SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Bhuvi Ghosh
60009210191

COURSE CODE: DJ19DSC501

COURSE NAME: Machine Learning - II **DATE: 4/12/23**

LAB EXPERIMENT NO.12

AIM :

Build Explainable AI to improve human decision making using a two-choice classification experiment with real world data.

THEORY:

Explainable artificial intelligence (XAI) is a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms. Explainable AI is used to describe an AI model, its expected impact and potential biases. It helps characterize model accuracy, fairness, transparency and outcomes in AI-powered decision making. Explainable AI is crucial for an organization in building trust and confidence when putting AI models into production. AI explanability also helps an organization adopt a responsible approach to AI development.

As AI becomes more advanced, humans are challenged to comprehend and retrace how the algorithm came to a result. The whole calculation process is turned into what is commonly referred to as a “black box" that is impossible to interpret. These black box models are created directly from the data. And, not even the engineers or data scientists who create the algorithm can understand or explain what exactly is happening inside them or how the AI algorithm arrived at a specific result.

There are many advantages to understanding how an AI-enabled system has led to a specific output. Explanability can help developers ensure that the system is working as expected, it might be necessary to meet regulatory standards, or it might be important in allowing those affected by a decision to challenge or change that outcome.

Industry Need of XAI.

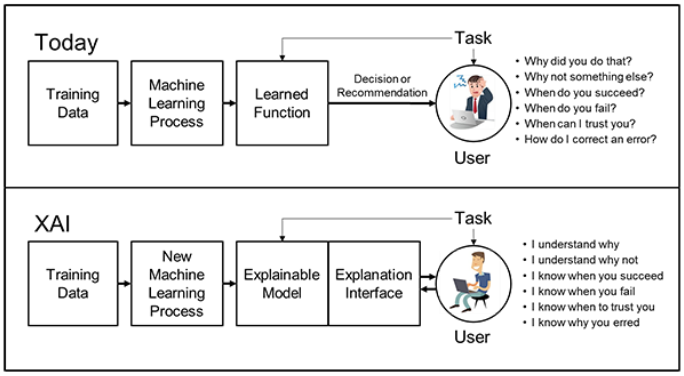
It is crucial for an organization to have a full understanding of the AI decision-making processes with model monitoring and accountability of AI and not to trust them blindly. Explainable AI can

help humans understand and explain machine learning (ML) algorithms, deep learning and neural networks.

ML models are often thought of as black boxes that are impossible to interpret. Neural networks used in deep learning are some of the hardest for a human to understand. Bias, often based on race, gender, age or location, has been a long-standing risk in training AI models. Further, AI model performance can drift or degrade because production data differs from training data. This makes it crucial for a business to continuously monitor and manage models to promote AI explanability while measuring the business impact of using such algorithms. Explainable AI also helps promote end user trust, model auditability and productive use of AI. It also mitigates compliance, legal, security and reputational risks of production AI.

Explainable AI is one of the key requirements for implementing responsible AI, a methodology for the large-scale implementation of AI methods in real organizations with fairness, model explanability and accountability. To help adopt AI responsibly, organizations need to embed ethical principles into AI applications and processes by building AI systems based on trust and transparency.

With explainable AI, a business can troubleshoot and improve model performance while helping stakeholders understand the behaviors of AI models. Investigating model behaviors through tracking model insights on deployment status, fairness, quality and drift is essential to scaling AI. Continuous model evaluation empowers a business to compare model predictions, quantify model risk and optimize model performance. Displaying positive and negative values in model behaviors with data used to generate explanation speeds model evaluations. A data and AI platform can generate feature attributions for model predictions and empower teams to visually investigate model behavior with interactive charts and exportable documents.



SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Tasks to be performed:

1. **Take any appropriate dataset [Breast_Cancer Dataset]**
1. **Perform any 3 XAI methods on this dataset [SHAP, LIME, SHAPASH]**
2. **Describe the model and its results [summaries, visualizations, numerical descriptions].**
3. **At the end of each implementation, explain how the use of that particular XAI technique helped in making the model more explainable.**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
import xgboost as xgb
from sklearn.metrics import accuracy_score
import shap

# 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

# 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



# build model - Xgboost

xgb_mod=xgb.XGBClassifier(random_state=42,gpu_id=0) # build classifier Gradient Boosted decision trees

xgb_mod=xgb_mod.fit(X_train,y_train.values.ravel())
```

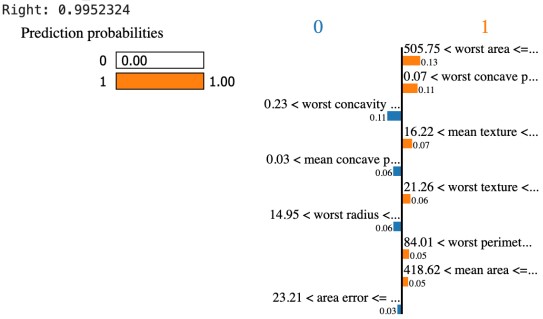
```
[3] # make prediction and check model accuracy

y_pred = xgb_mod.predict(X_test)

# Performance

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 95.61%
[05:33:08] WARNING: /workspace/src/common/error_msg.cc:45: `gpu_id` is deprecated since2.0.0, use `device` instead. E.g. device=cpu/cuda/cuda:0
[05:33:08] WARNING: /workspace/src/context.cc:44: No visible GPU is found, setting device to CPU.
```

```
#LIME

# Utilizing our same xgb_mod model object created above

# Import pacakages

import lime

import lime.lime_tabular

import numpy as np

import xgboost
```

```
explainer = lime.lime_tabular.LimeTabularExplainer(X_test.to_numpy(), feature_names=X_test.columns, class_names=['0','1'], verbose=True)


############# visualizations ############

exp = explainer.explain_instance(X_test.iloc[0], xgb_mod.predict_proba)#, num_features=20)

exp.show_in_notebook(show_table=True)
```

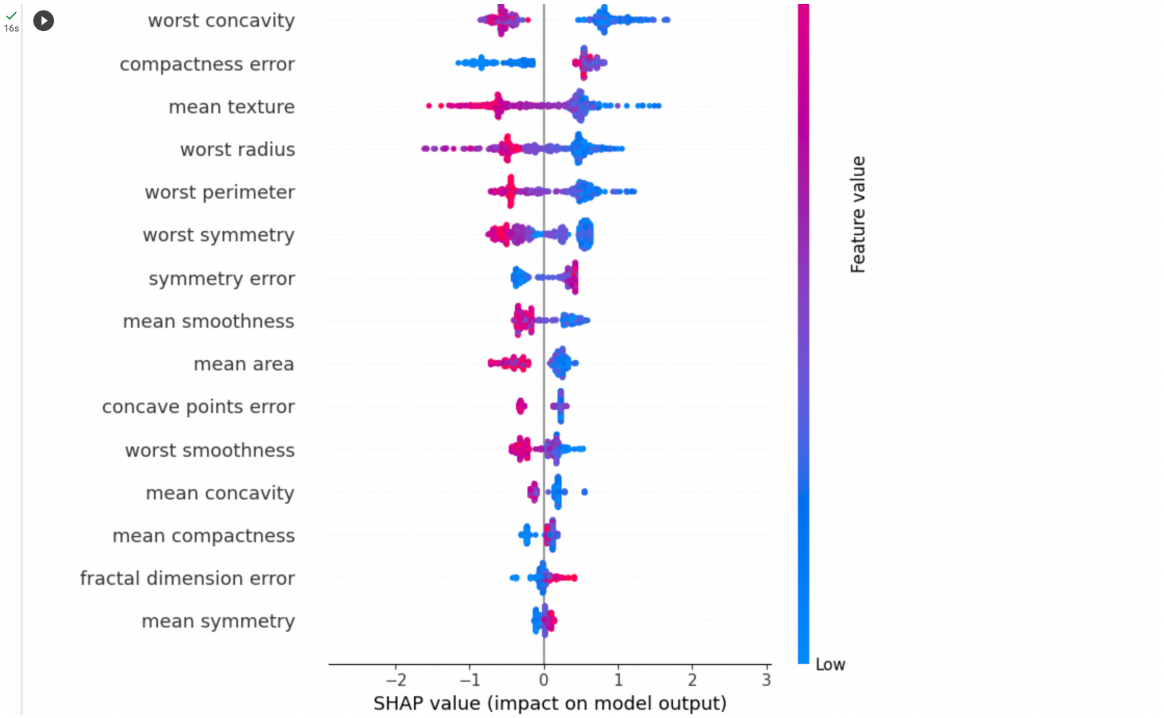
```
Intercept 0.5535166624162877
Prediction_local [0.77283178]
Right: 0.9952324
```

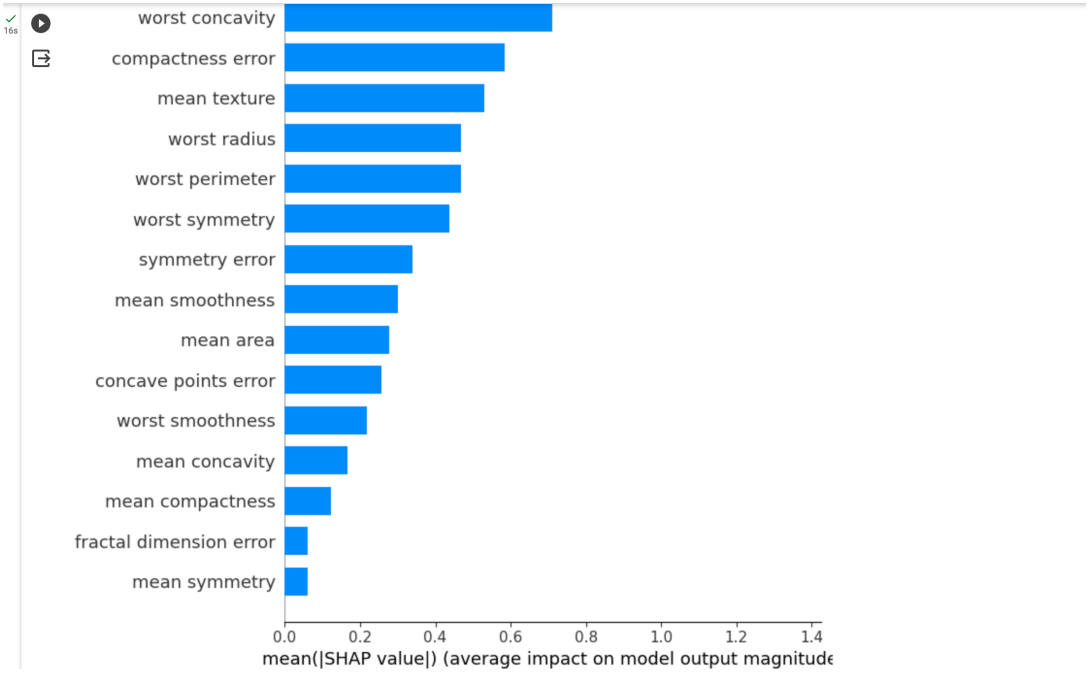


Feature	Value
worst area	677.90
worst concave points	0.10
worst concavity	0.27
mean texture	18.60
mean concave points	0.04
worst texture	24.64
worst radius	14.97
worst perimeter	96.05
mean area	481.90
area error	30.29

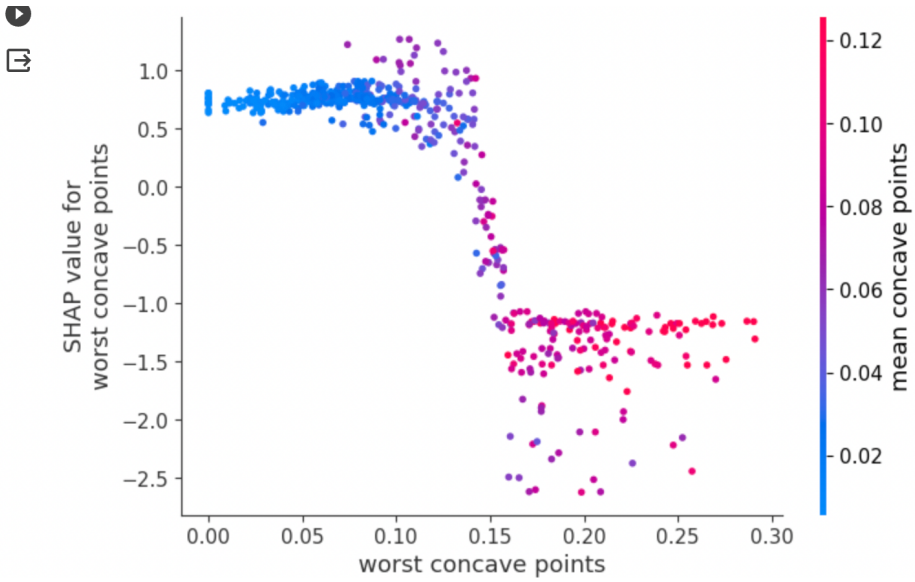
```
#SHAP

# Generate the Tree explainer and SHAP values

explainer = shap.TreeExplainer(xgb_mod) #fast and exact method to estimate SHAP values for tree models and ensembles of trees,

shap_values = explainer.shap_values(X)

expected_value = explainer.expected_value

############# visualizations ############

# Generate summary dot plot

shap.summary_plot(shap_values, X,title="SHAP summary plot")

# Generate summary bar plot

shap.summary_plot(shap_values, X,plot_type="bar")

# Generate waterfall plot

shap.plots._waterfall.waterfall_legacy(expected_value, shap_values[79], features=X.loc[79,:], feature_names=X.columns, max_display=15, show=True)

# 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")
```

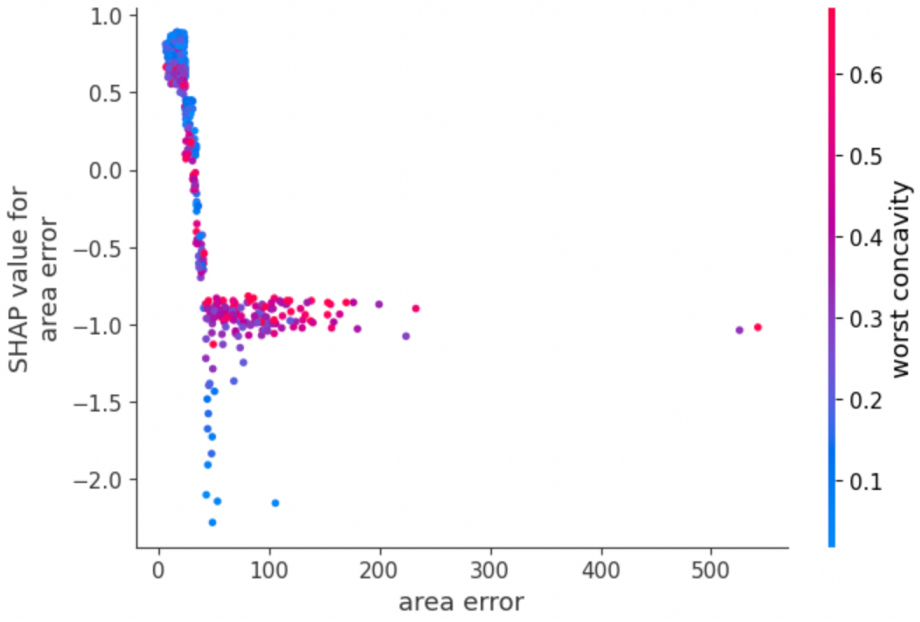
```
# Generate force plot - Multiple rows

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")
```

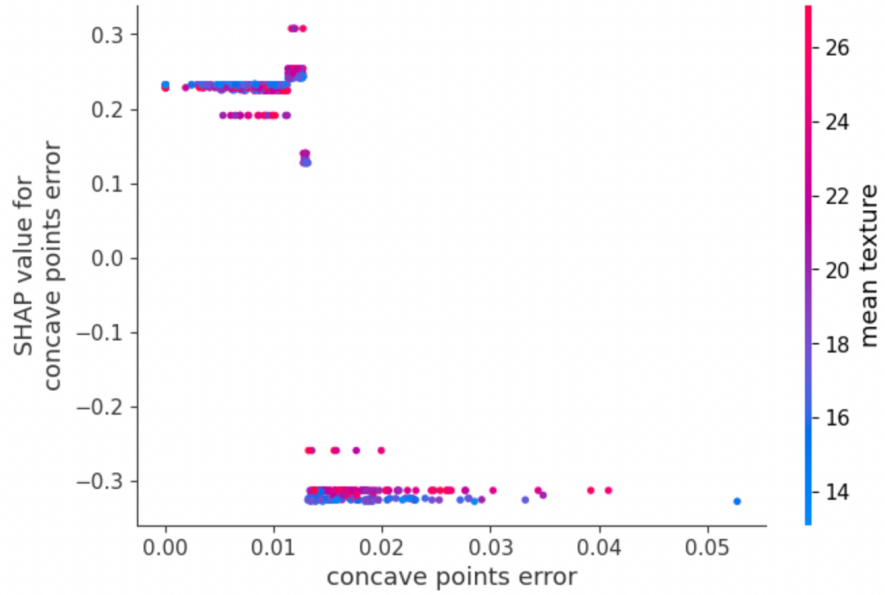

worst concavity
compactness error
mean texture
worst radius
worst perimeter
worst symmetry
symmetry error
mean smoothness
mean area
concave points error
worst smoothness
mean concavity
mean compactness
fractal dimension error
mean symmetry
−2
−1
0
1
2
3
SHAP value (impact on model output)
Feature value
Low

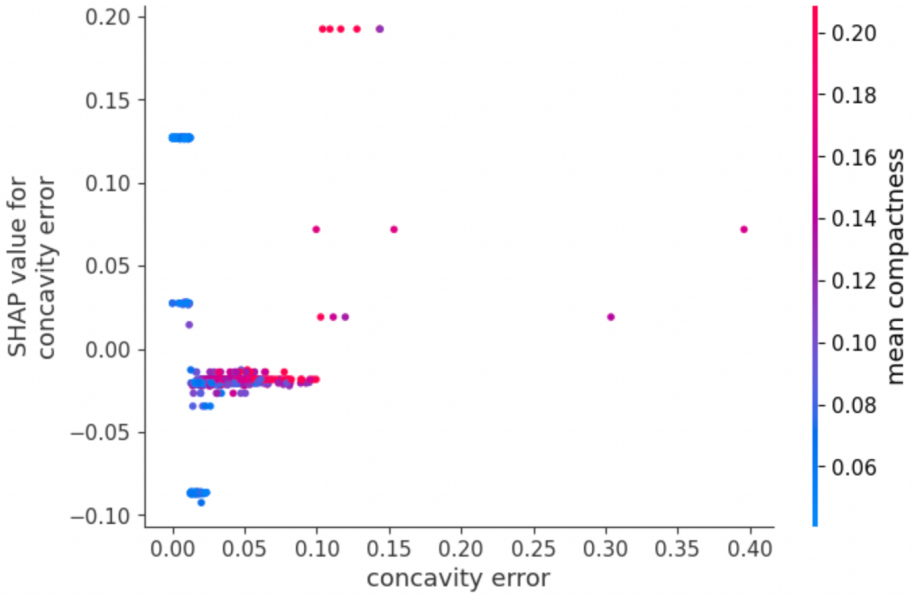

worst concavity
compactness error
mean texture
worst radius
worst perimeter
worst symmetry
symmetry error
mean smoothness
mean area
concave points error
worst smoothness
mean concavity
mean compactness
fractal dimension error
mean symmetry
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
mean(|SHAP value|) (average impact on model output magnitude

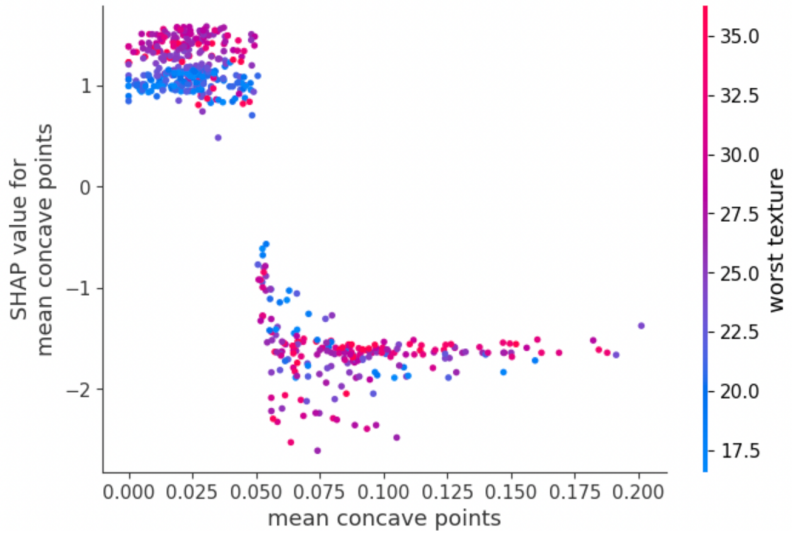

worst area
area error
worst concavity
worst concave points
worst radius
mean texture
worst perimeter
symmetry error
worst symmetry
compactness error
mean smoothness
concave points error
mean area
16 other features
+0.93
+0.88
+0.82
+0.76
+0.49
+0.48
+0.45
−0.27
+0.26
−0.25
−0.24
+0.23
+0.2
+0.83
1
2
3
4
5
6
7
E[f(X)] = 0.668

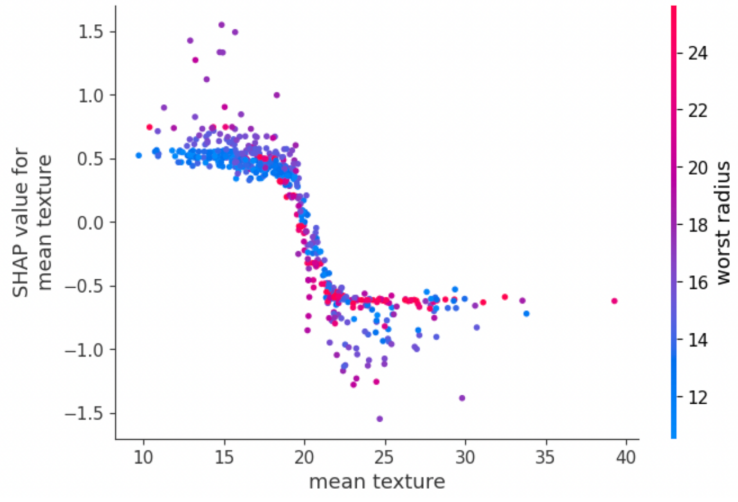

SHAP value for
worst concave points
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30
worst concave points
0.12
0.10
0.08
0.06
0.04
0.02
mean concave points

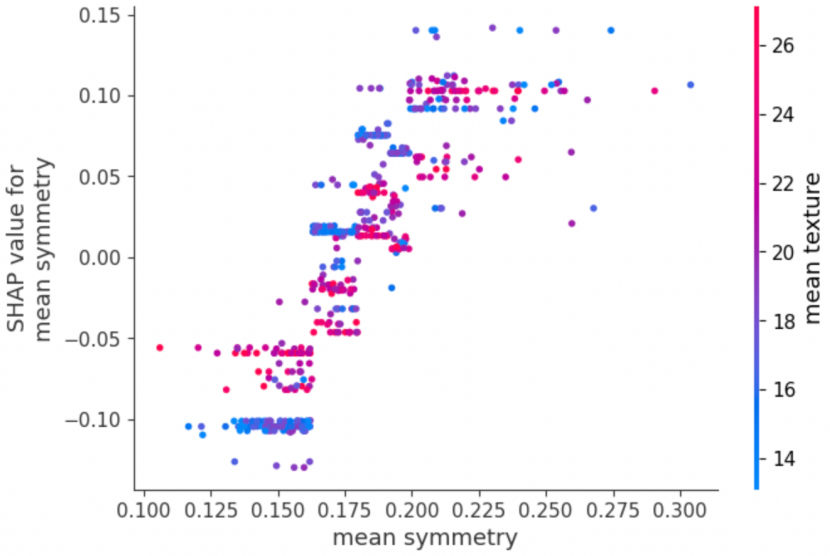

SHAP value for
area error
area error
worst concavity

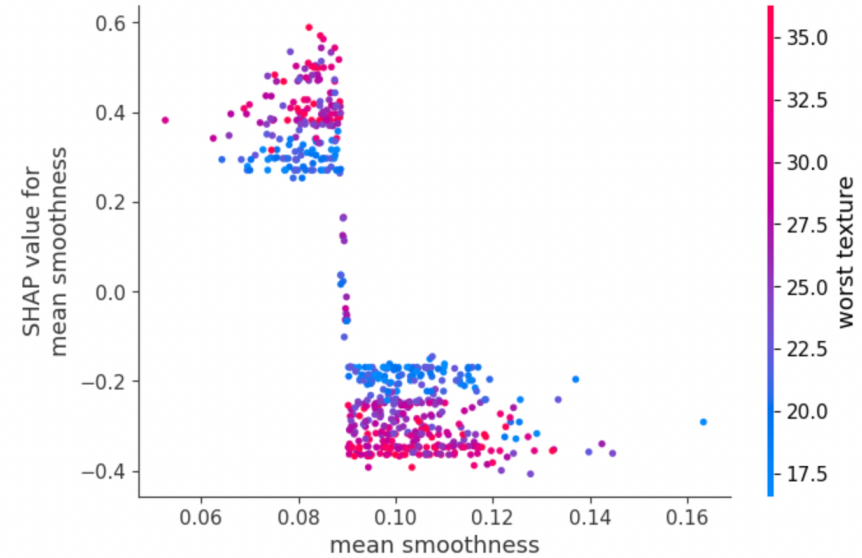

SHAP value for
concave points error
concave points error
mean texture

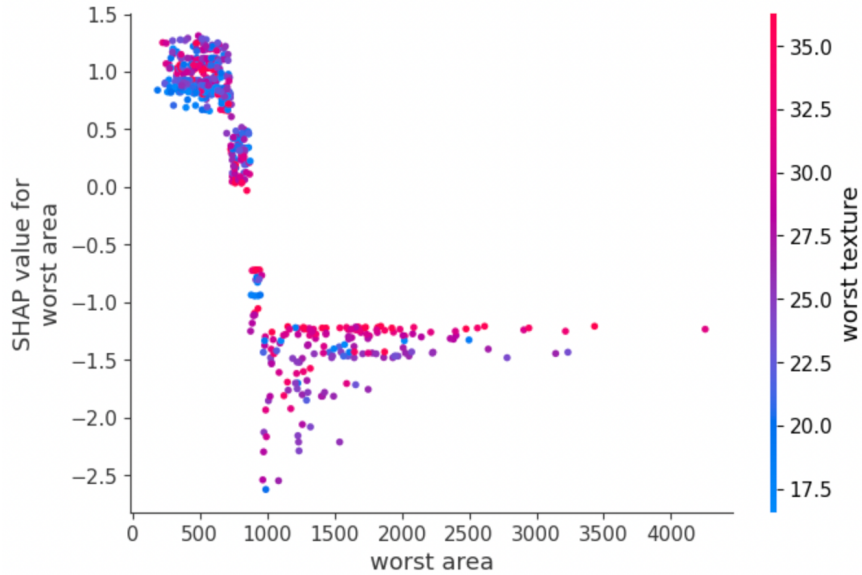

SHAP value for
concavity error
concavity error
mean compactness
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06

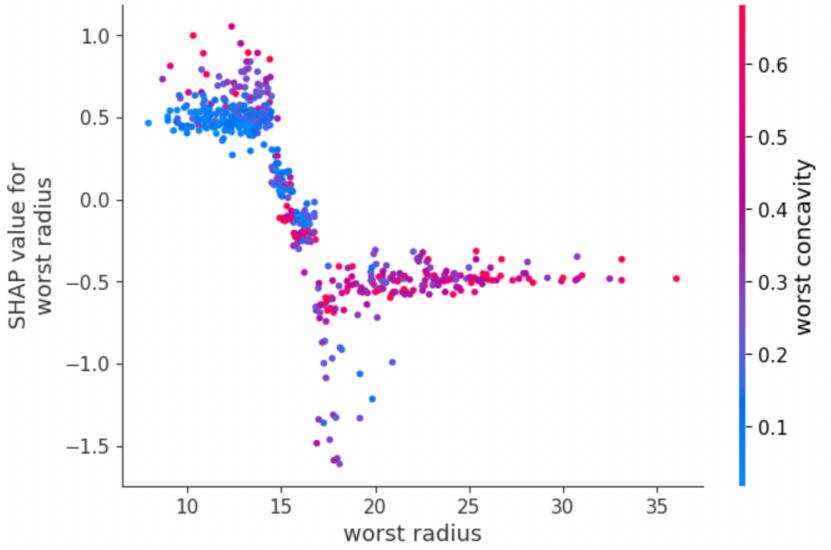

SHAP value for
mean concave points
mean concave points
worst texture
1
0
−1
−2
0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5

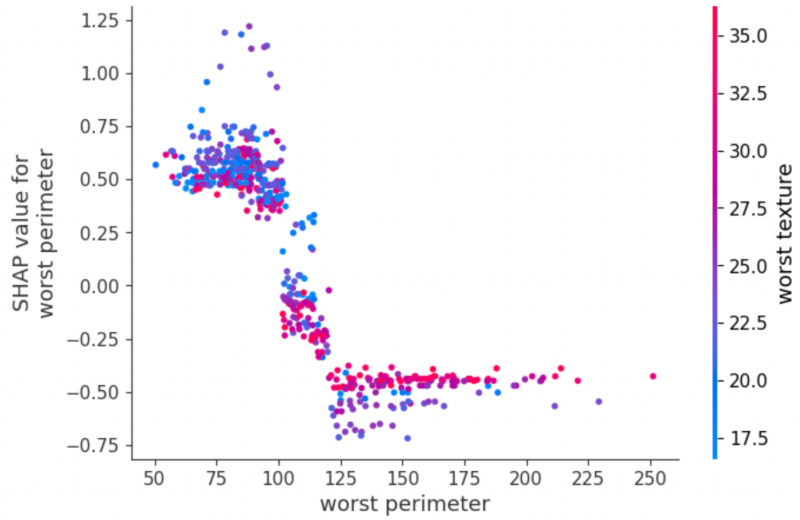

SHAP value for
mean texture
mean texture
worst radius
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
10 15 20 25 30 35 40
24
22
20
18
16
14
12

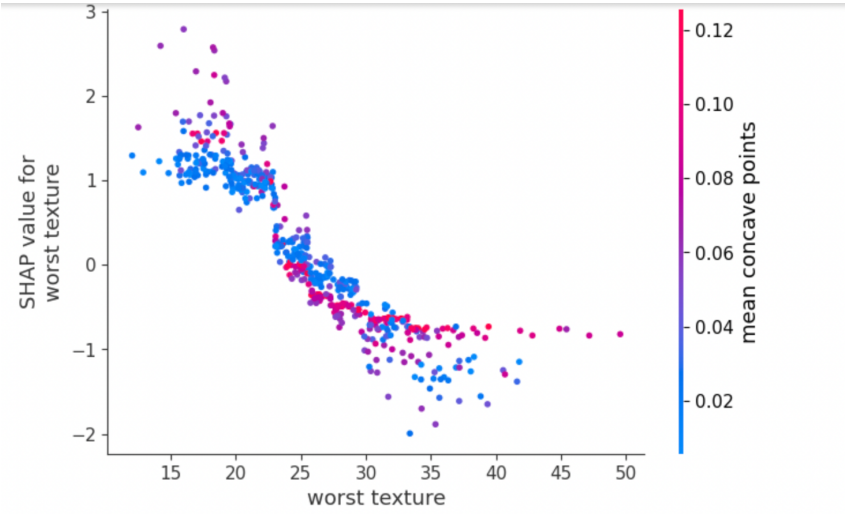

SHAP value for
mean symmetry
0.15
0.10
0.05
0.00
−0.05
−0.10
0.100 0.125 0.150 0.175 0.200 0.225 0.250 0.275 0.300
mean symmetry
mean texture
26
24
22
20
18
16
14

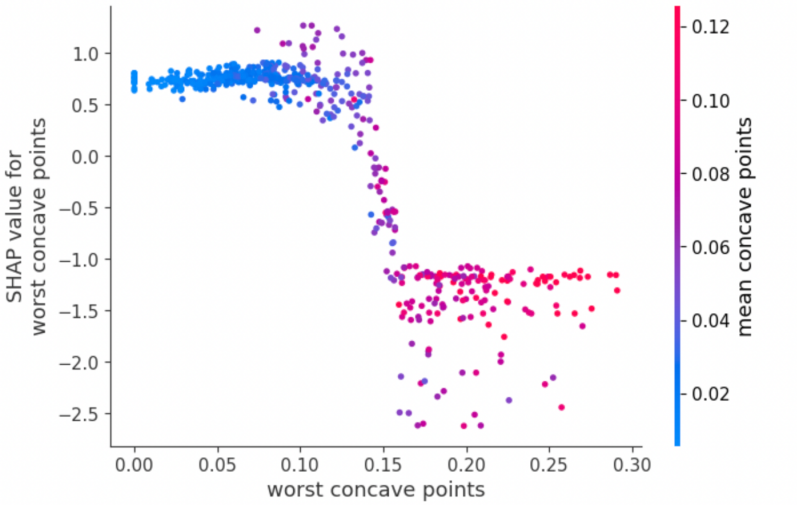

SHAP value for
mean smoothness
0.6
0.4
0.2
0.0
−0.2
−0.4
0.06 0.08 0.10 0.12 0.14 0.16
mean smoothness
worst texture
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5

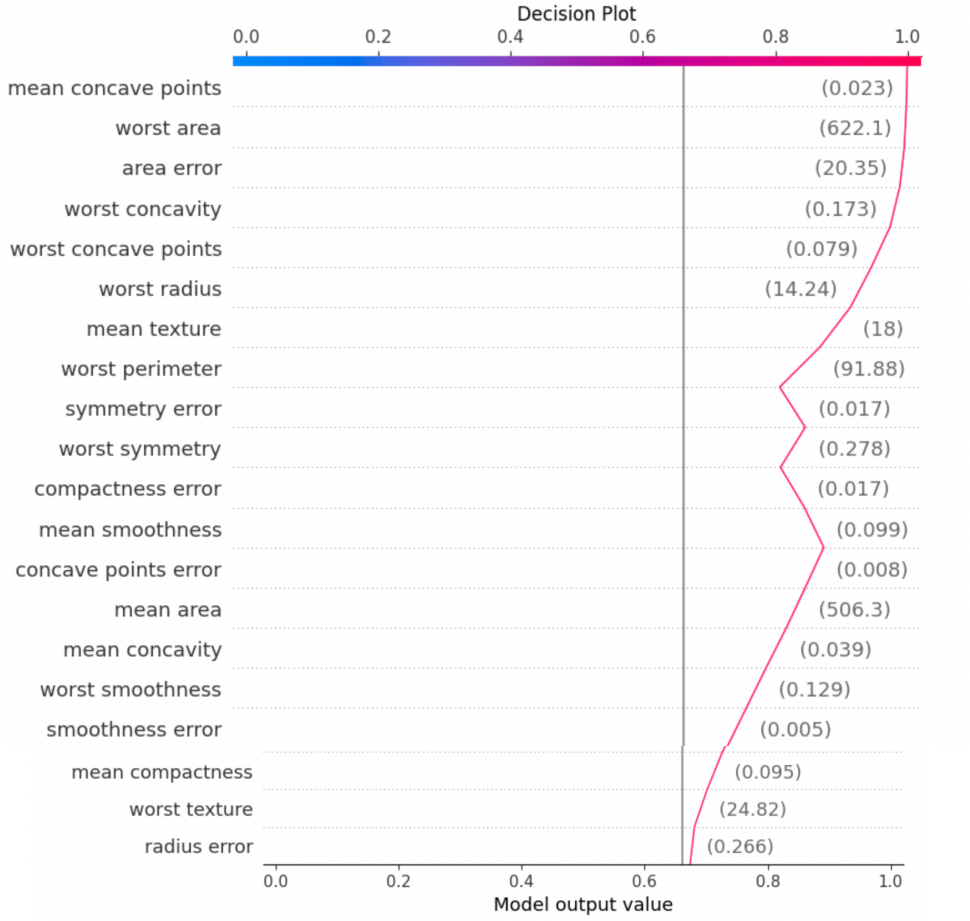

SHAP value for
worst area
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0 500 1000 1500 2000 2500 3000 3500 4000
worst area
worst texture
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5


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 perimeter
worst perimeter
worst texture
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
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5


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





Conclusion: In conclusion, the application of SHAP, LIME, and SHAPASH to the XG Boost model for breast cancer data has substantially improved interpretability. SHAP values have globally elucidated feature importance, LIME has locally explained predictions, and SHAPASH has provided user-friendly visualizations. This comprehensive approach enhances transparency, fostering trust in the model's predictions and offering valuable insights for diagnostic decision-making in breast cancer.