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)

COURSE CODE: DJ19DSC501 **DATE:**

COURSE NAME: Machine Learning - II **CLASS: AY 2023-24**

LAB EXPERIMENT NO.10

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.

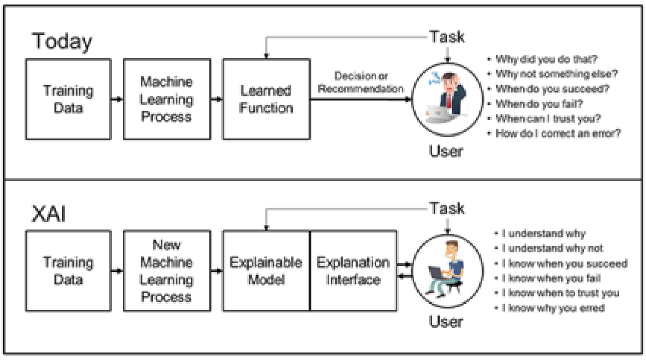
Industry Need of XAI.

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.



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.**

Implementing explainable AI using LIME and SHAP

In [1]:

```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import lime
import shap
import lime.lime_tabular
from lime import submodular_pick

from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder # For transforming categories to integer labels
```

In [2]:

```
data = load_breast_cancer()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = data.target
df.head(5)
```

Out[2]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	wo a
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	201
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	195
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	170
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	56
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	157

5 rows × 31 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  Target                   569 non-null    int32  
```

30 Target 569 non-null int32

```
dtypes: float64(30), int32(1)
memory usage: 135.7 KB
```

4]:

```
df.shape
```

4]: (569, 31)

5]:

```
X = data['data']
Y = data['target']
features = data.feature_names
```

6]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=42)
```

XG BOOST

In [7]:
```
model = XGBClassifier(n_estimators = 300, random_state = 123)
model.fit(X_train, Y_train)
```

Out[7]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=300,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=123, ...)
```

In [8]:
```
Y_pred = model.predict(X_test)
```

In [9]:
```
accuracy = accuracy_score(Y_test, Y_pred)

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

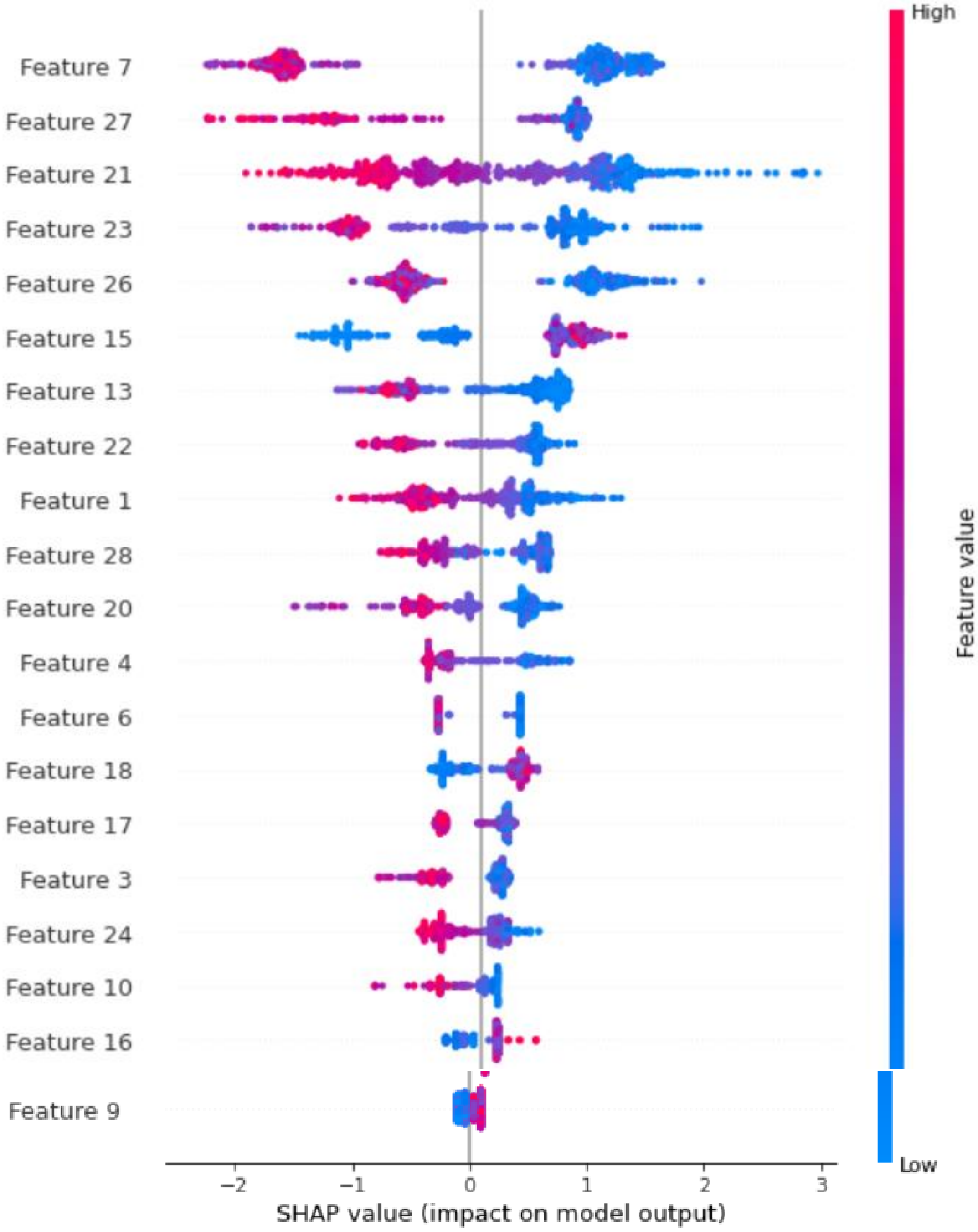
```
Accuracy: 98.25%
```

In [10]:
```
predict_fn = lambda x: model.predict_proba(x)
```

SHAP

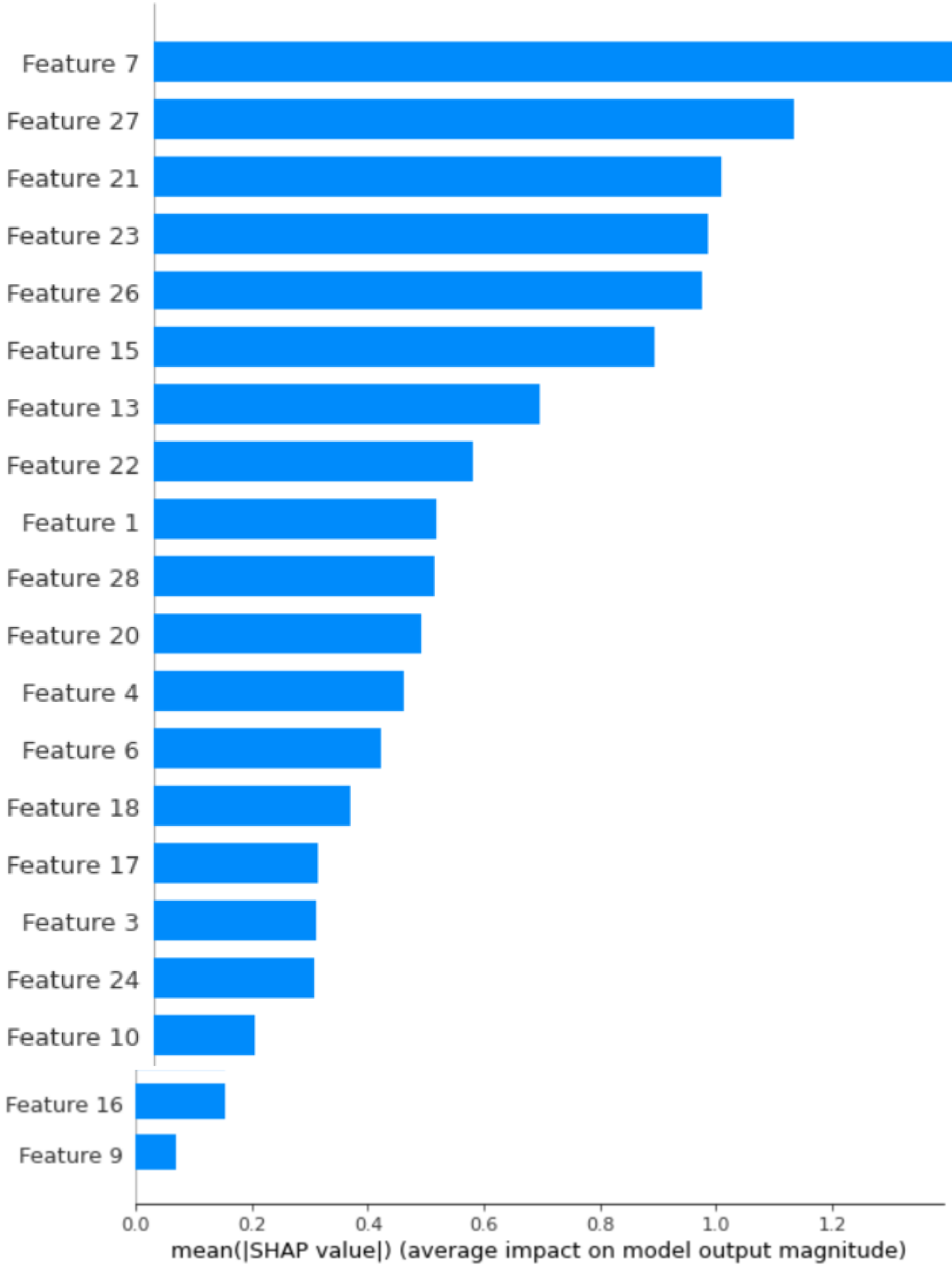
In [11]:
```
explainer = shap.TreeExplainer(model)
#fast and exact method to estimate SHAP values for tree models and ensembles of trees,

shap_values = explainer.shap_values(X)

expected_value = explainer.expected_value
```

In [12]:
```
shap.summary_plot(shap_values, X,title="SHAP summary plot")
```


Feature 7
Feature 27
Feature 21
Feature 23
Feature 26
Feature 15
Feature 13
Feature 22
Feature 1
Feature 28
Feature 20
Feature 4
Feature 6
Feature 18
Feature 17
Feature 3
Feature 24
Feature 10
Feature 16
Feature 9
−2
−1
0
1
2
3
SHAP value (impact on model output)
High
Low
Feature value

In [13]:

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



In [14]:

```
shap.plots._waterfall.waterfall_legacy(expected_value, shap_values[79], features=X[79,:], feature_names=features, max_displa
```

LIME

In [11]:

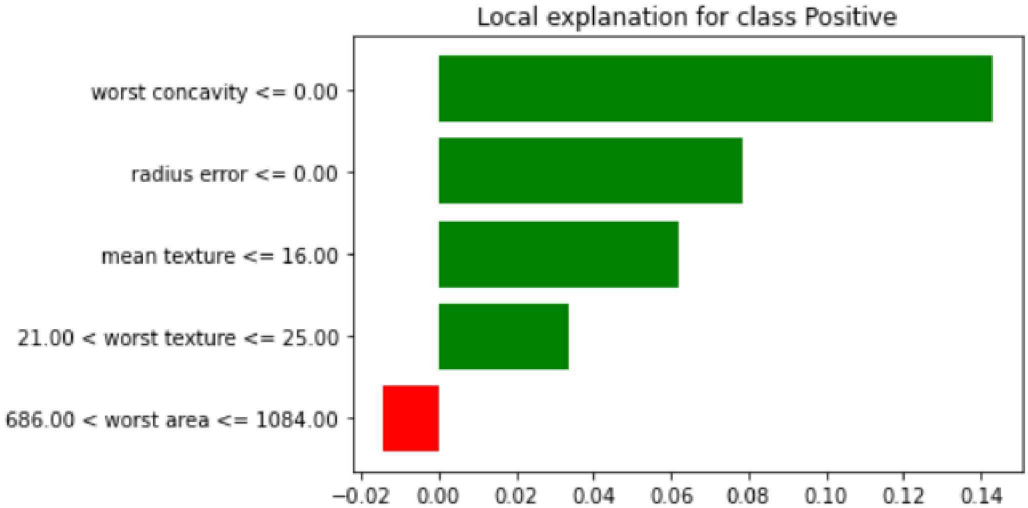
```
np.random.seed(123)

# Defining the LIME explainer object
explainer = lime.lime_tabular.LimeTabularExplainer(df[features].astype(int).values,
                                                   mode='classification',
                                                   class_names=['Negative', 'Positive'],
                                                   training_labels=df['Target'],
                                                   feature_names=features)
```

In [12]:

```
# using LIME to get the explanations
i = 5
exp = explainer.explain_instance(df.loc[i,features].astype(int).values, predict_fn, num_features=5)
exp.show_in_notebook(show_table=True)
```

In [13]:

```
figure = exp.as_pyplot_figure(label = exp.available_labels()[0])
```


Local explanation for class Positive
worst concavity <= 0.00
radius error <= 0.00
mean texture <= 16.00
21.00 < worst texture <= 25.00
686.00 < worst area <= 1084.00
−0.02
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14