St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2025-2026
Class: BE-ITA/B, Semester: VII
Subject: Healthcare Lab

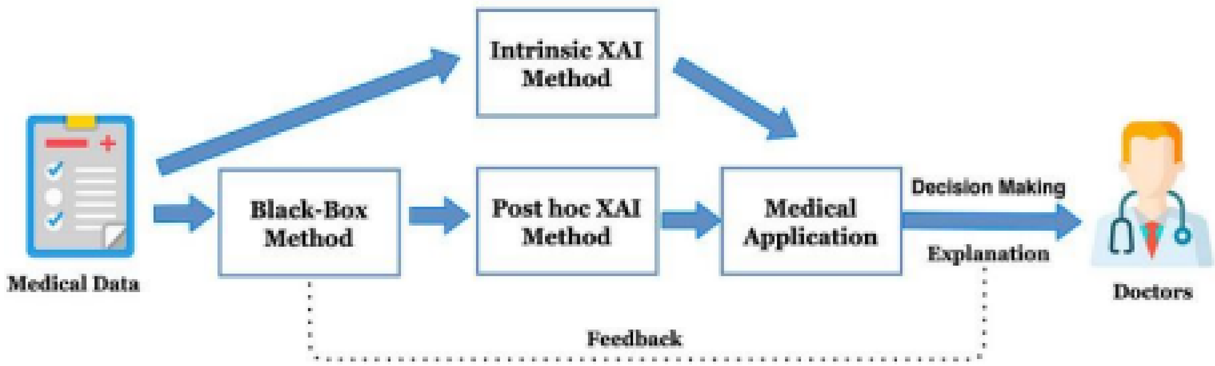
Experiment 8

1. **Aim:** To perform explainable AI on heart disease data.
2. **Objectives:** Students should be able to analyze and justify the performance of specific models as applied to healthcare problems.
3. **Prerequisite:** Python basics, A basic understanding of machine learning Models.
4. **Pre-Experiment Exercise:**

Theory:

<u>Explainable AI</u>

In general, the problem of explainable AI deals with so-called "black box" algorithms. In this type of AI, both the functioning of the algorithm and the final values of the parameters are known, but not why this result was achieved. With millions of parameters adjusted during a training process, the weighting cannot be traced back to a bigger picture. Consequently, the relationships between why a weight has a certain value and how it contributes to the overall model can no longer be explained.
Keeping these observations in mind, explainable AI can be defined as follows: "Given an audience, an explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand."
XAI can also be described as a bridge between human-computer interaction (HCI) and artificial intelligence. The main focus of XAI is mainly to explain the interaction to the end user in order to create a trustworthy environment



5. Laboratory Exercise:

Implementation :

```
!pip install eli5
!pip install pdpbox
!pip install shap

from collections import defaultdict
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy.stats import spearmanr
from scipy.cluster import hierarchy
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.inspection import permutation_importance
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score, roc_curve, auc
from xgboost import XGBClassifier, plot_importance
import warnings
import eli5
import shap
from eli5.sklearn import PermutationImportance
from pdpbox import pdp, get_dataset, info_plots
from sklearn.tree import DecisionTreeClassifier
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
%matplotlib inline

data = pd.read_csv('../input/heart-disease-cleveland-uci/heart_cleveland_upload.csv')
# To display the top 5 rows
data.head(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000
mean	54.542088	0.676768	2.158249	131.693603	247.350168	0.144781	0.996633	149.599327	0.326599	1.055556	0.602694
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	0.994914	22.941562	0.469761	1.166123	0.618187
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	2.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	0.000000
50%	56.000000	1.000000	2.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000
75%	61.000000	1.000000	3.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.000000	1.600000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000

```
data.shape
```

```
(297, 14)
```

\+ Code + Markdown

```
heart = data.copy()
```

```
target = 'condition'
features_list = list(heart.columns)
features_list.remove(target)
```

```
y = heart.pop('condition')
```

\+ Code + Markdown

```
X_train, X_test, y_train, y_test = train_test_split(heart, y, test_size=0.2, random_state=33)
X_train.shape, X_test.shape
```

```
((237, 13), (60, 13))
```

```
%%time

# ML in two lines ;)
xgb = XGBClassifier(objective='binary:logistic', random_state=33, n_jobs=-1)
xgb.fit(X_train, y_train)
```

```
[14:58:02] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the obj
ective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore th
e old behavior.
CPU times: user 223 ms, sys: 9.53 ms, total: 233 ms
Wall time: 72.6 ms
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1, random_state=33,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
# make predictions for test data
xgb_predictions = xgb.predict(X_test)
```

Permutation Importance

```
import eli5
from eli5.sklearn import PermutationImportance
eli5.show_weights(xgb.get_booster(), top=15)
```

Weight	Feature
0.2559	thal
0.1621	cp
0.1615	ca
0.0592	exang
0.0560	oldpeak
0.0507	slope
0.0466	fbs
0.0460	sex
0.0398	age
0.0348	chol
0.0318	thalach
0.0279	trestbps
0.0277	restecg

```
tgt = 6
```

```
print('Reference:', y_test.iloc[tgt])
print('Predicted:', xgb_predictions[tgt])
eli5.show_prediction(xgb.get_booster(), X_test.iloc[tgt],
                     feature_names=list(heart.columns), show_feature_values=True)
```

```
Reference: 1
Predicted: 1
```

y (score 8.832) top features

Contribution?	Feature	Value
+2.232	ca	2.000
+1.348	oldpeak	2.800
+1.124	cp	3.000
+1.028	thal	2.000
+0.649	restecg	2.000
+0.645	trestbps	145.000
+0.543	age	60.000
+0.474	chol	282.000
+0.461	exang	1.000
+0.395	sex	1.000
+0.312	slope	1.000
+0.060	fbs	0.000
+0.023	thalach	142.000
-0.464	<BIAS>	1.000

```
%%time

# we need to retrain a new model with arrays
# as eli5 has a bug with Dataframes and XGBoost
# cf. https://github.com/TeamHG-Memex/eli5/pull/261
xgb_array = XGBClassifier(objective='binary:logistic', random_state=33, n_jobs=-1)
xgb_array.fit(X_train.values, y_train)
```

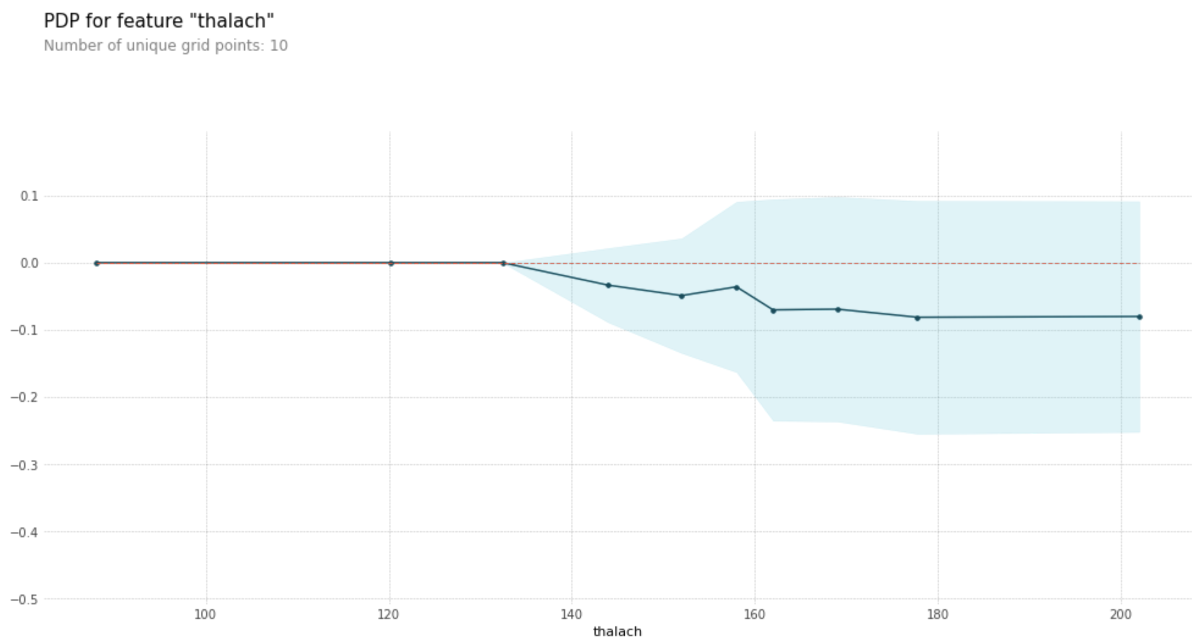
```
[14:58:36] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the obj
ective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore th
e old behavior.
CPU times: user 202 ms, sys: 8.49 ms, total: 210 ms
Wall time: 63.8 ms
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1, random_state=33,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
model = DecisionTreeClassifier(random_state=1)
model = model.fit(X_train, y_train)
permutation = PermutationImportance(model, random_state=33).fit(X_train, y_train)
eli5.show_weights(permutation, feature_names = features_list, top=30)
```

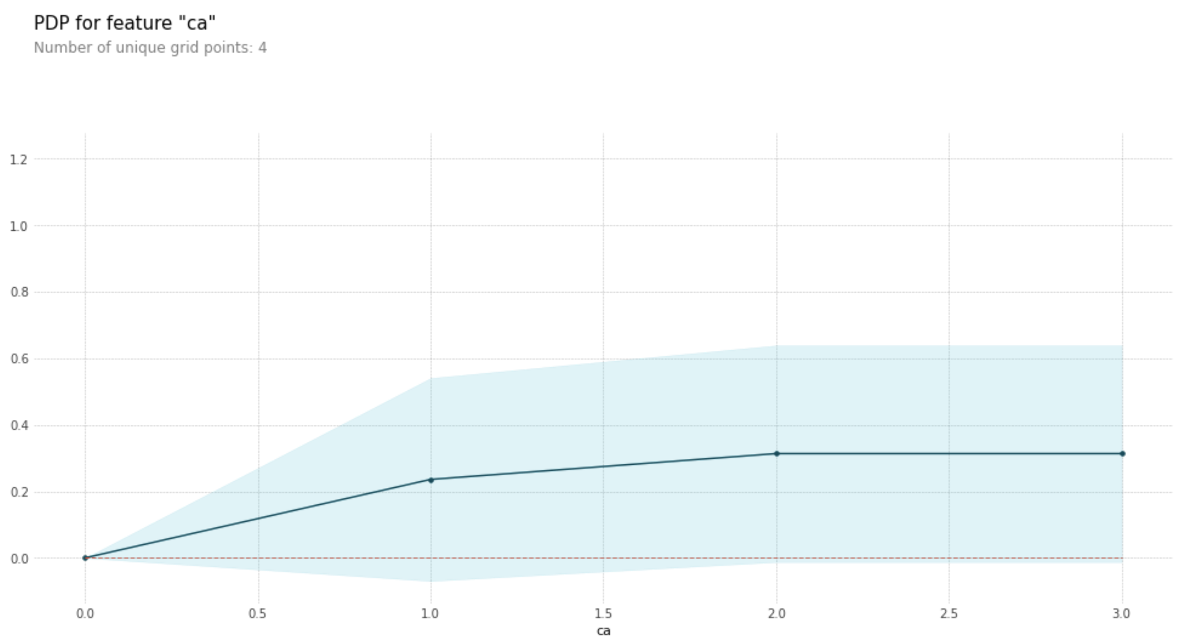
Weight	Feature
0.2025 ± 0.0509	thal
0.1899 ± 0.0381	ca
0.1266 ± 0.0350	age
0.0785 ± 0.0157	chol
0.0734 ± 0.0174	cp
0.0658 ± 0.0182	trestbps
0.0363 ± 0.0086	thalach
0.0346 ± 0.0216	oldpeak
0.0295 ± 0.0075	sex
0.0253 ± 0.0213	restecg
0.0152 ± 0.0086	exang
0.0127 ± 0.0053	fbs
0.0034 ± 0.0098	slope

PDP

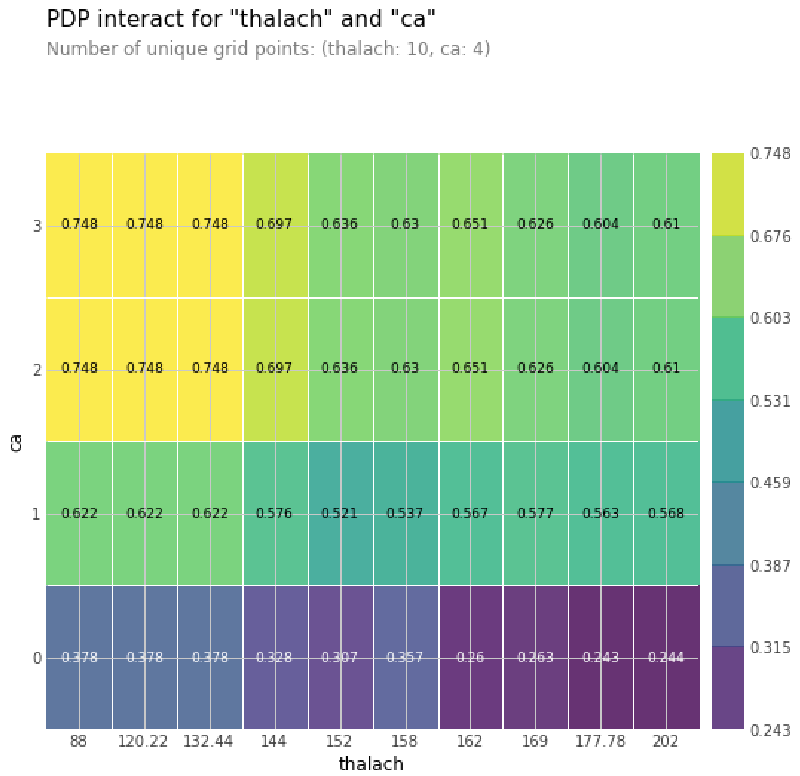
```
def plot_pdp(model, df, feature, cluster_flag=False, nb_clusters=None, lines_flag=False):
    pdp_goals = pdp.pdp_isolate(model=model, dataset=df, model_features=df.columns.tolist(),
feature=feature)
    pdp.pdp_plot(pdp_goals, feature, cluster=cluster_flag, n_cluster_centers=nb_clusters, plot_lines=lines_flag)
    plt.show()
plot_pdp(xgb, X_train, 'thalach')
```



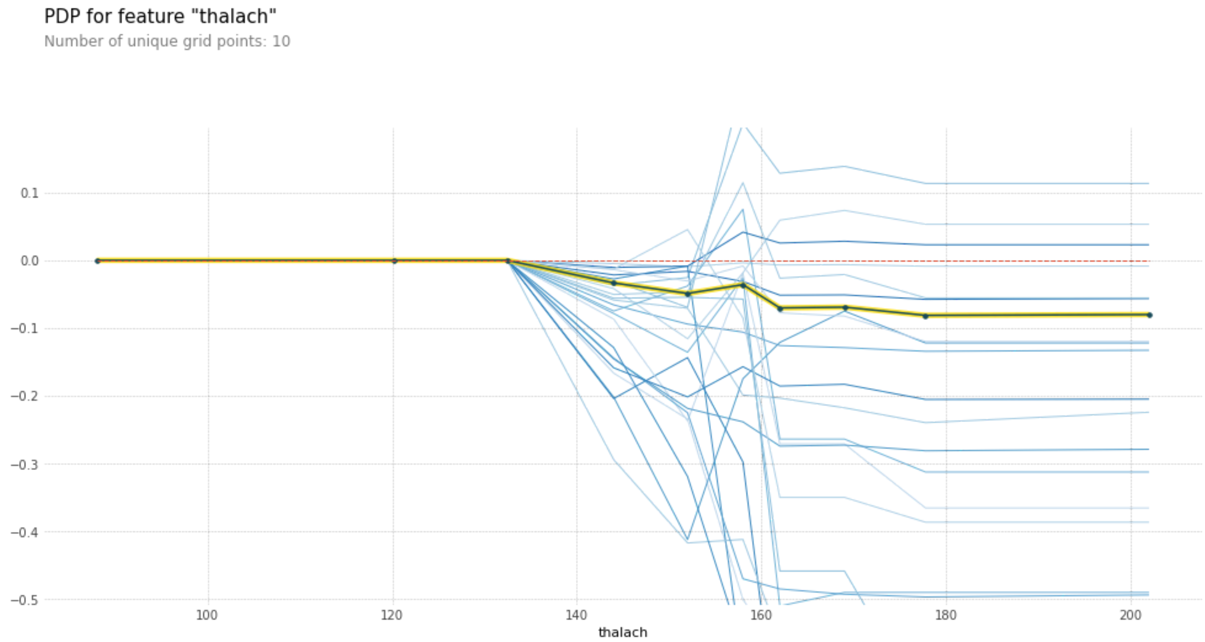

```
plot_pdp(xgb, X_train, 'ca')
```




```
features_to_plot = ['thalach', 'ca']
inter1  =  pdp.pdp_interact(model=xgb, dataset=X_train, model_features=features_list,
features=features_to_plot)
pdp.pdp_interact_plot(pdp_interact_out=inter1, feature_names=features_to_plot, plot_type='grid')
plt.show()
```



plot_pdp(xgb, X_train, 'thalach', cluster_flag=True, nb_clusters=24, lines_flag=True)

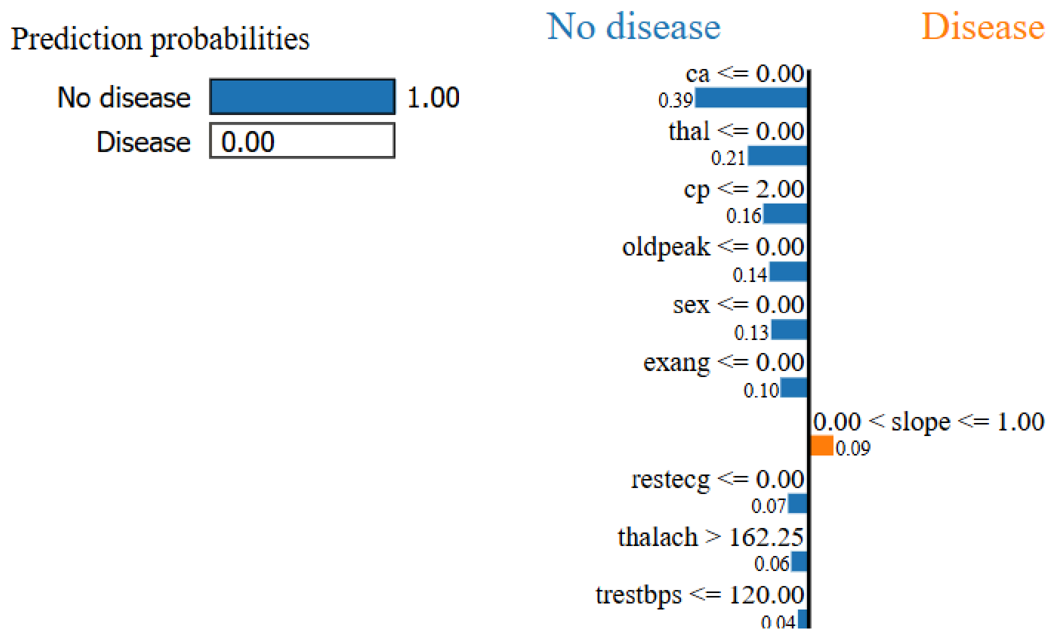



```
!pip install skater
from skater.core.explanations import Interpretation
from skater.model import InMemoryModel

interpreter = Interpretation(training_data=X_test, feature_names=features_list)
im_model = InMemoryModel(xgb.predict_proba, examples=X_train, target_names=['Disease', 'No Disease'])

predictions = xgb_array.predict_proba(X_test.values)
```

LIME

```
from skater.core.local_interpretation.lime.lime_tabular import LimeTabularExplainer

exp = LimeTabularExplainer(X_test.values, feature_names=features_list, discretize_continuous=True,
class_names=['No disease', 'Disease'])
tgt = 1
print('Reference:', y_test.iloc[tgt])
print('Predicted:', predictions[tgt])
exp.explain_instance(X_test.iloc[tgt].values, xgb_array.predict_proba).show_in_notebook()
```

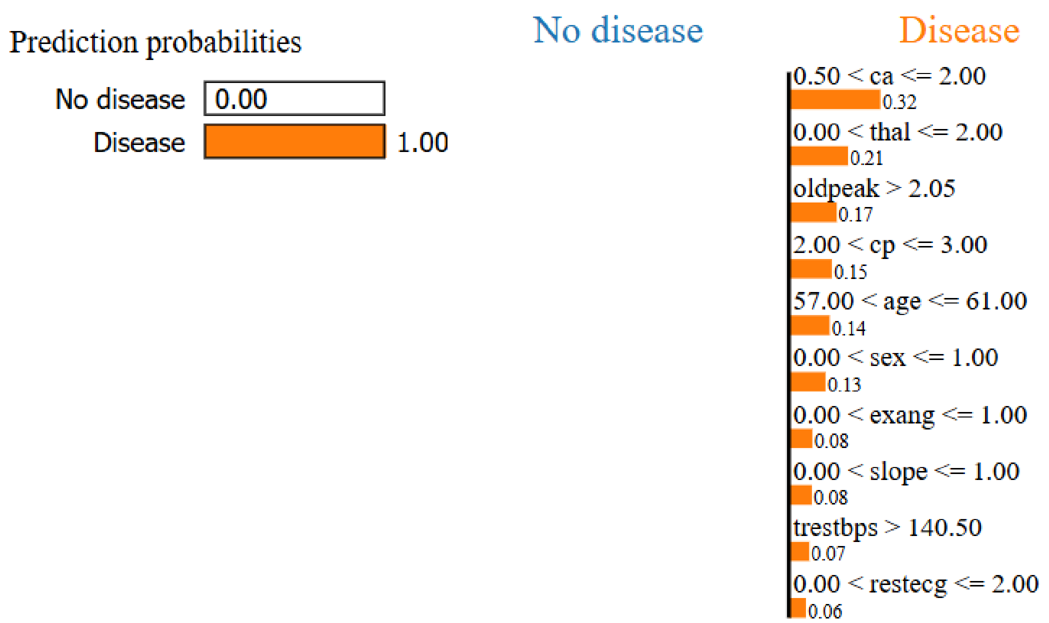
```
Reference: 0
Predicted: [0.9983744  0.00162558]
```



Feature	Value
ca	0.00
thal	0.00
cp	2.00
oldpeak	0.00
sex	0.00
exang	0.00
slope	1.00
restecg	0.00
thalach	173.00
trestbps	120.00

```
tgt = 6
print('Reference:', y_test.iloc[tgt])
print('Predicted:', predictions[tgt])
exp.explain_instance(X_test.iloc[tgt].values, xgb_array.predict_proba).show_in_notebook()
```

```
Reference: 1
Predicted: [1.4603138e-04 9.9985397e-01]
```

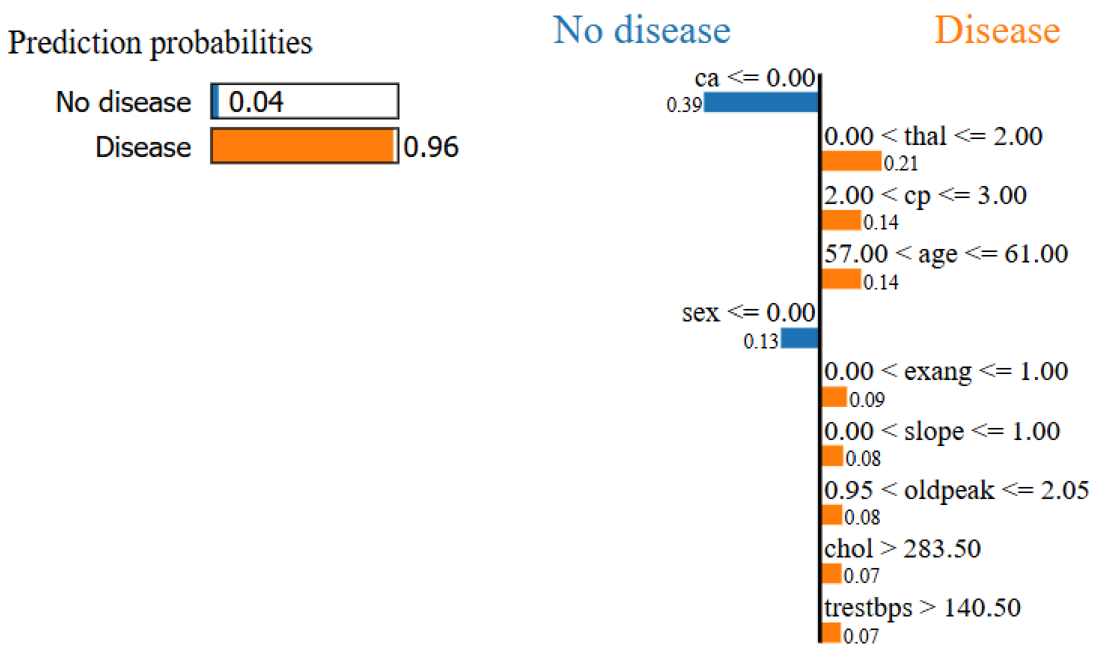


Feature	Value
ca	2.00
thal	2.00
oldpeak	2.80
cp	3.00
age	60.00
sex	1.00
exang	1.00
slope	1.00
trestbps	145.00
restecg	2.00

```
tgt = 15
print('Reference:', y_test.iloc[tgt])
print('Predicted:', predictions[tgt])
exp.explain_instance(X_test.iloc[tgt].values, xgb_array.predict_proba).show_in_notebook()
```

Feature	Value
ca	0.00
thal	2.00
cp	3.00
age	61.00
sex	0.00
exang	1.00
slope	1.00
oldpeak	1.00
chol	307.00
trestbps	145.00

```
Reference: 1
Predicted: [0.03760159 0.9623984 ]
```



6. Post-Experiments Exercise

A. Extended Theory:

a. Write advantage and disadvantage of explainable AI in healthcare applications.

B. Conclusion:

- Write what was performed in the program (s).
- What is the significance of program and what Objective is achieved?

C. References:

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

[2] https://www.kaggle.com/code/smitisinghal/explainable-ai-on-heart-disease-dataset

[3][https://medium.com/@thinkdata/state-of-the-art-of-explainable-ai-in-healthcare-in 2022-c02225deba1](https://medium.com/@thinkdata/state-of-the-art-of-explainable-ai-in-healthcare-in-2022-c02225deba1)