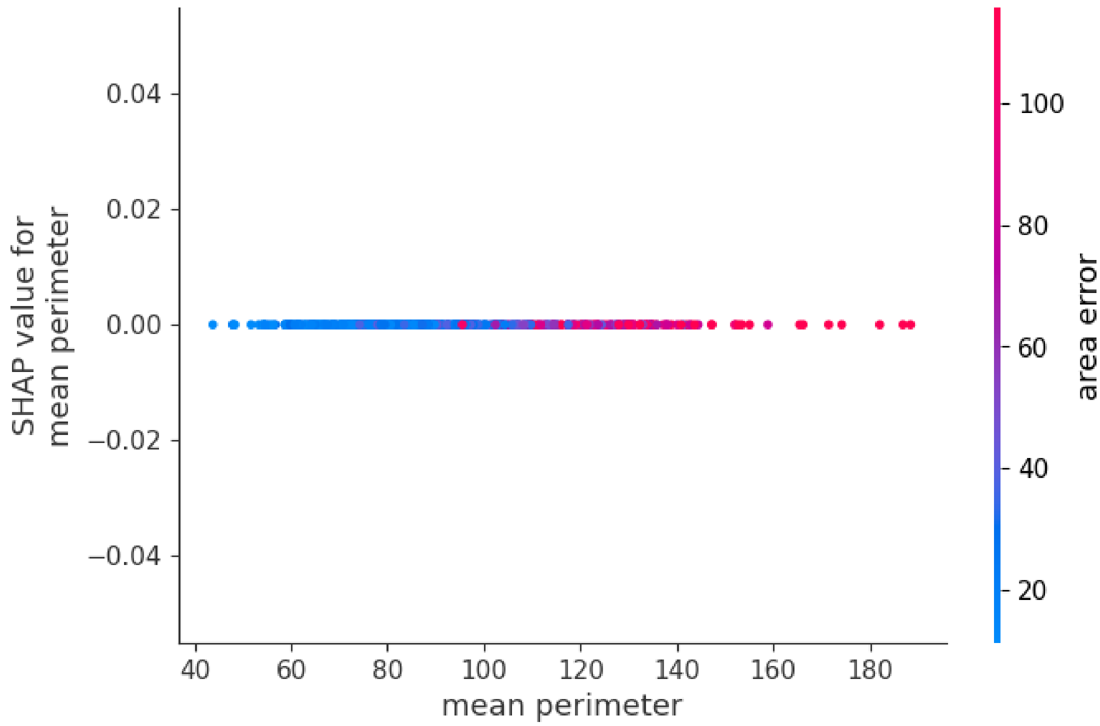
1
0
−1
−2
SHAP value for
mean concave points
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
mean concave points
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
worst texture



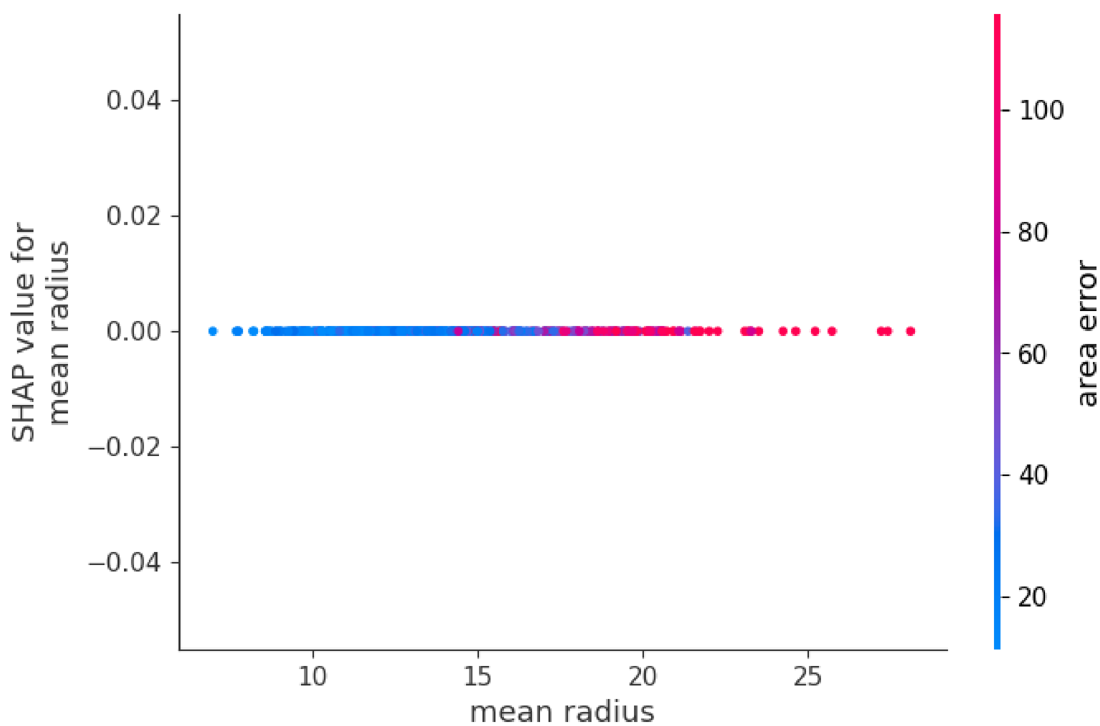
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
SHAP value for
mean concavity
0.0
0.1
0.2
0.3
0.4
mean concavity
26
24
22
20
18
16
14
mean texture



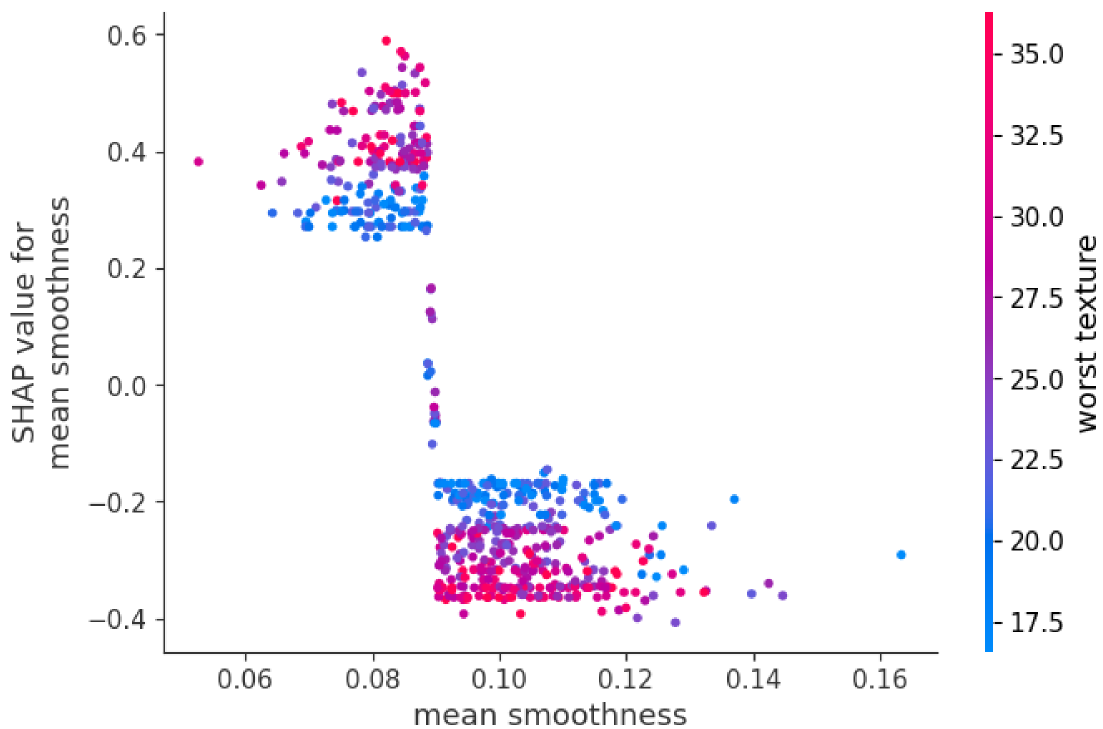
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
SHAP value for
mean fractal dimension
0.05
0.06
0.07
0.08
0.09
mean fractal dimension
0.115
0.110
0.105
0.100
0.095
0.090
0.085
0.080
mean smoothness



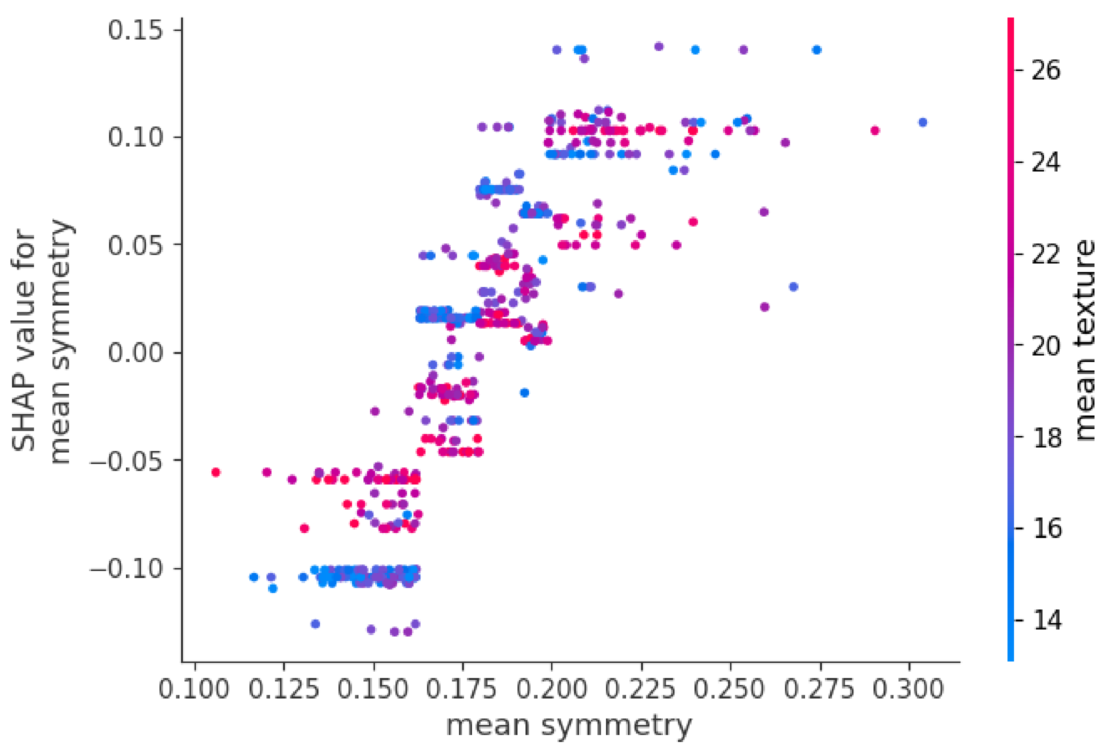
0.04
0.02
0.00
−0.02
−0.04
SHAP value for
mean perimeter
40
60
80
100
120
140
160
180
mean perimeter
100
80
60
40
20
area error



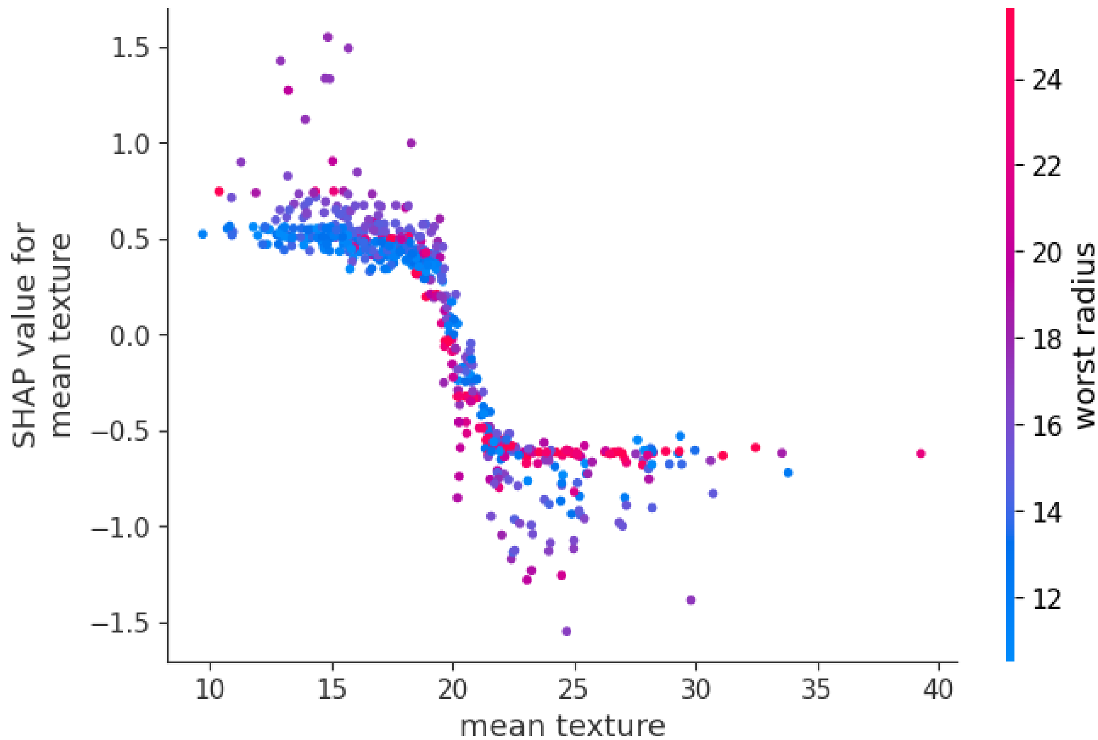
0.04
0.02
0.00
−0.02
−0.04
SHAP value for
mean radius
10
15
20
25
mean radius
100
80
60
40
20
area error

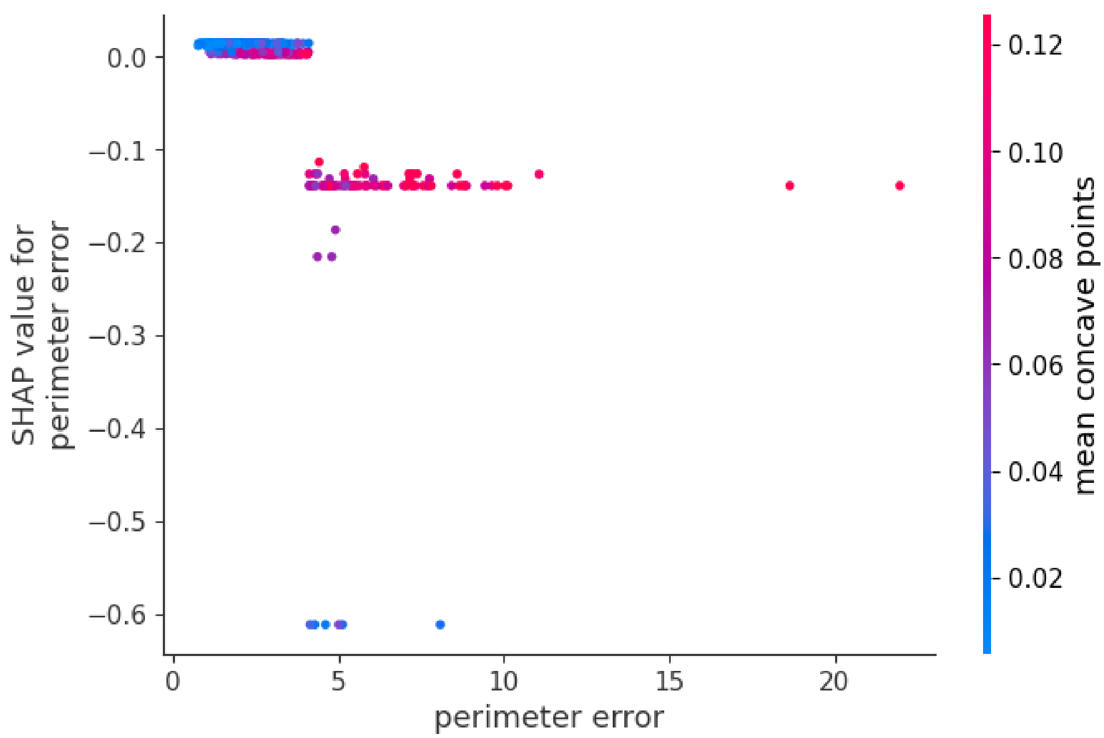


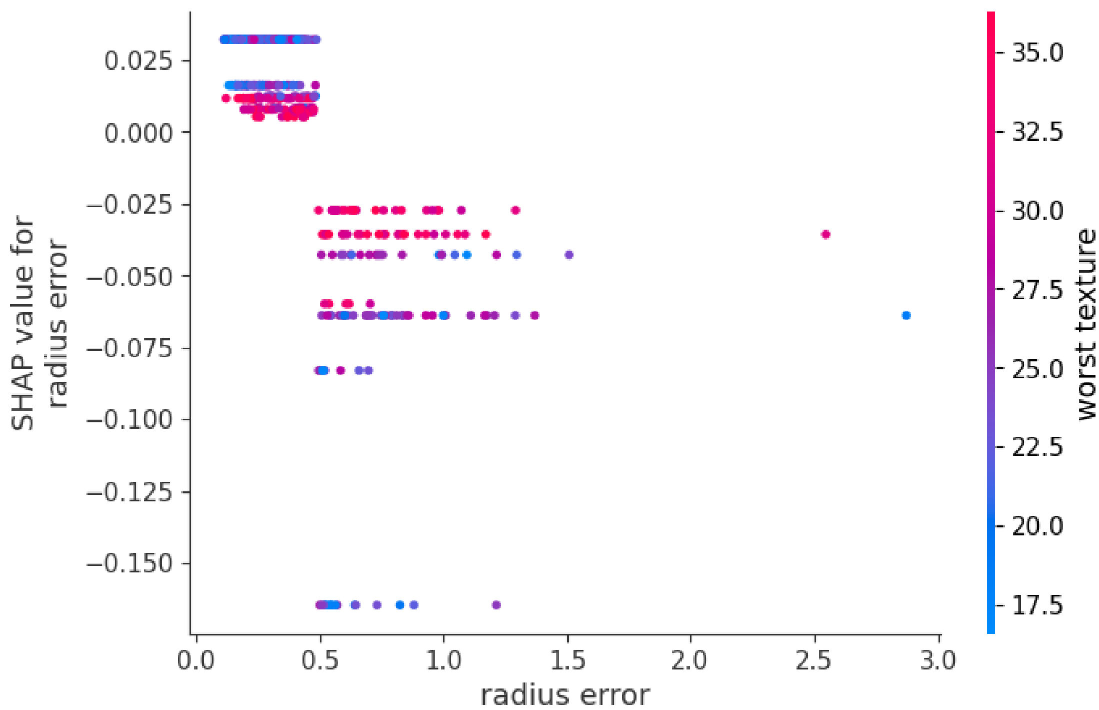
SHAP value for mean smoothness
mean smoothness
worst texture

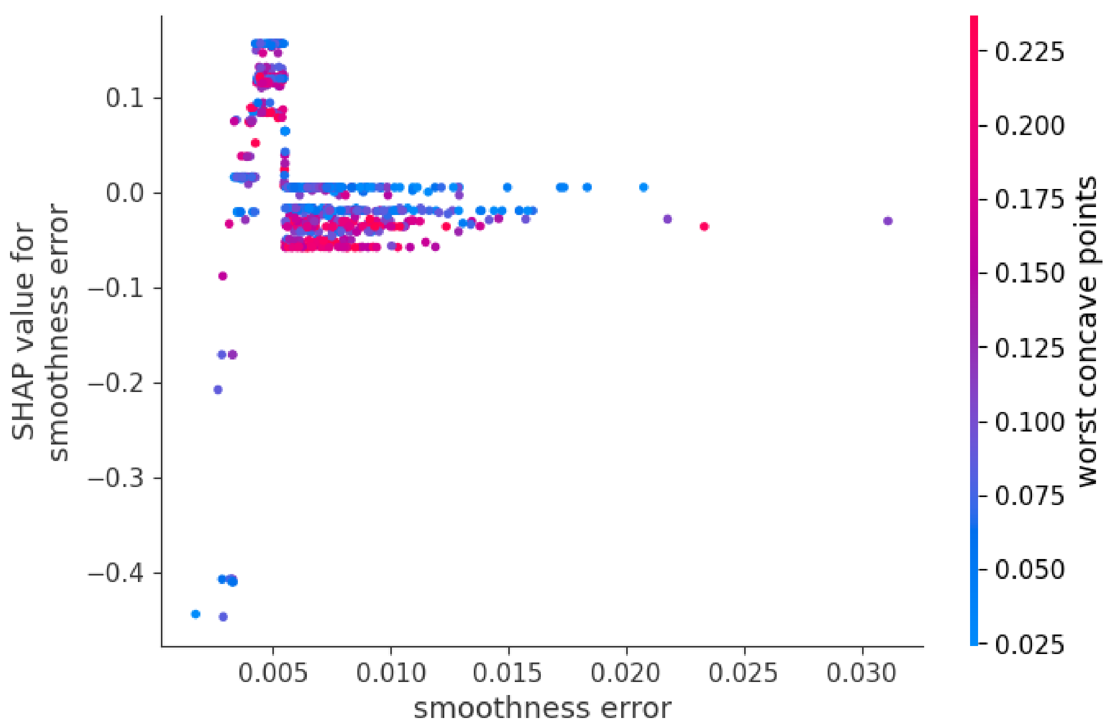


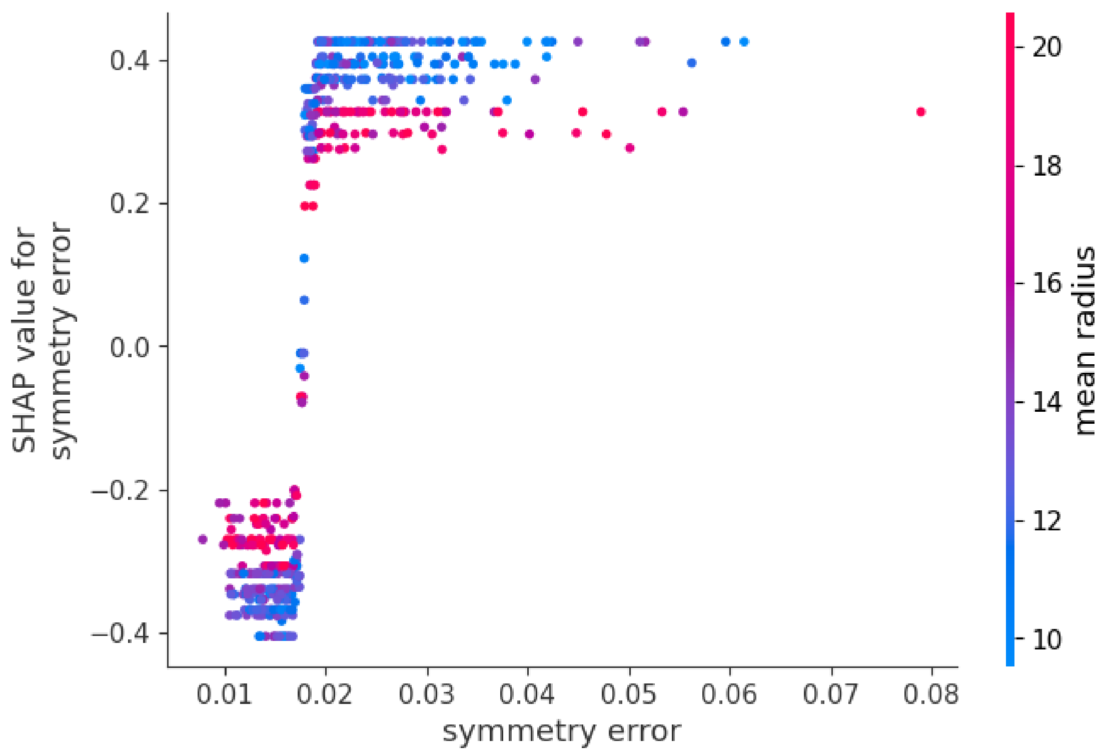
SHAP value for mean symmetry
mean symmetry
mean texture

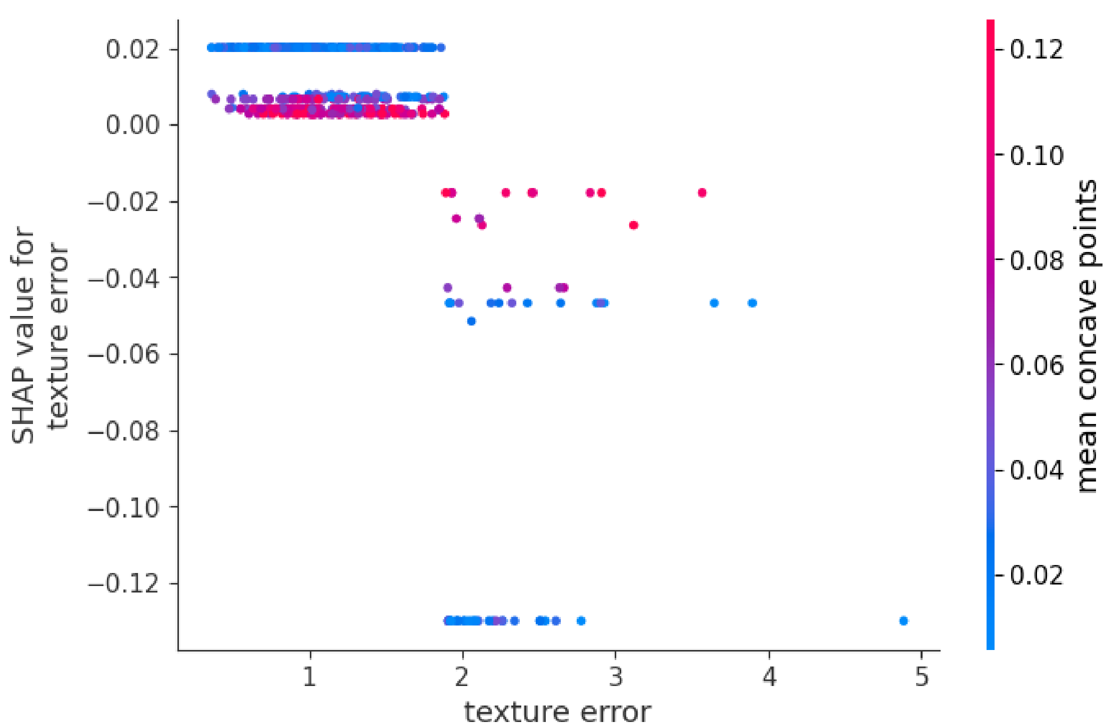

SHAP value for mean texture
mean texture
worst radius
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
10
15
20
25
30
35
40
24
22
20
18
16
14
12

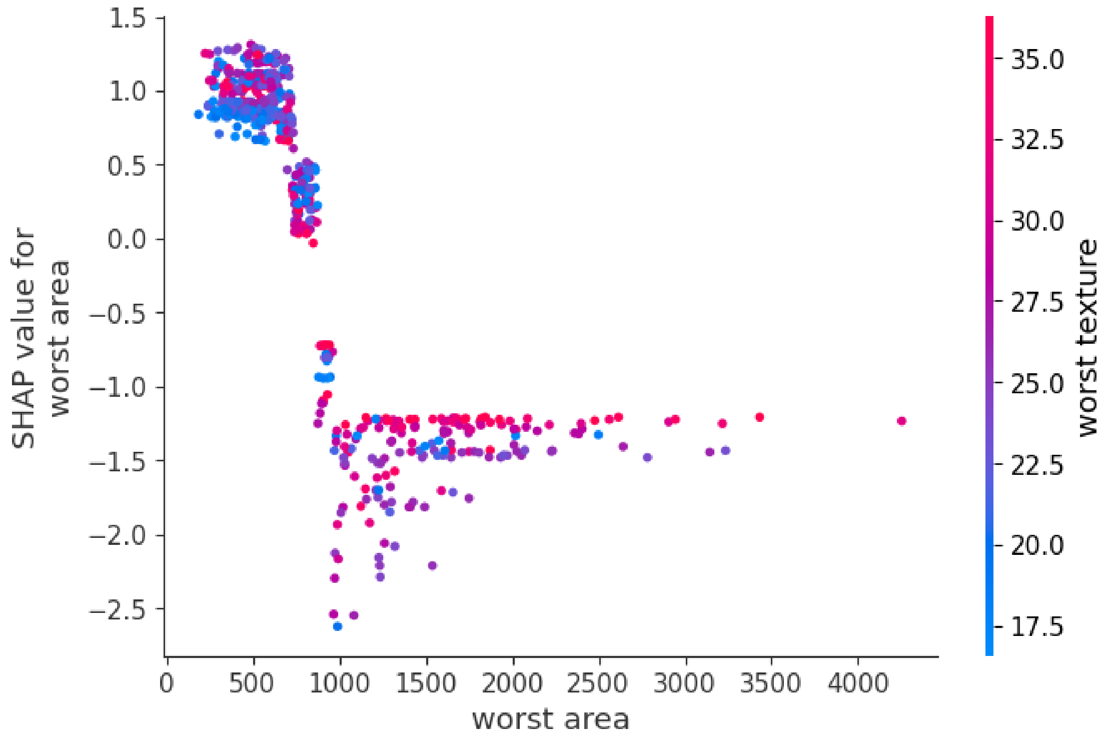

SHAP value for perimeter error
perimeter error
mean concave points
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0
5
10
15
20
0.12
0.10
0.08
0.06
0.04
0.02


SHAP value for
radius error
radius error
worst texture
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.125
−0.150
0.0
0.5
1.0
1.5
2.0
2.5
3.0
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5

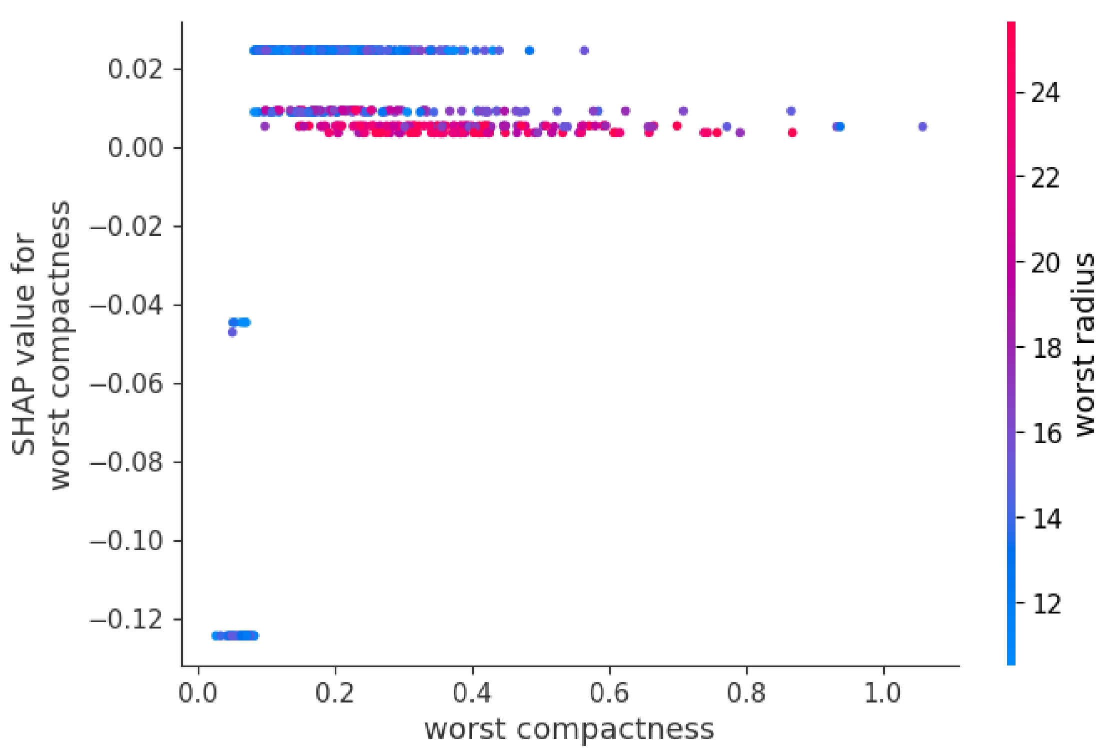

SHAP value for
smoothness error
smoothness error
worst concave points
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0.005
0.010
0.015
0.020
0.025
0.030
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025


SHAP value for symmetry error
symmetry error
mean radius
0.4
0.2
0.0
−0.2
−0.4
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
20
18
16
14
12
10

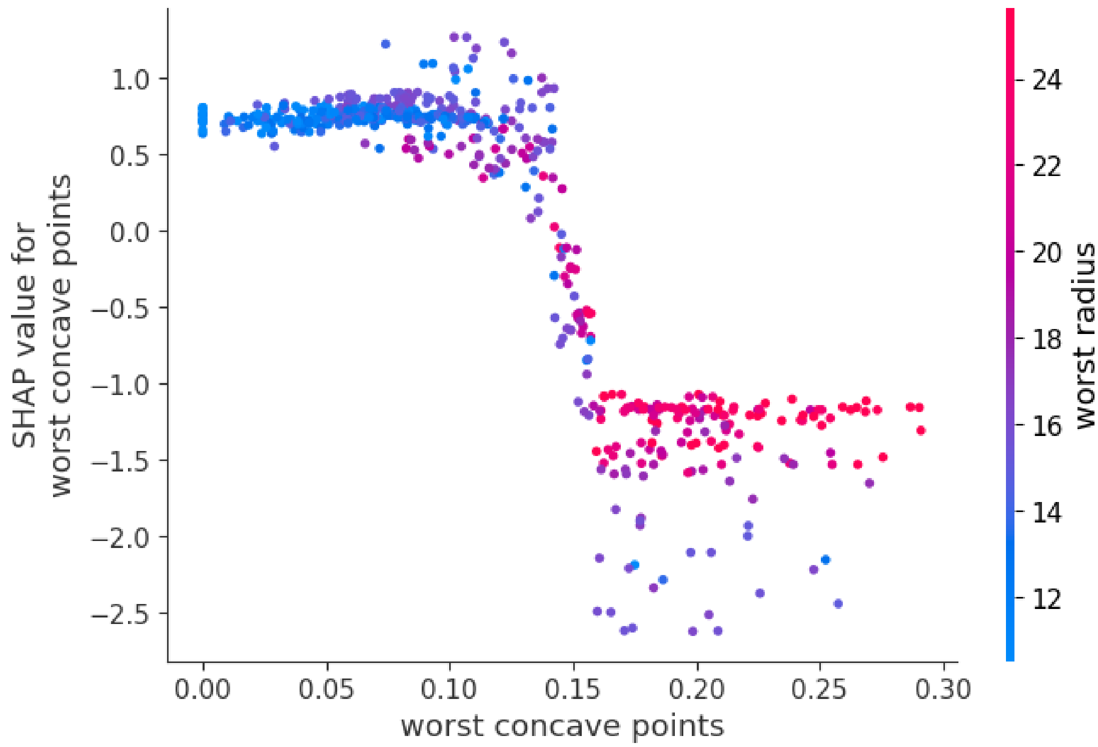

SHAP value for texture error
texture error
mean concave points
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
1
2
3
4
5
0.12
0.10
0.08
0.06
0.04
0.02

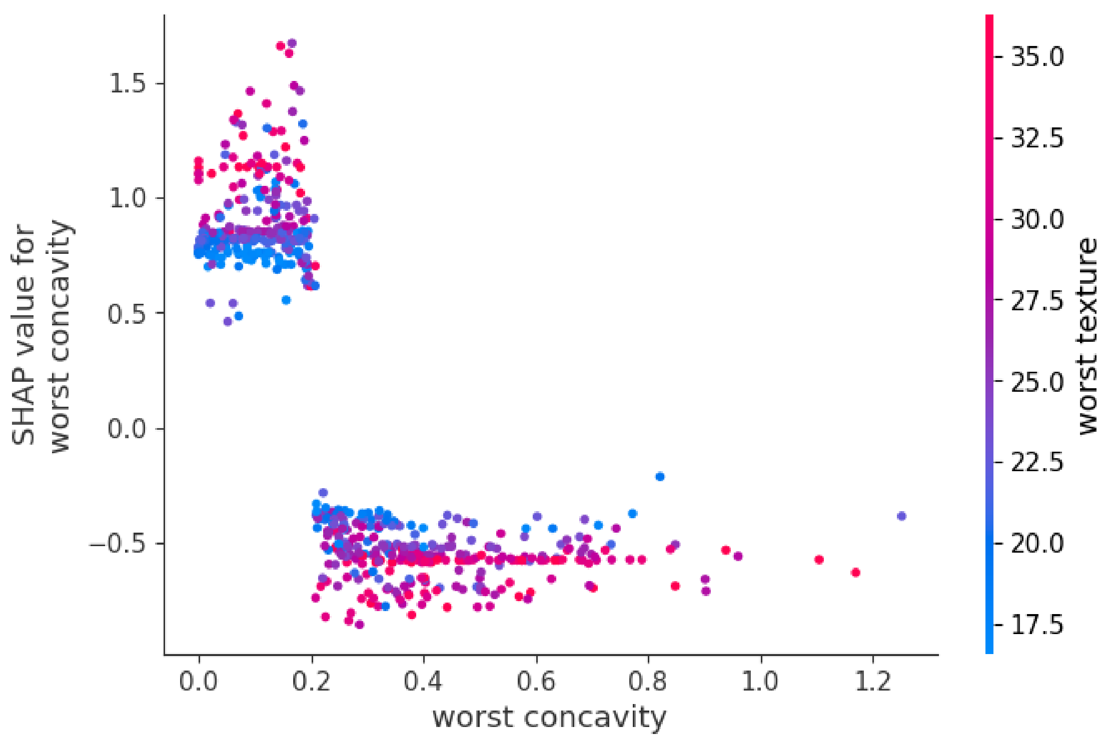


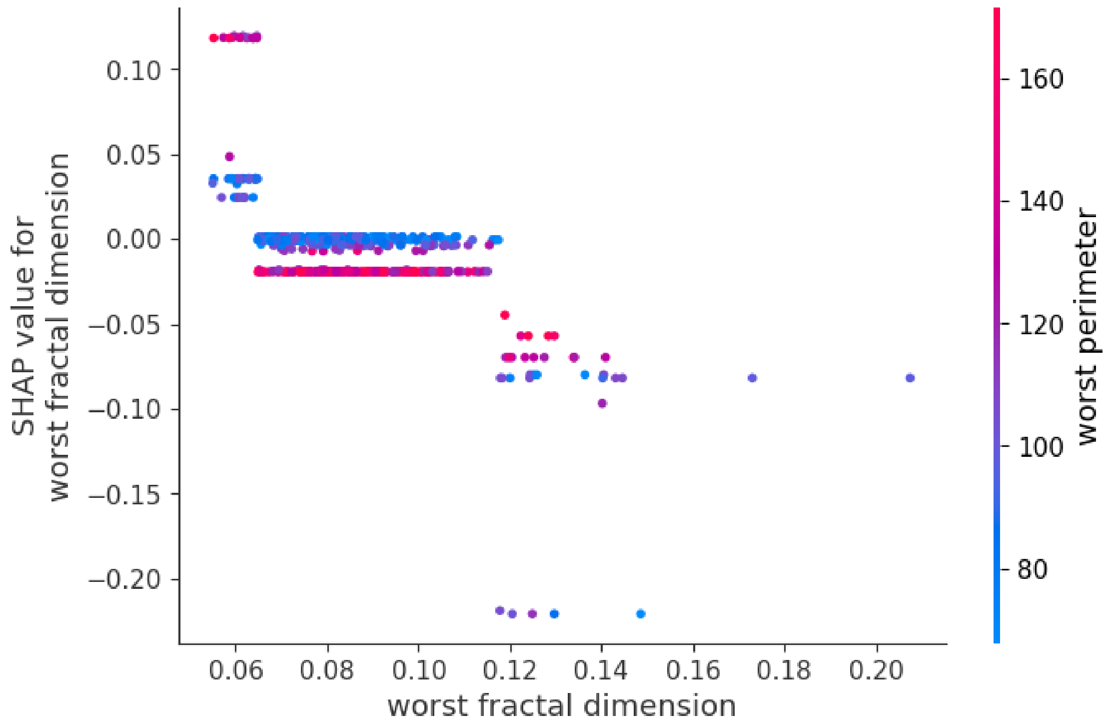
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
SHAP value for
worst area
0
500
1000
1500
2000
2500
3000
3500
4000
worst area
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
worst texture

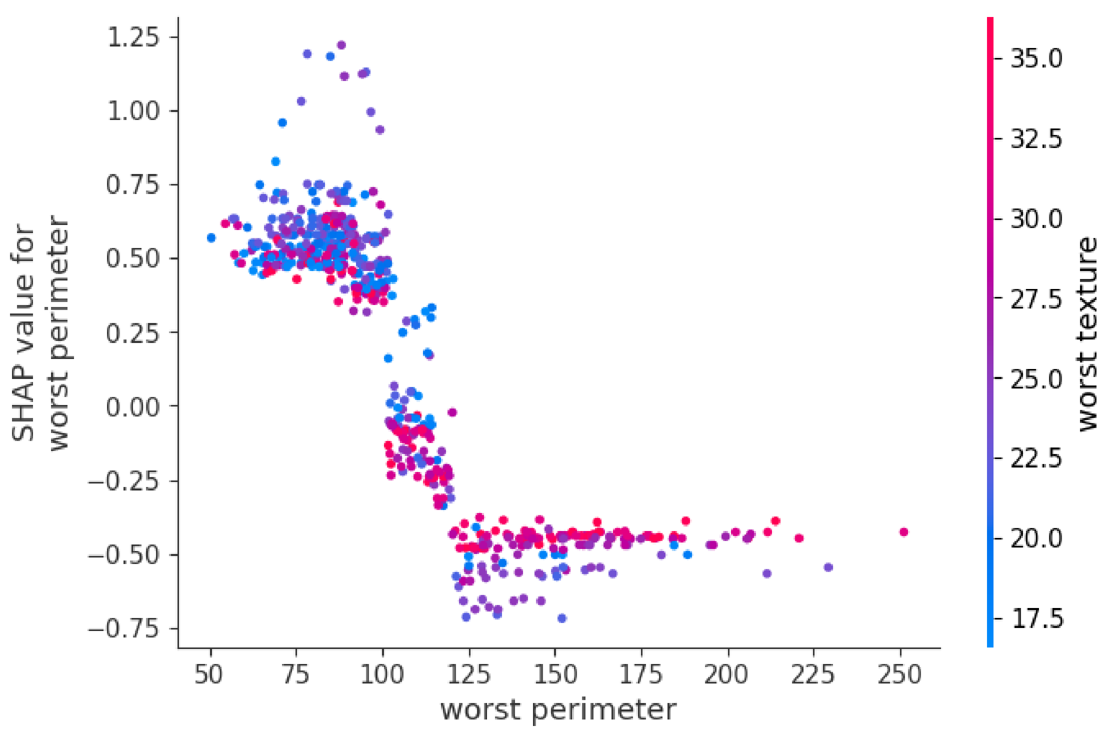


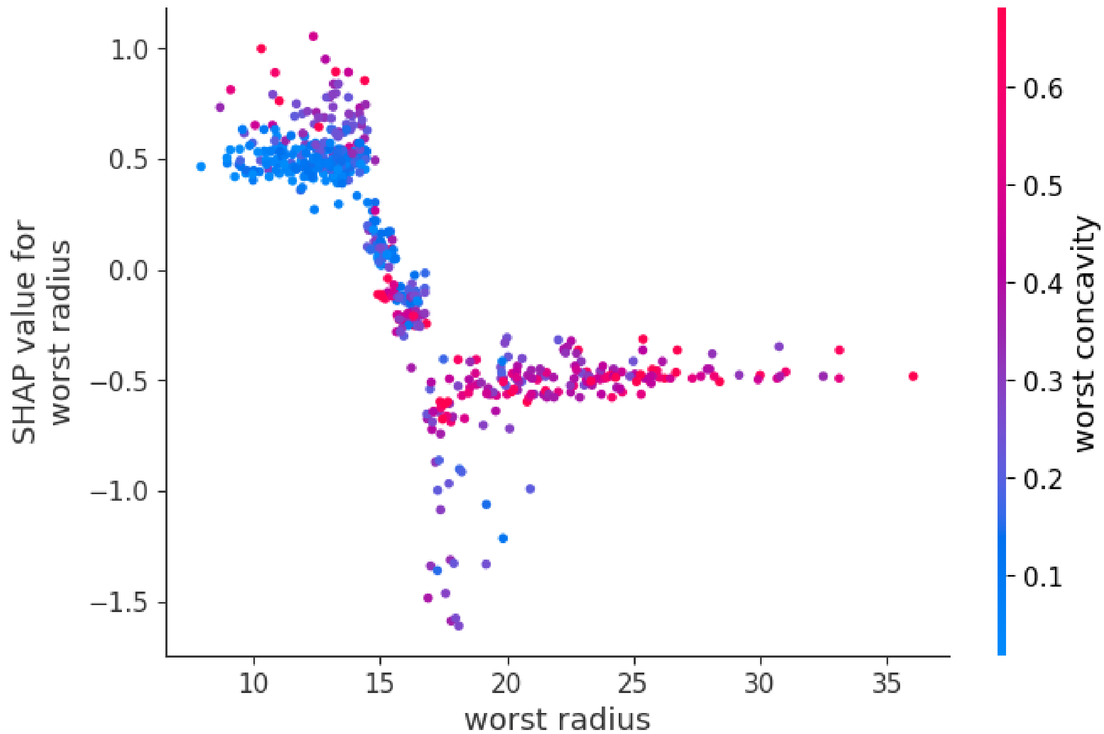
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
SHAP value for
worst compactness
0.0
0.2
0.4
0.6
0.8
1.0
worst compactness
24
22
20
18
16
14
12
worst radius

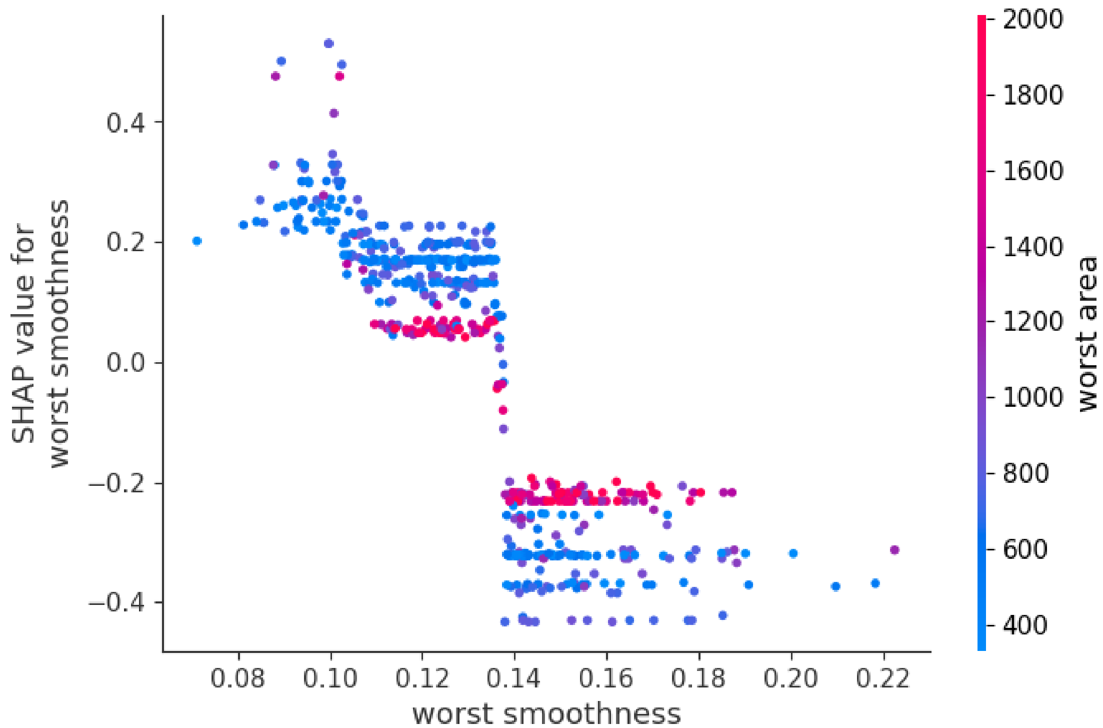

SHAP value for
worst concave points
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30
worst concave points
24
22
20
18
16
14
12
worst radius

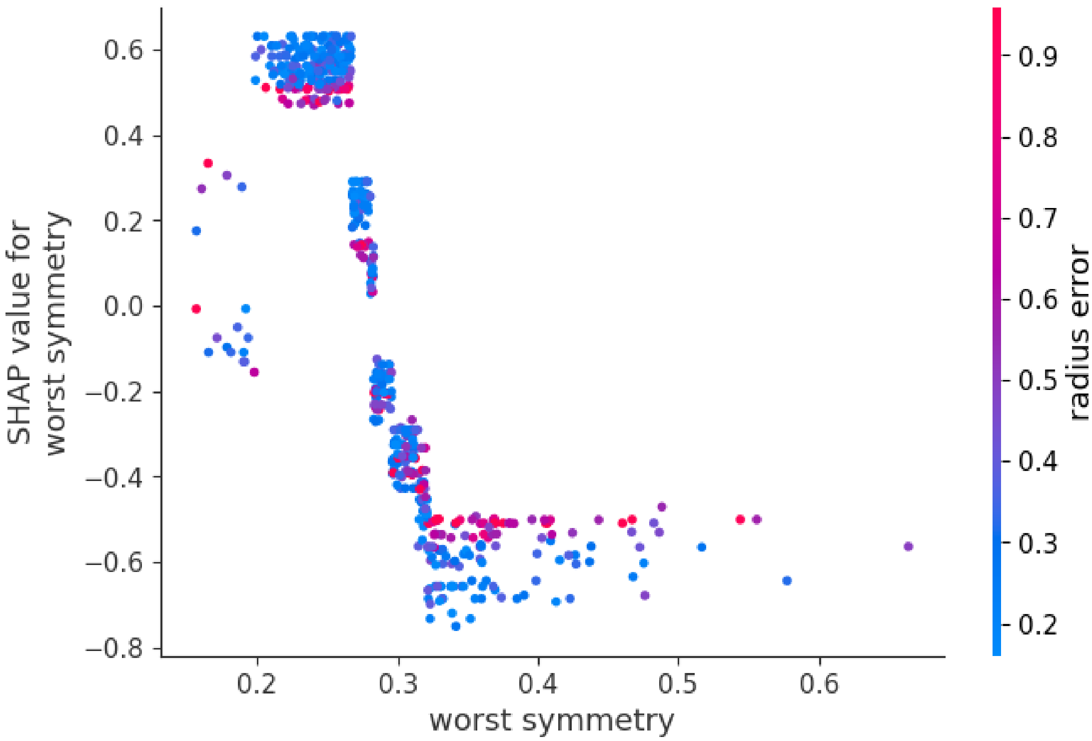

SHAP value for
worst concavity
1.5
1.0
0.5
0.0
−0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
worst concavity
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
worst texture

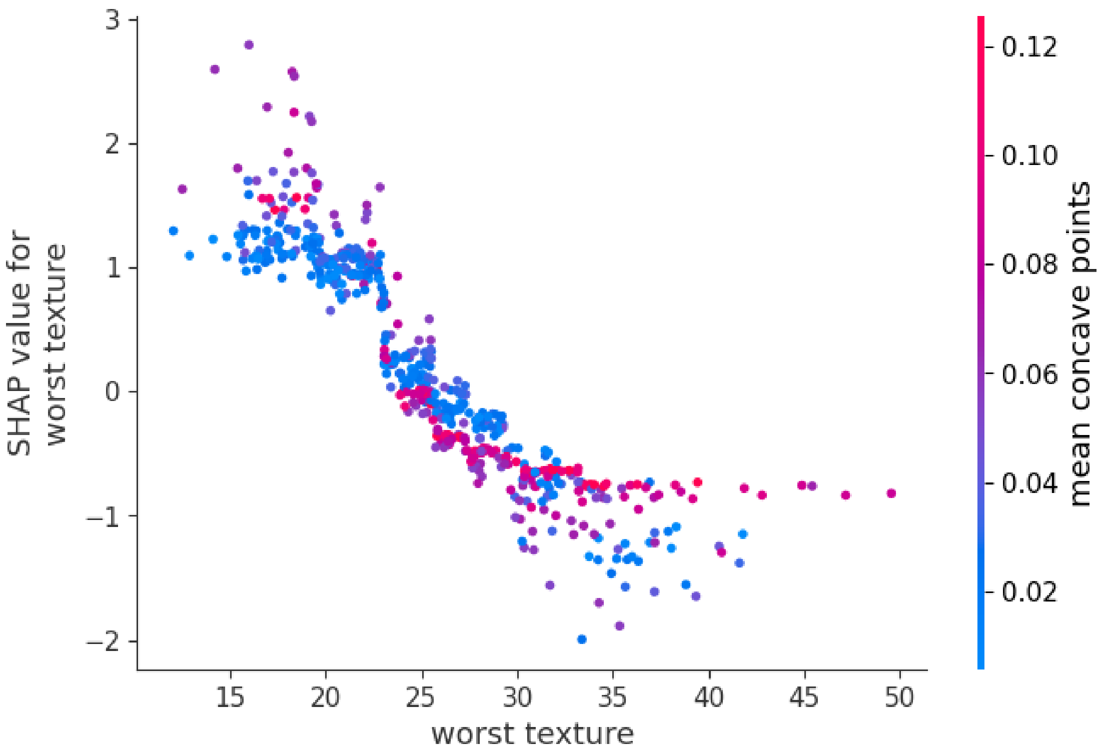

SHAP value for
worst fractal dimension
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20
worst fractal dimension
160
140
120
100
80
worst perimeter

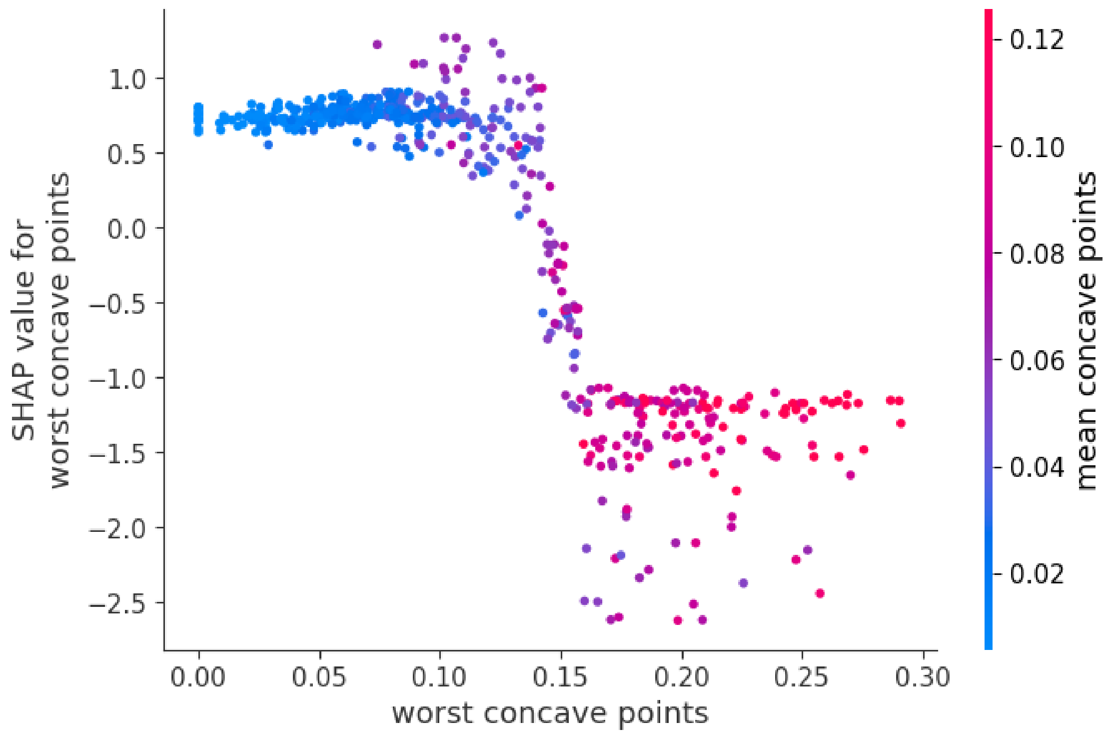

SHAP value for
worst perimeter
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
50
75
100
125
150
175
200
225
250
worst perimeter
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
worst texture


SHAP value for
worst radius
worst radius
worst concavity
1.0
0.5
0.0
−0.5
−1.0
−1.5
10
15
20
25
30
35
0.6
0.5
0.4
0.3
0.2
0.1


SHAP value for
worst smoothness
worst smoothness
worst area
0.4
0.2
0.0
−0.2
−0.4
0.08
0.10
0.12
0.14
0.16
0.18
0.20
0.22
2000
1800
1600
1400
1200
1000
800
600
400


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
SHAP value for
worst symmetry
0.2
0.3
0.4
0.5
0.6
worst symmetry
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
radius error


3
2
1
0
−1
−2
SHAP value for
worst texture
15
20
25
30
35
40
45
50
worst texture
0.12
0.10
0.08
0.06
0.04
0.02
mean concave points



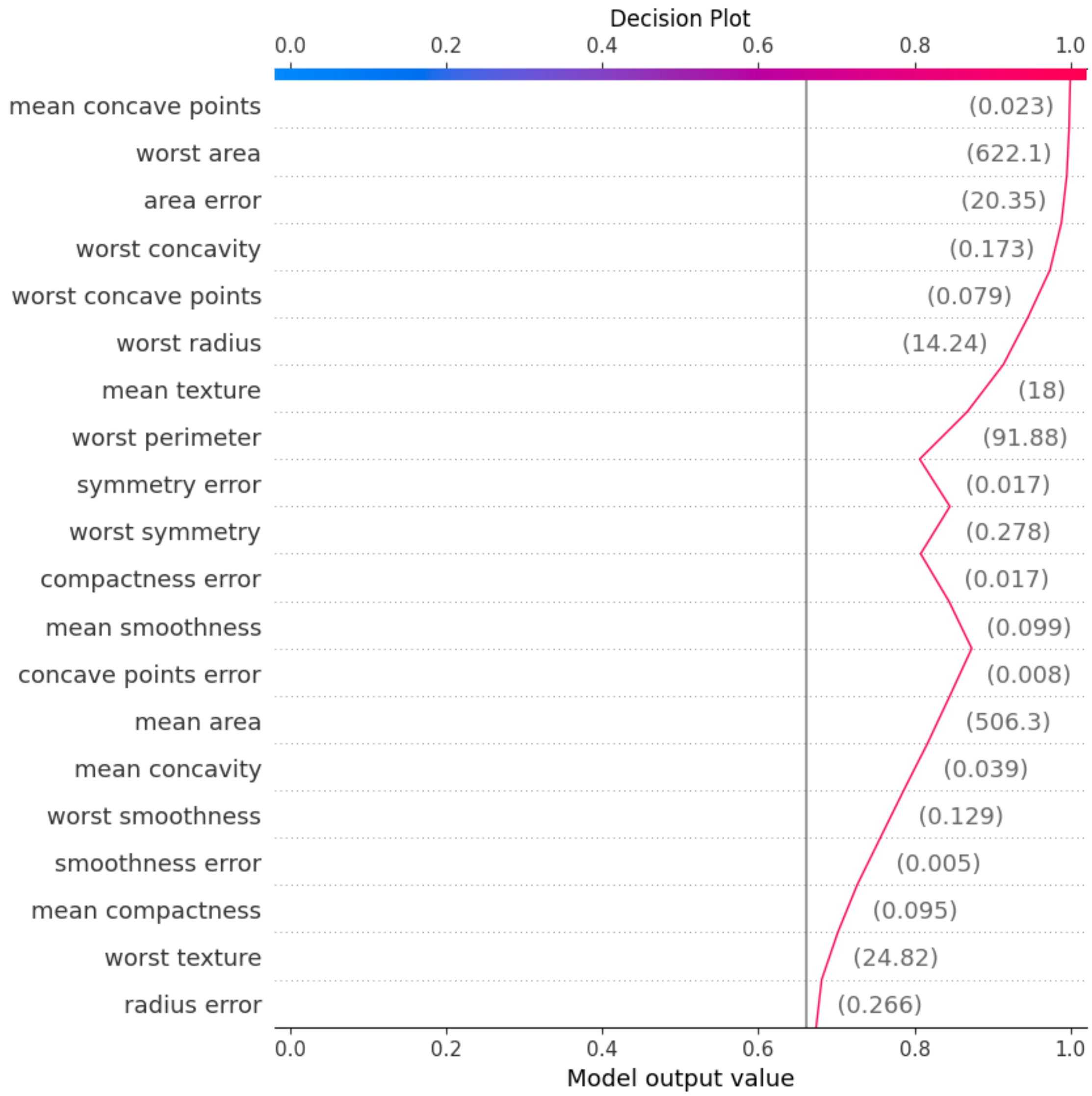
3.7 SHAP - Force plots (não é necessário fazer esses)

Adicionar aspas

[ ]:
```
# Generate force plot - Multiple rows
shap.force_plot(explainer.expected_value, shap_values[:100,:], X.iloc[:100,:])
```

[ ]:
```
# Generate force plot - Single
shap.force_plot(explainer.expected_value, shap_values[0,:], X.iloc[0,:])
```

[ ]:
```
# Generate Decision plot
shap.decision_plot(expected_value, shap_values[79],
                   link='logit' ,features=X.loc[79,:],
                   feature_names=(X.columns.tolist()),
                   show=True,title="Decision Plot")
```



#Push the limits of explainability — an ultimate guide to SHAP library

https://medium.com/swlh/push-the-limits-of-explainability-an-ultimate-guide-to-shap-library-a110af566a02

```
[ ]:
```