GRADIENT BOOSTING FRAMEWORK

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

Load datset

```
import pandas as pd
from sklearn.datasets import load_breast_cancer,fetch_california_housing

# reg - dataset
california_housing = fetch_california_housing(as_frame=True)
housing_df = california_housing.frame
print("REGRESSION")
display(housing_df.head())

# cls - dataset
breast_cancer = load_breast_cancer(as_frame=True)
Bcancer_df = breast_cancer.frame
print("\nCLASSIFICATION")
display(Bcancer_df.head())
```

```
REGRESSION
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

```
CLASSIFICATION
```

	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

shape

```
print("reg",housing_df.shape)
print("cls",Bcancer_df.shape)
```

```
reg (20640, 9)
cls (569, 31)
```

info

```
housing_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   MedInc       20640 non-null  float64
 1   HouseAge     20640 non-null  float64
 2   AveRooms     20640 non-null  float64
 3   AveBedrms    20640 non-null  float64
 4   Population   20640 non-null  float64
 5   AveOccup     20640 non-null  float64
 6   Latitude     20640 non-null  float64
 7   Longitude    20640 non-null  float64
 8   MedHouseVal  20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

```
Bcancer_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    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

```
housing_df["MedHouseVal"].median()
```

```
1.797
```

remove noise columns

```
housing_df.drop(columns=["Population", "Latitude", "Longitude"],inplace=True)
```

```
Bcancer_df.drop(columns=[col for col in Bcancer_df.columns if "error" in col],inplace=True)
```

```
Bcancer_df.columns
```

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension',
       'target'],
      dtype='object')
```

```
print("reg",housing_df.shape)
print("cls",Bcancer_df.shape)
```

```
reg (20640, 6)
cls (569, 21)
```

train test split

```
from sklearn.model_selection import train_test_split

# For housing_df
X_housing = housing_df.drop(columns=['MedHouseVal'])
y_housing = housing_df['MedHouseVal']
X_train_housing, X_test_housing, y_train_housing, y_test_housing = train_test_split(X_housing, y_housing, test_size=0.2, random_st


# For Bcancer_df
X_bcancer = Bcancer_df.drop(columns=['target'])
y_bcancer = Bcancer_df['target']
X_train_bcancer, X_test_bcancer, y_train_bcancer, y_test_bcancer = train_test_split(X_bcancer, y_bcancer, test_size=0.2, random_st
```

XGBoost

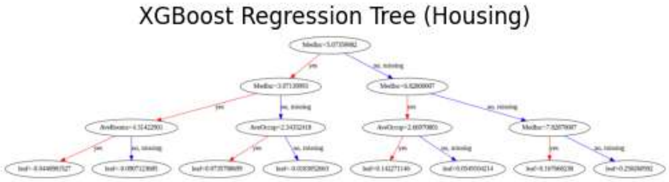
REGRESSION

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score

# Model
xgb_reg = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
xgb_reg.fit(X_train_housing, y_train_housing)

# Predict & Evaluate
y_pred = xgb_reg.predict(X_test_housing)
print("XGBoost Regression MSE:", mean_squared_error(y_test_housing, y_pred))
print("R2:", r2_score(y_test_housing, y_pred))
```

```
XGBoost Regression MSE: 0.4411640910853581
R2: 0.6633386229712962
```

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from xgboost import plot_tree

plt.figure(figsize=(20,10))
plot_tree(xgb_reg, num_trees=0)
plt.title("XGBoost Regression Tree (Housing)")
plt.show()
```

```
<Figure size 2000x1000 with 0 Axes>
```



CLASSIFICATION

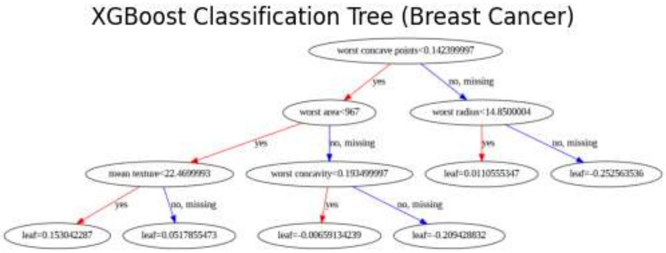
```
from sklearn.metrics import accuracy_score

# Model
xgb_clf = xgb.XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
xgb_clf.fit(X_train_bcancer, y_train_bcancer)

# Predict & Evaluate
```

```
y_pred_c = xgb_clf.predict(X_test_bcancer)
print("XGBoost Classification Accuracy:", accuracy_score(y_test_bcancer, y_pred_c))
```

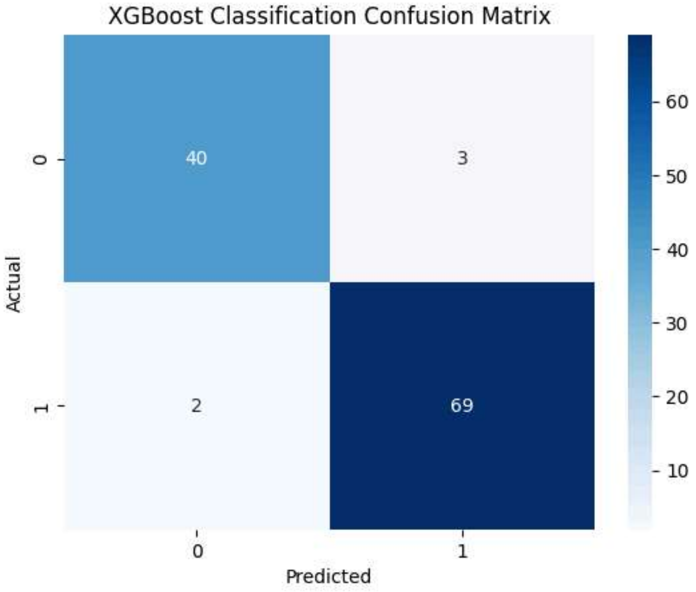
```
XGBoost Classification Accuracy: 0.956140350877193
```

```
plt.figure(figsize=(15,8))
plot_tree(xgb_clf, num_trees=0)
plt.title("XGBoost Classification Tree (Breast Cancer)")
plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```



```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test_bcancer, y_pred_c)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("XGBoost Classification Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



LightGBM

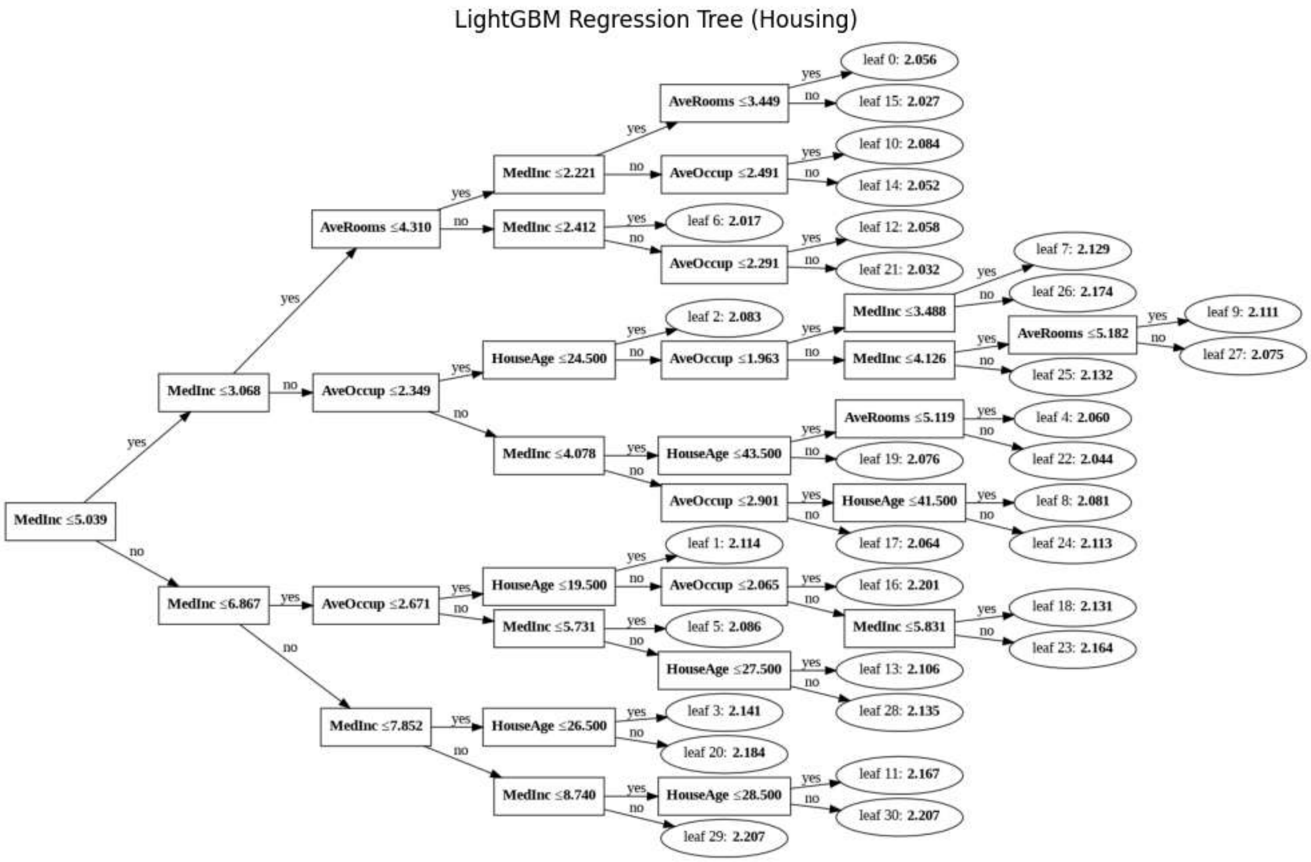
REGRESSION

```
import lightgbm as lgb
from sklearn.metrics import mean_squared_error, r2_score

lgb_reg = lgb.LGBMRegressor(
    n_estimators=200,
    learning_rate=0.05,
    max_depth=-1,
    num_leaves=31,
    min_child_samples=5,
    random_state=42,
     verbose=-1)
```

```
lgb_reg.fit(X_train_housing, y_train_housing)
y_pred_L = lgb_reg.predict(X_test_housing)
print("LightGBM Regression MSE:", mean_squared_error(y_test_housing, y_pred_L))
print("R2:", r2_score(y_test_housing, y_pred_L))
```

```
LightGBM Regression MSE: 0.4189918915761474
R2: 0.6802586837136873
```

```
import matplotlib.pyplot as plt
import lightgbm as lgb
from lightgbm import plot_tree

ax = plot_tree(lgb_reg, tree_index=0, figsize=(15,8))
plt.title("LightGBM Regression Tree (Housing)")
plt.show()
```

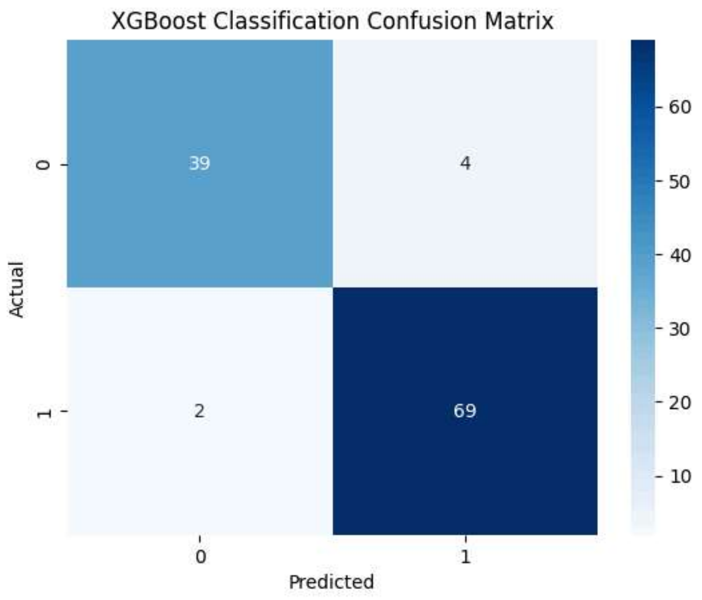


CLASSIFICATION

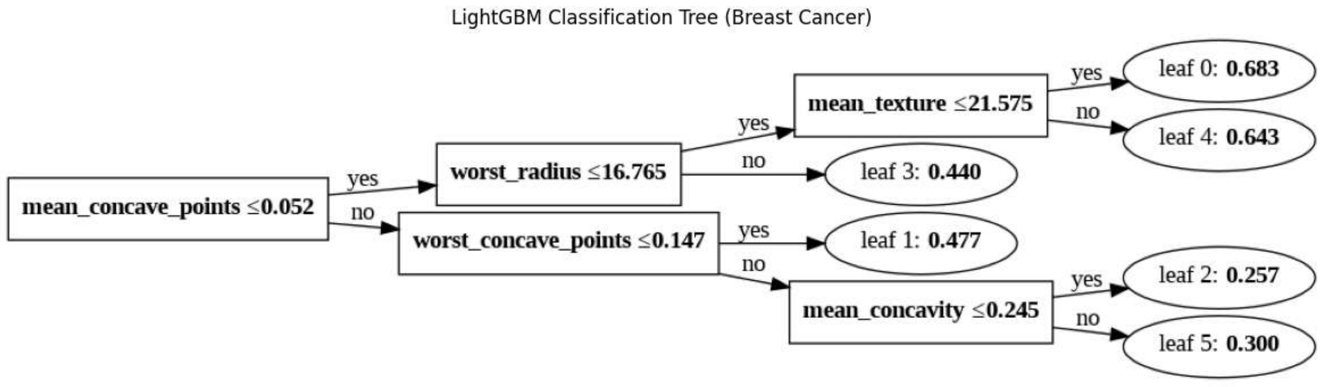
```
lgb_clf = lgb.LGBMClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
lgb_clf.fit(X_train_bcancer, y_train_bcancer)

y_pred_LC = lgb_clf.predict(X_test_bcancer)
print("LightGBM Classification Accuracy:", accuracy_score(y_test_bcancer, y_pred_LC))
```

```
LightGBM Classification Accuracy: 0.9473684210526315
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test_bcancer, y_pred_LC)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("XGBoost Classification Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



```
ax = plot_tree(lgb_clf, tree_index=0, figsize=(15,8))
plt.title("LightGBM Classification Tree (Breast Cancer)")
plt.show()
```



CatBoost

```
pip install catboost
```

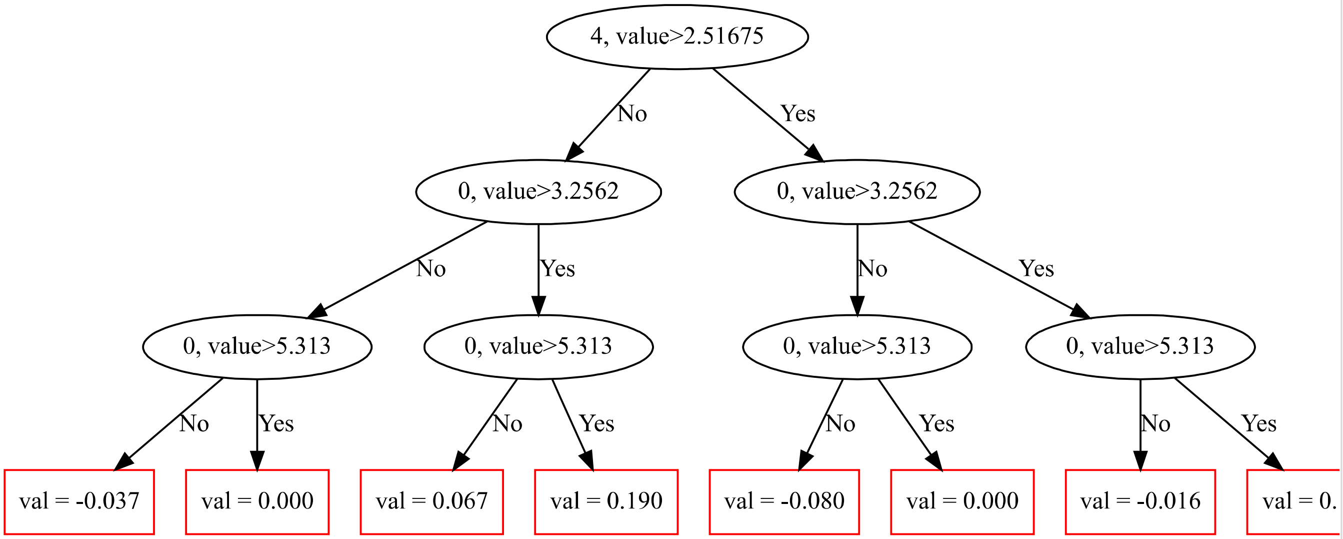
```
Requirement already satisfied: catboost in /usr/local/lib/python3.12/dist-packages (1.2.8)
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-packages (from catboost) (0.21)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from catboost) (1.16.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2.9
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (4.60.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (3.2.4)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly->catboost) (8.5.0)
```

REGRESSION

```
from catboost import CatBoostRegressor

cat_reg = CatBoostRegressor(iterations=100, learning_rate=0.1, depth=3, verbose=0, random_state=42)
cat_reg.fit(X_train_housing, y_train_housing)

y_pred_cat = cat_reg.predict(X_test_housing)
print("CatBoost Regression MSE:", mean_squared_error(y_test_housing, y_pred_cat))
print("R2:", r2_score(y_test_housing, y_pred_cat))
```

```
CatBoost Regression MSE: 0.4504888076209366
R2: 0.6562227403038132
```

```
cat_reg.plot_tree(tree_idx=0)
```

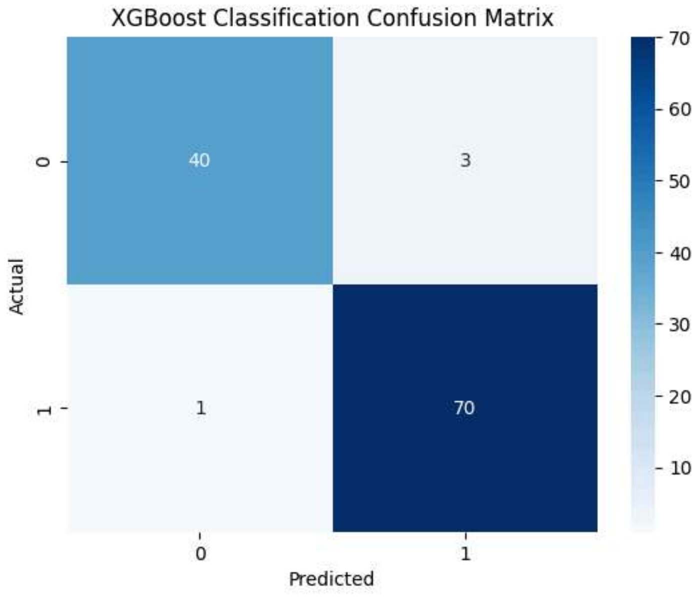


CLASSIFICATION

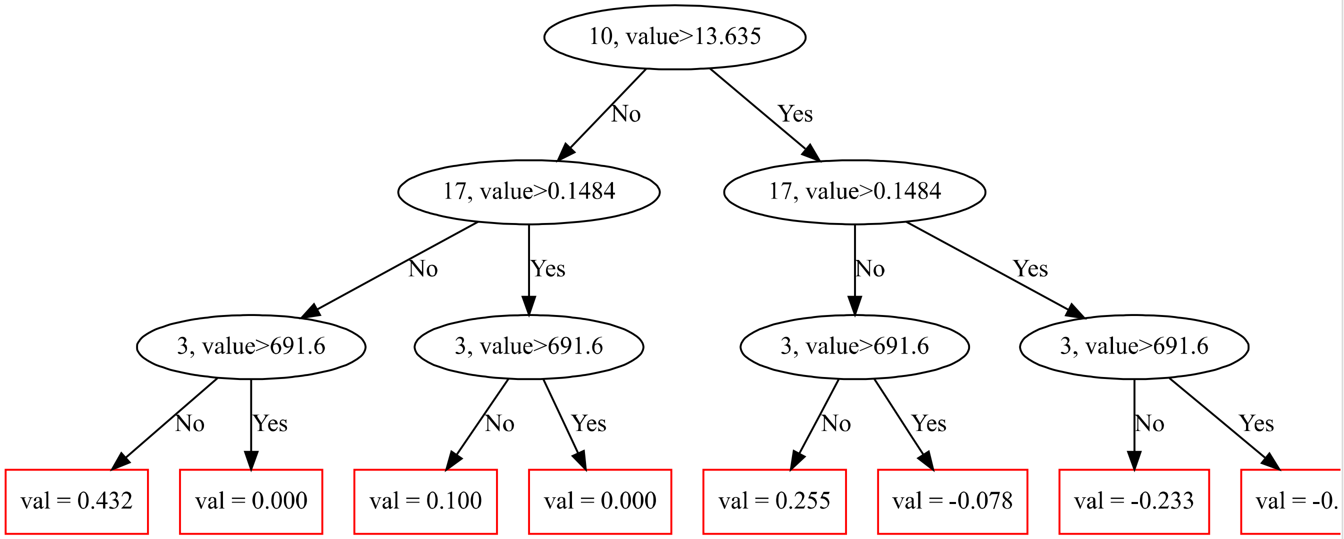
```
from catboost import CatBoostClassifier


cat_clf = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=3, verbose=0, random_state=42)
cat_clf.fit(X_train_bcancer, y_train_bcancer)

y_pred_catc = cat_clf.predict(X_test_bcancer)
print("CatBoost Classification Accuracy:", accuracy_score(y_test_bcancer, y_pred_catc))
```

```
CatBoost Classification Accuracy: 0.9649122807017544
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test_bcancer, y_pred_catc)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("XGBoost Classification Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



```
cat_clf.plot_tree(tree_idx=0)
```


10, value>13.635
No
Yes
17, value>0.1484
17, value>0.1484
No
Yes
No
Yes
3, value>691.6
3, value>691.6
3, value>691.6
3, value>691.6
No
Yes
No
Yes
No
Yes
No
Yes
val = 0.432
val = 0.000
val = 0.100
val = 0.000
val = 0.255
val = -0.078
val = -0.233
val = -0.