In [1]:
```
import os
import sys
import csv
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
from sklearn import *
from  sklearn.tree._classes import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
import shap
shap.initjs()
```

Load heart.csv dataset

In [2]:
```
data = pd.read_csv('heart copy.csv')
data.head()
```

Out[2]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Split Dataset and Scale Train Data

In [3]:
```
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)



# scaling data
# scaler = StandardScaler()
# X_train = scaler.fit_transform(X_train)
# print('\nChecking X-variable values after scaling\n', X_train)
```

Train sets using Gini Index

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower gini index should be preferred.
- Sklearn supports “gini” criteria for Gini Index and by default, it takes “gini” value.

In [4]:
```
feature_names = []
for feature in X:
    feature_names.append(feature)
print('\nChecking feature names in columns from X-variable: \n', feature_names)

# Check SHAP output when DecisionTreeClassifier has pre-defined train parameters vs no parameters
# dt_gini = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=17, min_samples_split=45, min_samples_

dt_gini = DecisionTreeClassifier()


dt_gini.fit(X_train, y_train)
```

```
Checking feature names in columns from X-variable:
 ['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh', 'exng', 'oldpeak', 'slp', 'caa', 'thall']
```

Out[4]:
```
DecisionTreeClassifier()
```

Calculate Model Predictions

In [5]:
```
y_pred = dt_gini.predict(X_test)
print("Predicted values:")
print(y_pred)
```

```
Predicted values:
[0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 0
 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1
 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1]
```

Calculate Model Accuracy

In [6]:
```
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("\nAccuracy: \n", (accuracy_score(y_test,y_pred)*100))
print("\nReport: \n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[31 10]
 [17 33]]

Accuracy:
 70.32967032967034

Report:
              precision    recall  f1-score   support

           0       0.65      0.76      0.70        41
           1       0.77      0.66      0.71        50

    accuracy                           0.70        91
   macro avg       0.71      0.71      0.70        91
weighted avg       0.71      0.70      0.70        91
```

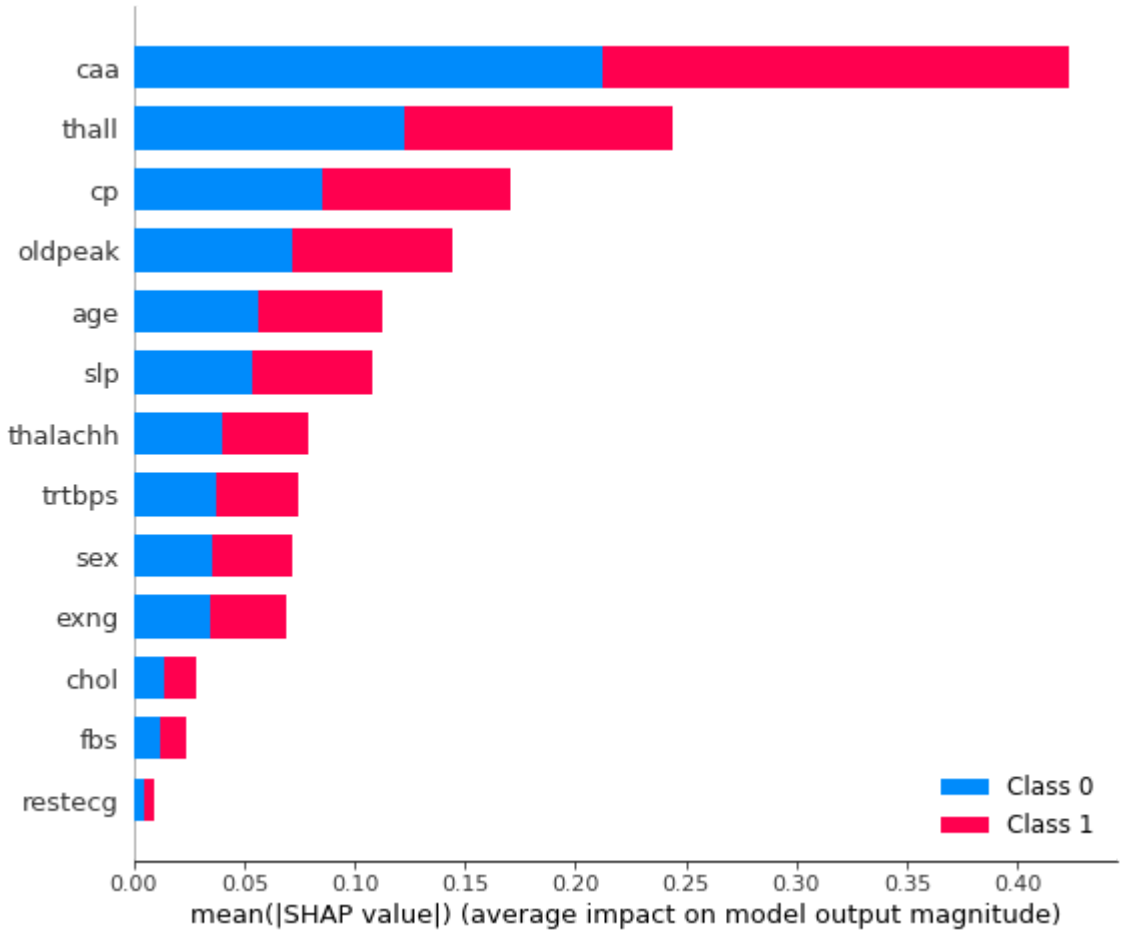
Calculate SHAP

NOTES

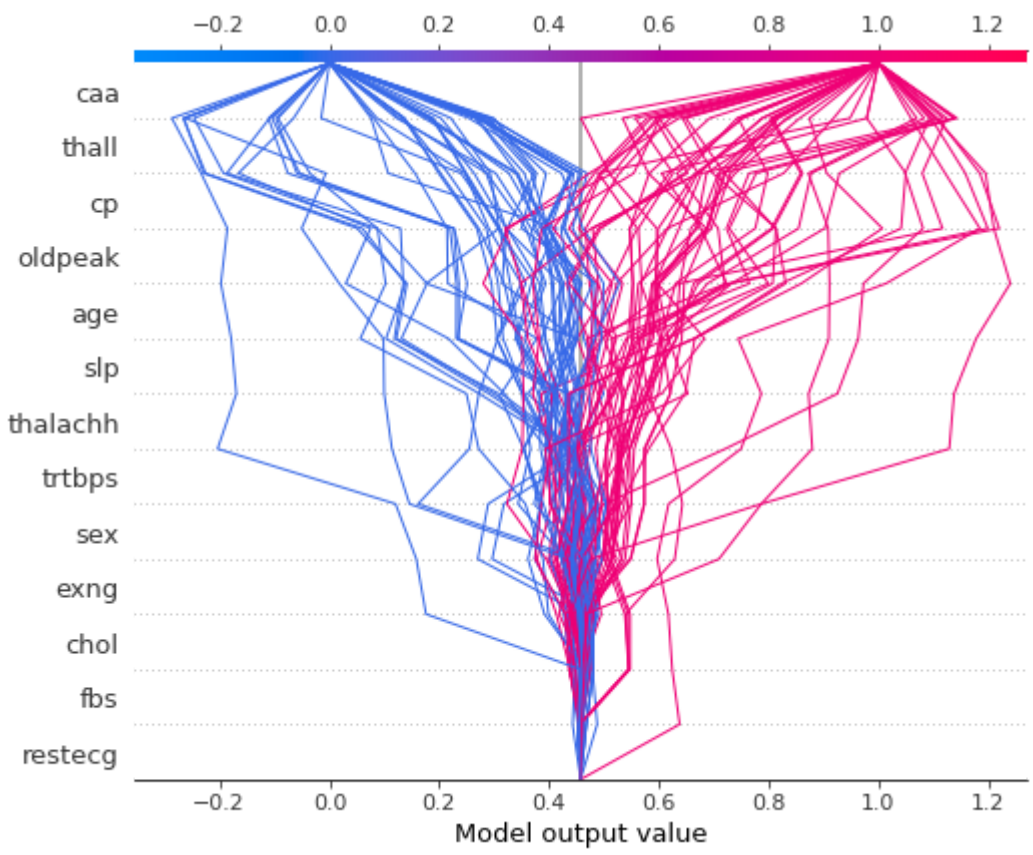
- Using Force Plot on model results in "NotImplementedError: matplotlib = True is not yet supported for force plots with multiple samples!"

In [8]:
```
explainer = shap.TreeExplainer(dt_gini)
shap_values = explainer.shap_values(X_test)

# shap.TreeExplainer(model).shap_interaction_values(X).

print('\nSummary SHAP Bar Plot: \n')
shap.summary_plot(shap_values, feature_names, plot_type='bar')


print('\nDecision Plot for SHAP Values from Class 0 in Test Set: \n')
shap.decision_plot(explainer.expected_value[0], shap_values[0], feature_names)

print('\nDecision Plot for SHAP Values from Class 1 in Test Set: \n')
shap.decision_plot(explainer.expected_value[1], shap_values[1], feature_names)


# print('\nForce Plot for Local, Instance-Wise Effects: \n')
# shap.force_plot(explainer.expected_value[0], shap_values[0])
# shap.force_plot(explainer.expected_value[0], shap_values[0], X_test.iloc[1],  matplotlib = True, show = False)


print('\nSummary SHAP Bar Plot for SHAP Values from Class 0: \n')
for feature in feature_names:
    shap.dependence_plot(feature, shap_values[0], X_test, interaction_index=None)
```

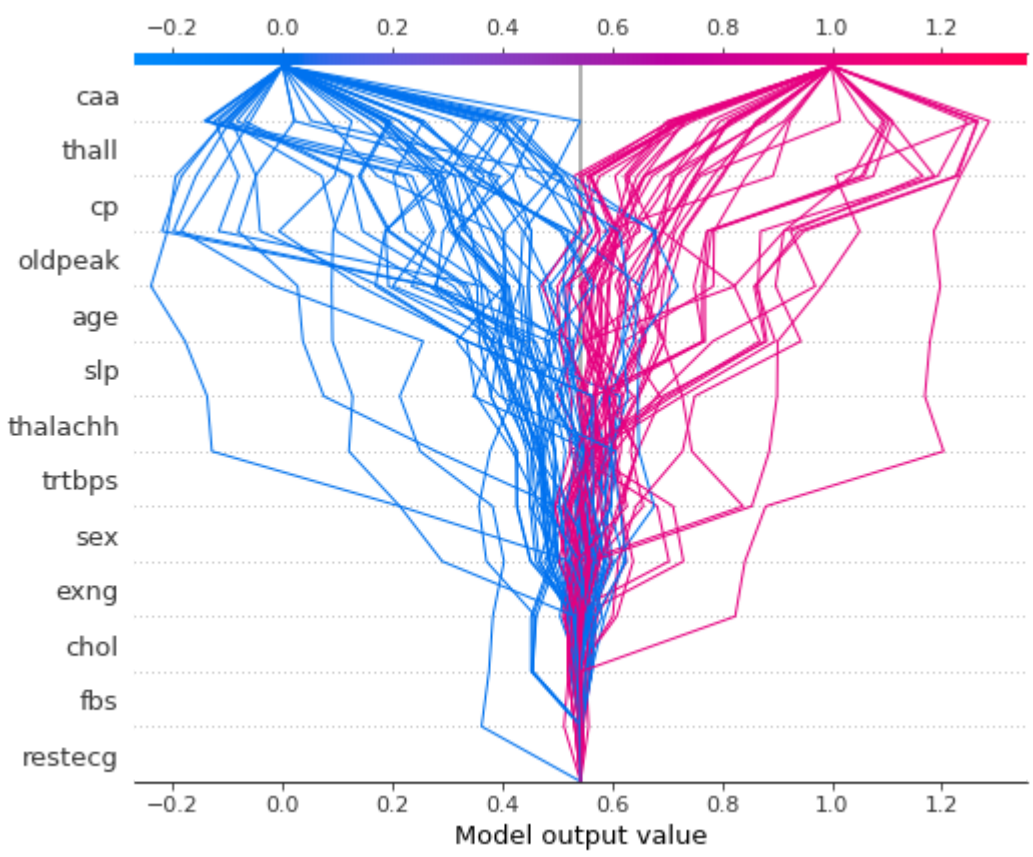
```
Summary SHAP Bar Plot:
```



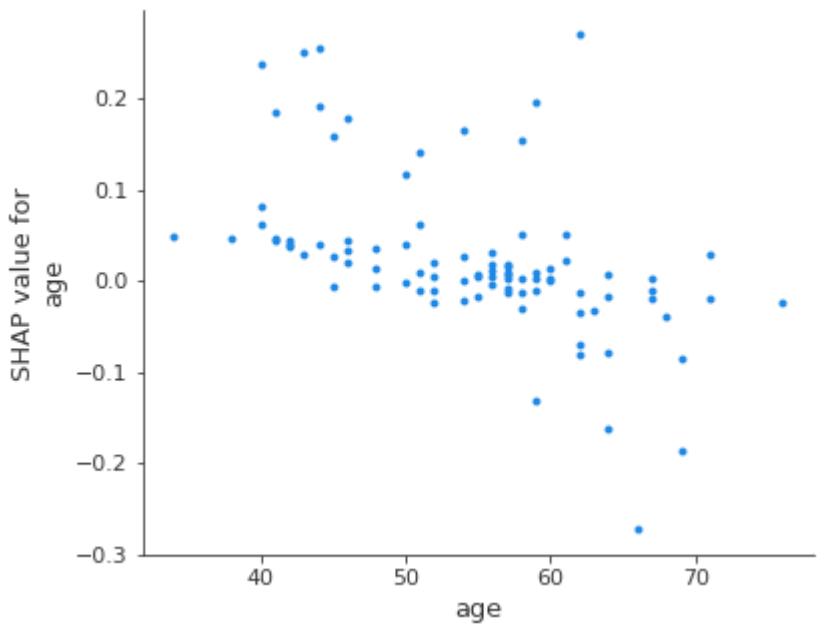
Decision Plot for SHAP Values from Class 0 in Test Set:

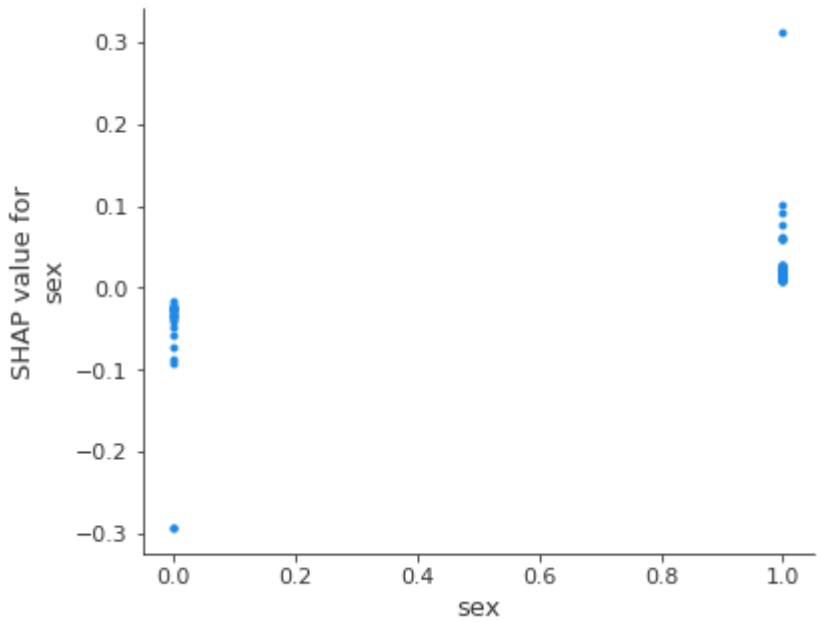


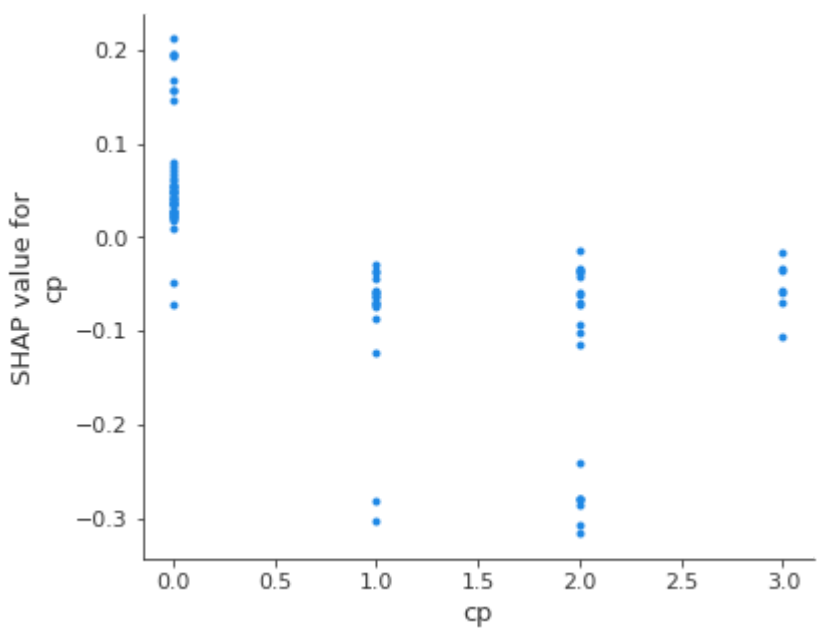
Decision Plot for SHAP Values from Class 1 in Test Set:

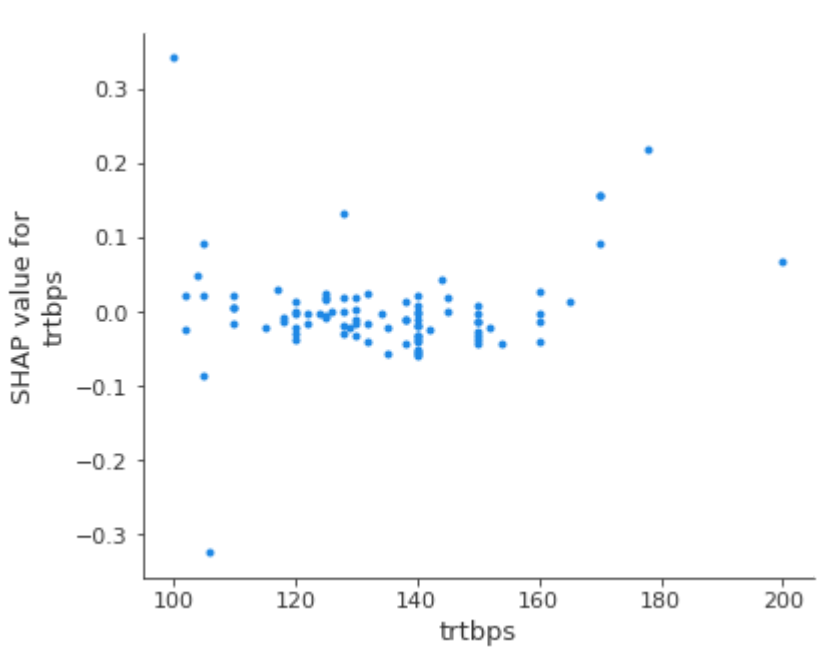


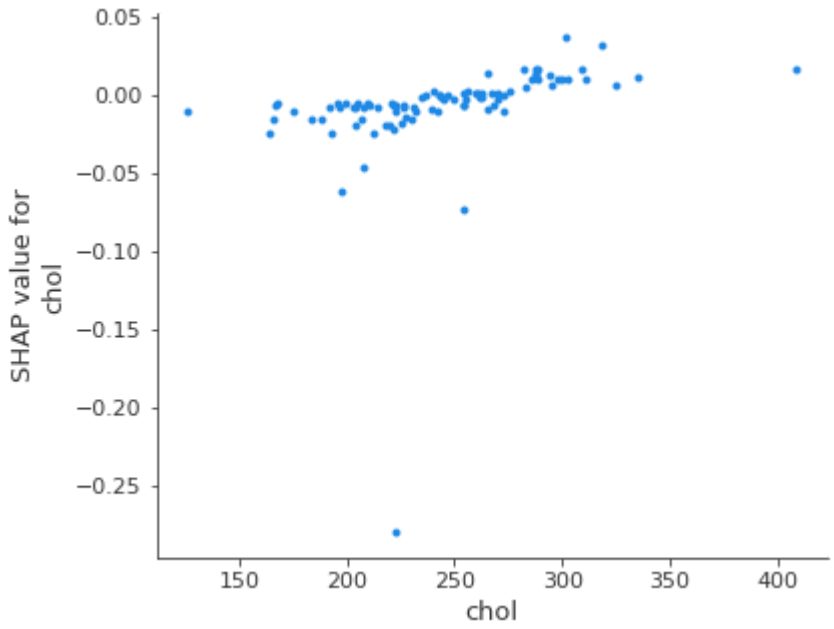
Summary SHAP Bar Plot for SHAP Values from Class 0:

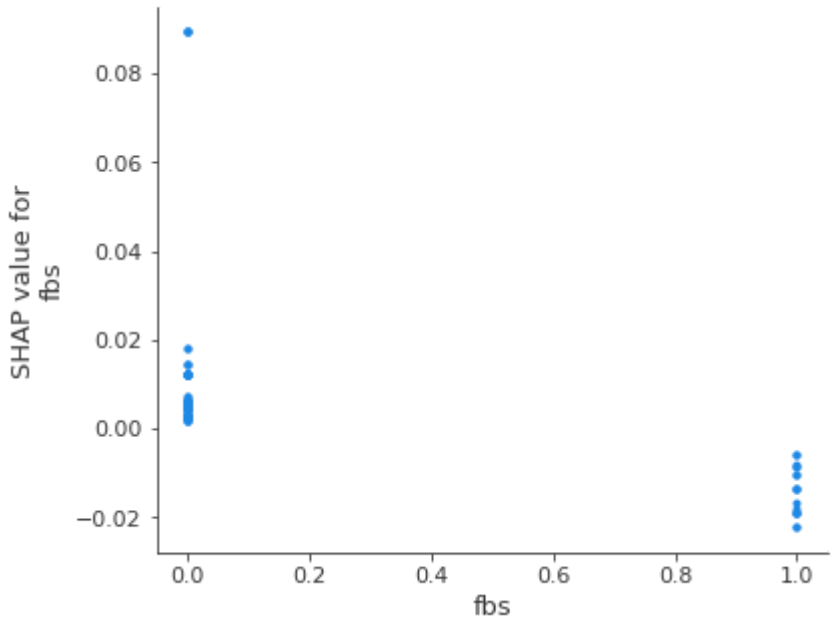

SHAP value for
age
0.2
0.1
0.0
−0.1
−0.2
−0.3
40
50
60
70
age

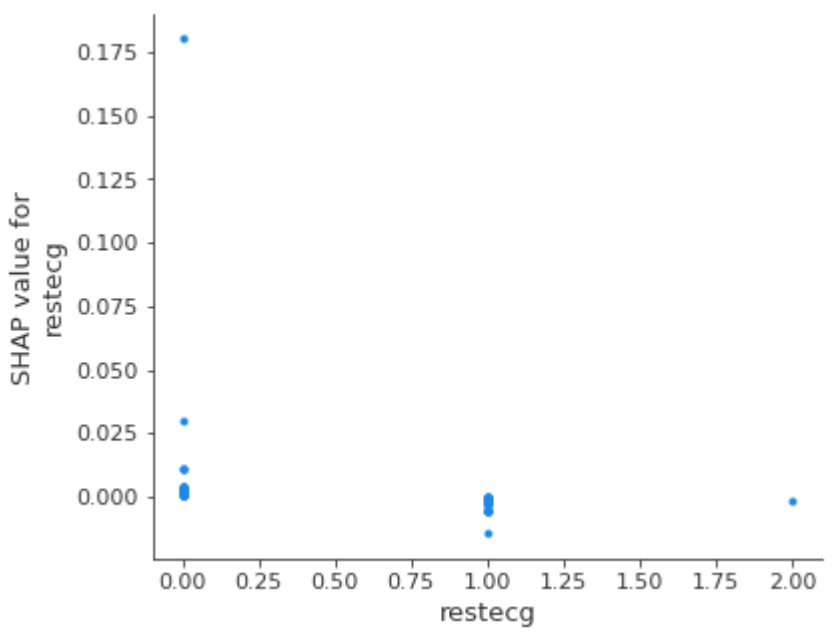

SHAP value for
sex
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.0
0.2
0.4
0.6
0.8
1.0
sex

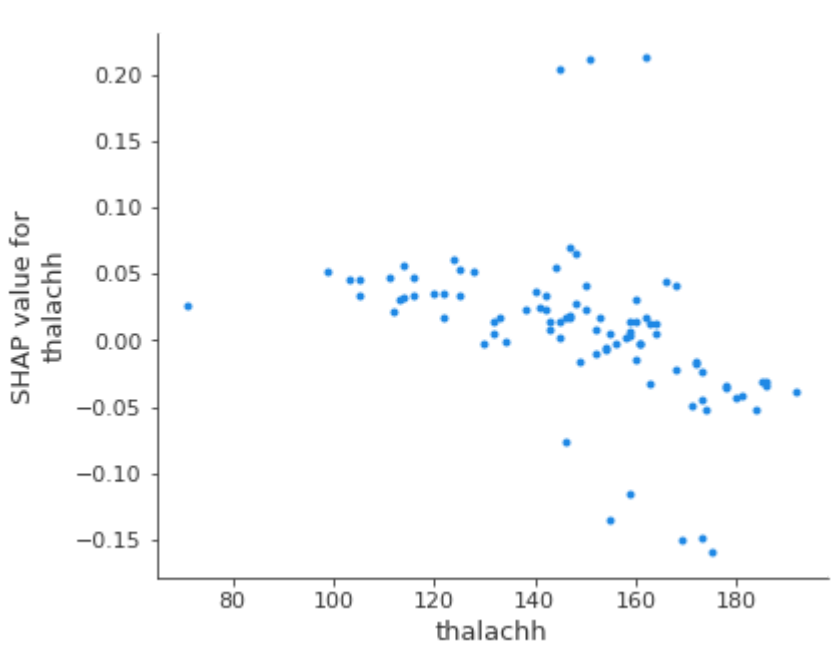

SHAP value for
cp
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
cp

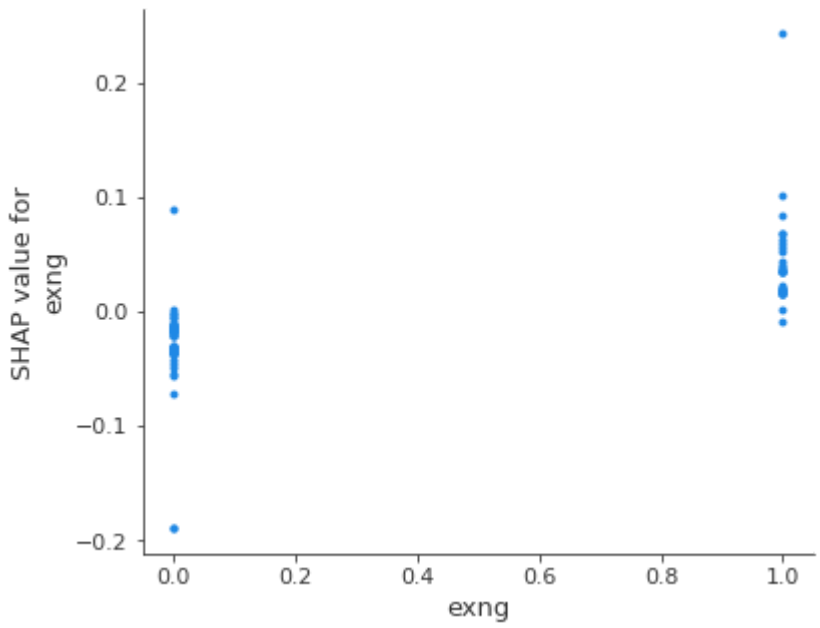

SHAP value for
trtbps
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
100
120
140
160
180
200
trtbps

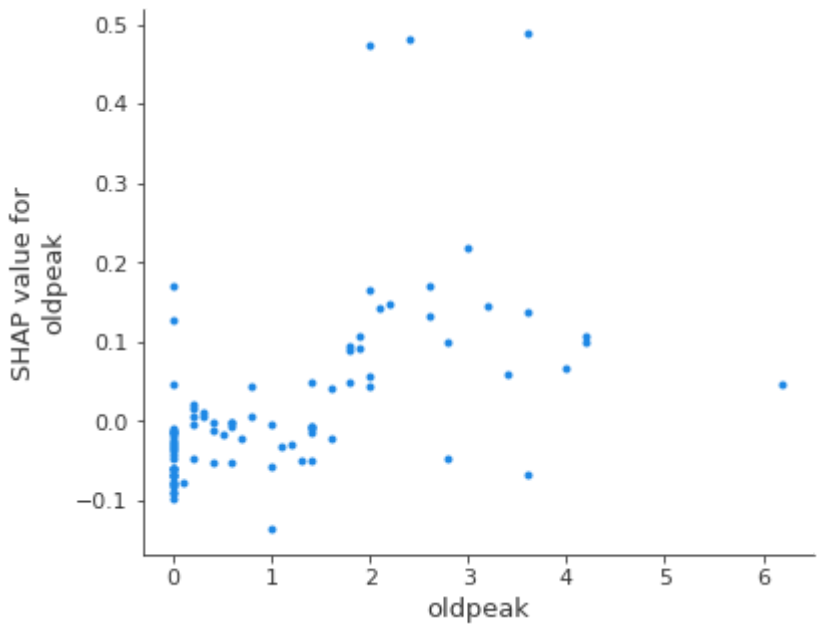

SHAP value for
chol
chol
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
150
200
250
300
350
400

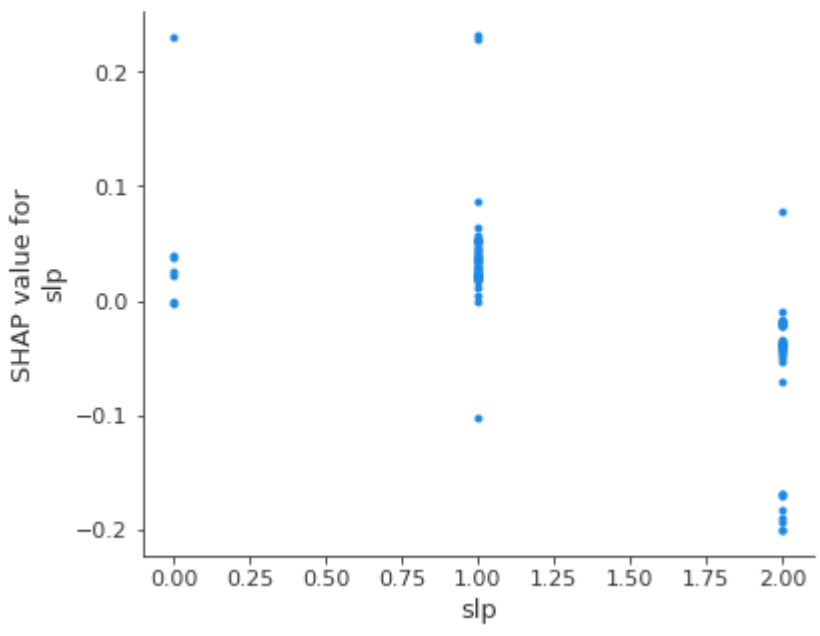

SHAP value for
fbs
fbs
0.08
0.06
0.04
0.02
0.00
−0.02
0.0
0.2
0.4
0.6
0.8
1.0

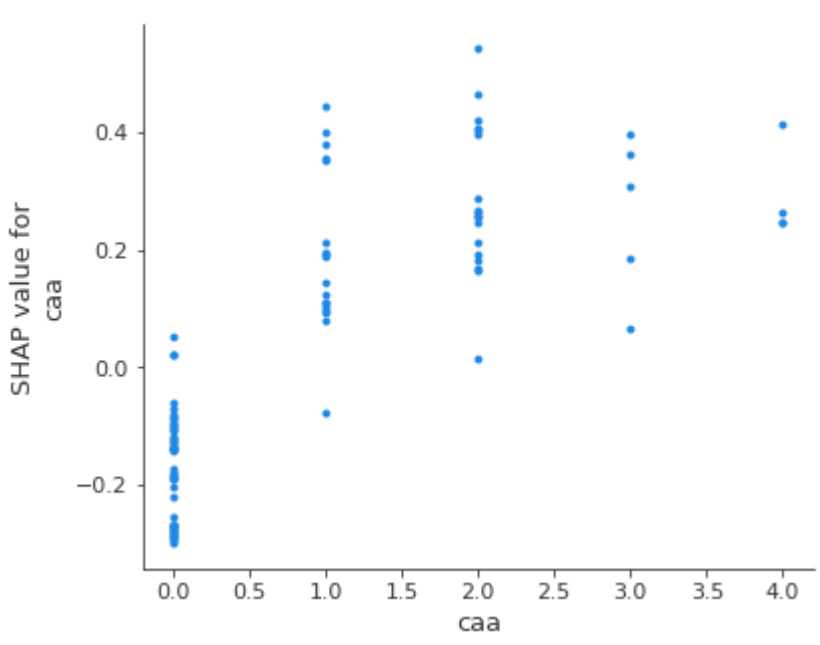

SHAP value for
restecg
restecg
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

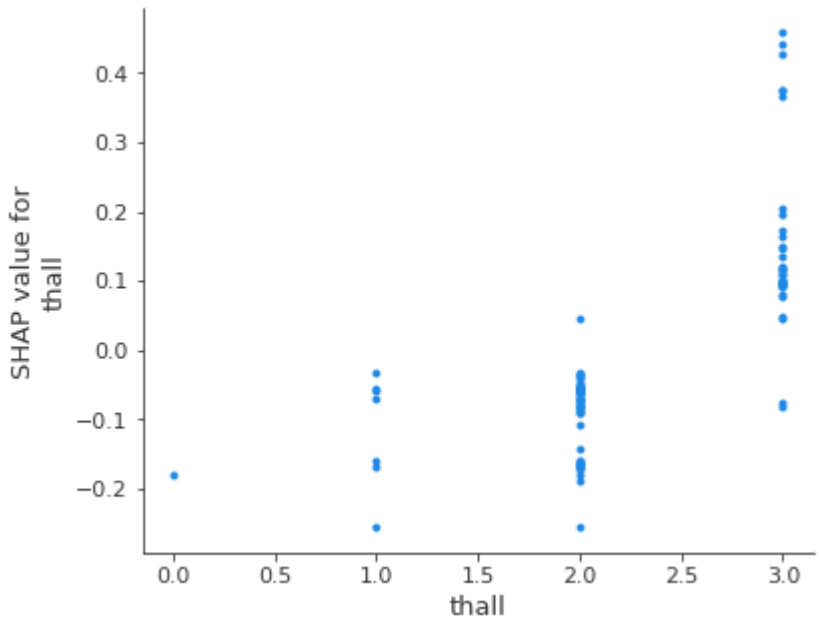

SHAP value for
thalachh
thalachh
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
80
100
120
140
160
180


SHAP value for
exng
exng
0.2
0.1
0.0
−0.1
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


SHAP value for
oldpeak
oldpeak
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
0
1
2
3
4
5
6

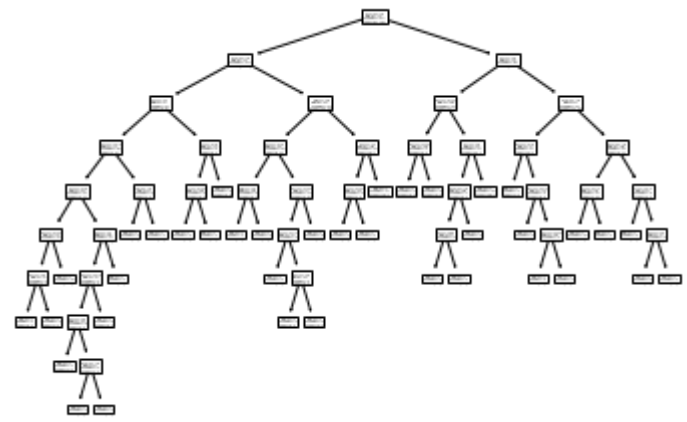

SHAP value for
slp
slp
0.2
0.1
0.0
−0.1
−0.2
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


SHAP value for
caa
caa
0.4
0.2
0.0
−0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0



In [9]:
```
print("Showing how Decision Tree Made Predictions...\n")
tree_graph = tree.plot_tree(dt_gini)
```

```
Showing how Decision Tree Made Predictions...
```



In [ ]:

In [ ]: