In [ ]:

```
%matplotlib inline

import os
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

In [ ]:

```
eng_levels = [0, 1]
```

Face Features

In [ ]:

```
base_dir = "../pose-action/features/"
```

In [ ]:

```
# Load data
df = pd.read_csv(os.path.join(base_dir, 'pose_keypoints_with_labels.csv'))
labels = (df['label'] <= 0).astype(int) #binarize labels
df = df.iloc[:,:-1]
df.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	...	290	291	292	293	294	295	296	297
0	178.864	180.766	0.968300	198.379	261.046	0.854602	131.697	270.835	0.764647	125.874	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	739.041	147.424	0.908013	758.704	212.014	0.949156	715.641	210.162	0.872539	695.949	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	196.373	180.794	0.960075	206.199	261.029	0.837968	133.707	266.929	0.740769	120.070	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	196.522	182.778	0.885628	208.134	259.180	0.857302	133.798	263.053	0.737462	120.051	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	694.069	196.346	0.955753	729.249	214.107	0.902440	686.212	217.955	0.793841	668.633	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 300 columns

In [ ]:

```
print(len(df), len(labels))
```

```
6935 6935
```

In [ ]:

```
labels.value_counts()
```

Out[ ]:

```
0    5491
1    1444
Name: label, dtype: int64
```

In [ ]:

```
r_p1 = range(0,24)
r_p2 = range(75,99)
r_p3 = range(150, 174)
r_p4 = range(225,249)
r_all = np.r_[r_p1, r_p2, r_p3, r_p4]

df_p1 = df.iloc[:, r_p1]
df_p2 = df.iloc[:, r_p2]
df_p3 = df.iloc[:, r_p3]
df_p4 = df.iloc[:, r_p4]
df_all = df.iloc[:, r_all]
```

In [ ]:

```
df_p1['label'] = labels.values
df_p2['label'] = labels.values
df_p3['label'] = labels.values
df_p4['label'] = labels.values
df_all['label'] = labels.values
```

```
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_9320\2577857134.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  df_p1['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_9320\2577857134.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  df_p2['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_9320\2577857134.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  df_p3['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_9320\2577857134.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  df_p4['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_9320\2577857134.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  df_all['label'] = labels.values
```

In [ ]:

```
feature_sets = {
    "P1": df_p1,
    "P2": df_p2,
    "P3": df_p3,
    "P4": df_p4,
    "All Features": df_all
}
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score
```

In [ ]:

```
#!pip install git+https://github.com/christophM/rulefit.git
from rulefit import RuleFit
```

In [ ]:

```
for title in feature_sets:
    dfc = feature_sets[title]
    not_zero_ind = ~(dfc == 0).all(axis=1)

    dfc = dfc.loc[not_zero_ind]
    labels = dfc['label'].loc[not_zero_ind]


    scaler = StandardScaler()
    scaled_samples = scaler.fit_transform(dfc.iloc[:,:-2])

    X_train, X_test, y_train, y_test = train_test_split(scaled_samples, labels, test_size=0.2, random_state=42, stratif

    features = dfc.columns

    rf = RuleFit(model_type='r', rfmode='classify', max_iter=5000, n_jobs=-1) ## Classification task with only rule-bas
    rf.fit(X_train, y_train, feature_names=features)
    y_pred = rf.predict(X_test)
    res = f1_score(y_test, y_pred, average='weighted')
    print(rf, "f1", res)
    rules = rf.get_rules()
    rules = rules[rules.coef != 0].sort_values("support", ascending=False)
    rules.to_csv("reports/interpret/pose/rule-%s.csv" % title)
```

```
RuleFit(max_iter=1000, model_type='r', n_jobs=-1, rfmode='classify',
        tree_generator=GradientBoostingClassifier(learning_rate=0.01,
                                                  max_depth=100,
                                                  max_leaf_nodes=2,
                                                  n_estimators=550,
                                                  random_state=549,
                                                  subsample=0.09857775536929808)) f1 0.8184506891569311
RuleFit(max_iter=1000, model_type='r', n_jobs=-1, rfmode='classify',
        tree_generator=GradientBoostingClassifier(learning_rate=0.01,
                                                  max_depth=100,
                                                  max_leaf_nodes=3,
                                                  n_estimators=567,
                                                  random_state=566,
                                                  subsample=0.09857775536929808)) f1 0.8080967814539559
RuleFit(max_iter=1000, model_type='r', n_jobs=-1, rfmode='classify',
        tree_generator=GradientBoostingClassifier(learning_rate=0.01,
                                                  max_depth=100,
                                                  max_leaf_nodes=9,
                                                  n_estimators=556,
                                                  random_state=555,
                                                  subsample=0.09857775536929808)) f1 0.801881292594165
RuleFit(max_iter=1000, model_type='r', n_jobs=-1, rfmode='classify',
        tree_generator=GradientBoostingClassifier(learning_rate=0.01,
                                                  max_depth=100,
                                                  max_leaf_nodes=5,
                                                  n_estimators=588,
                                                  random_state=587,
                                                  subsample=0.09857775536929808)) f1 0.8360663648295513
RuleFit(max_iter=1000, model_type='r', n_jobs=-1, rfmode='classify',
        tree_generator=GradientBoostingClassifier(learning_rate=0.01,
                                                  max_depth=100,
                                                  max_leaf_nodes=2,
                                                  n_estimators=546,
                                                  random_state=545,
                                                  subsample=0.09857775536929808)) f1 0.8564249976297093
```

In [ ]:

```
"""
pca = PCA()
rf = RandomForestClassifier(n_estimators=100, n_jobs=-1)

blackbox_model = Pipeline([('pca', pca), ('rf', rf)])
"""
blackbox_model = SVC(gamma=2, C=1, probability=True)
```

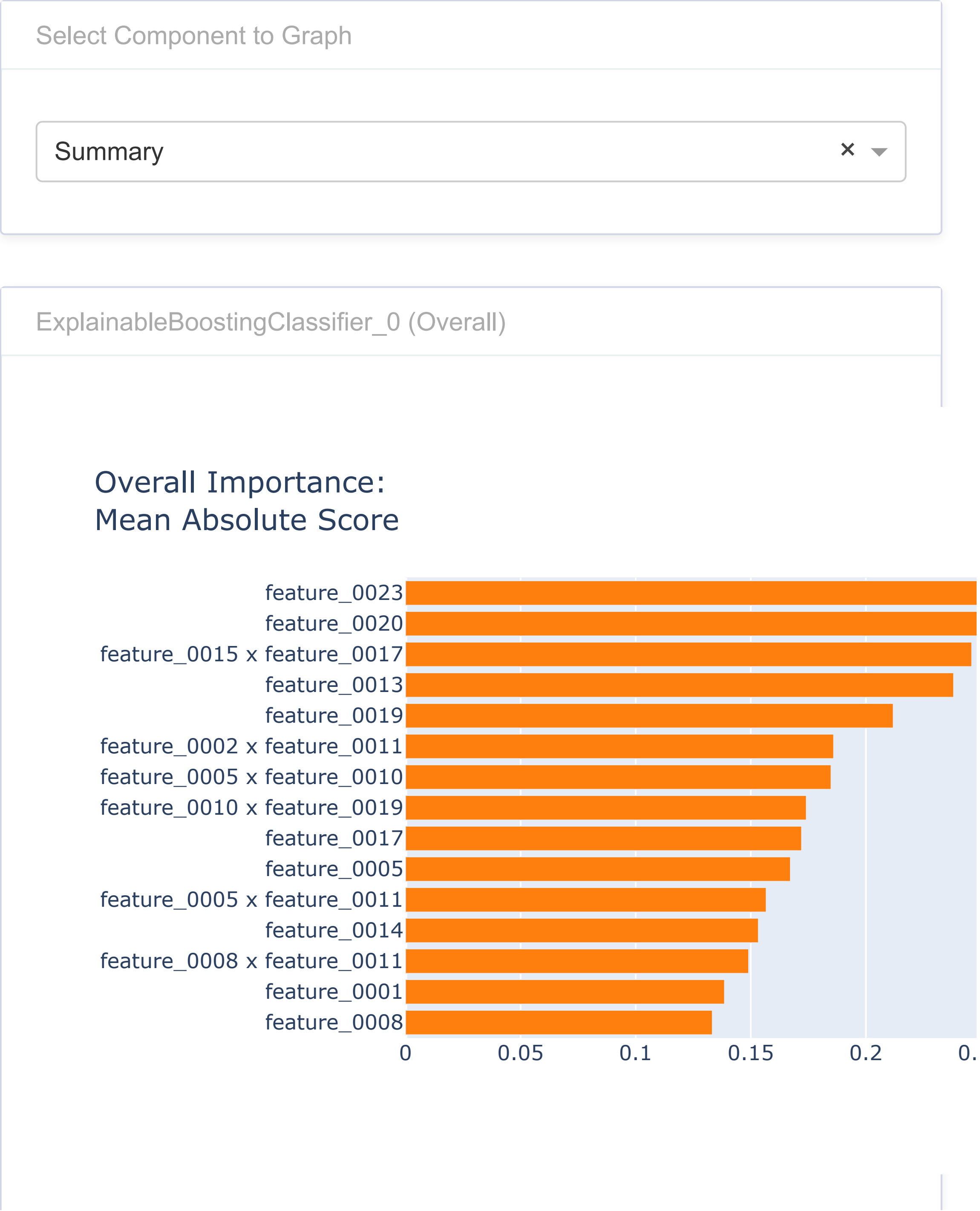
In [ ]:

```
from interpret import show
from interpret.perf import ROC
from interpret.blackbox import LimeTabular
from interpret import show
from interpret.blackbox import ShapKernel
from interpret.blackbox import MorrisSensitivity
from interpret.blackbox import PartialDependence
from interpret.glassbox import ExplainableBoostingClassifier
```

In [ ]:

```
for title in feature_sets:
    ebm = ExplainableBoostingClassifier()
    dfc = feature_sets[title]
    not_zero_ind = ~(dfc == 0).all(axis=1)

    dfc = dfc.loc[not_zero_ind]
    labels = dfc['label'].loc[not_zero_ind]


    scaler = StandardScaler()
    scaled_samples = scaler.fit_transform(dfc.iloc[:,:-1])

    X_train, X_test, y_train, y_test = train_test_split(scaled_samples, labels, test_size=0.2, random_state=42, stratif

    ebm.fit(X_train, y_train)
    ebm_global = ebm.explain_global()
    show(ebm_global)
```

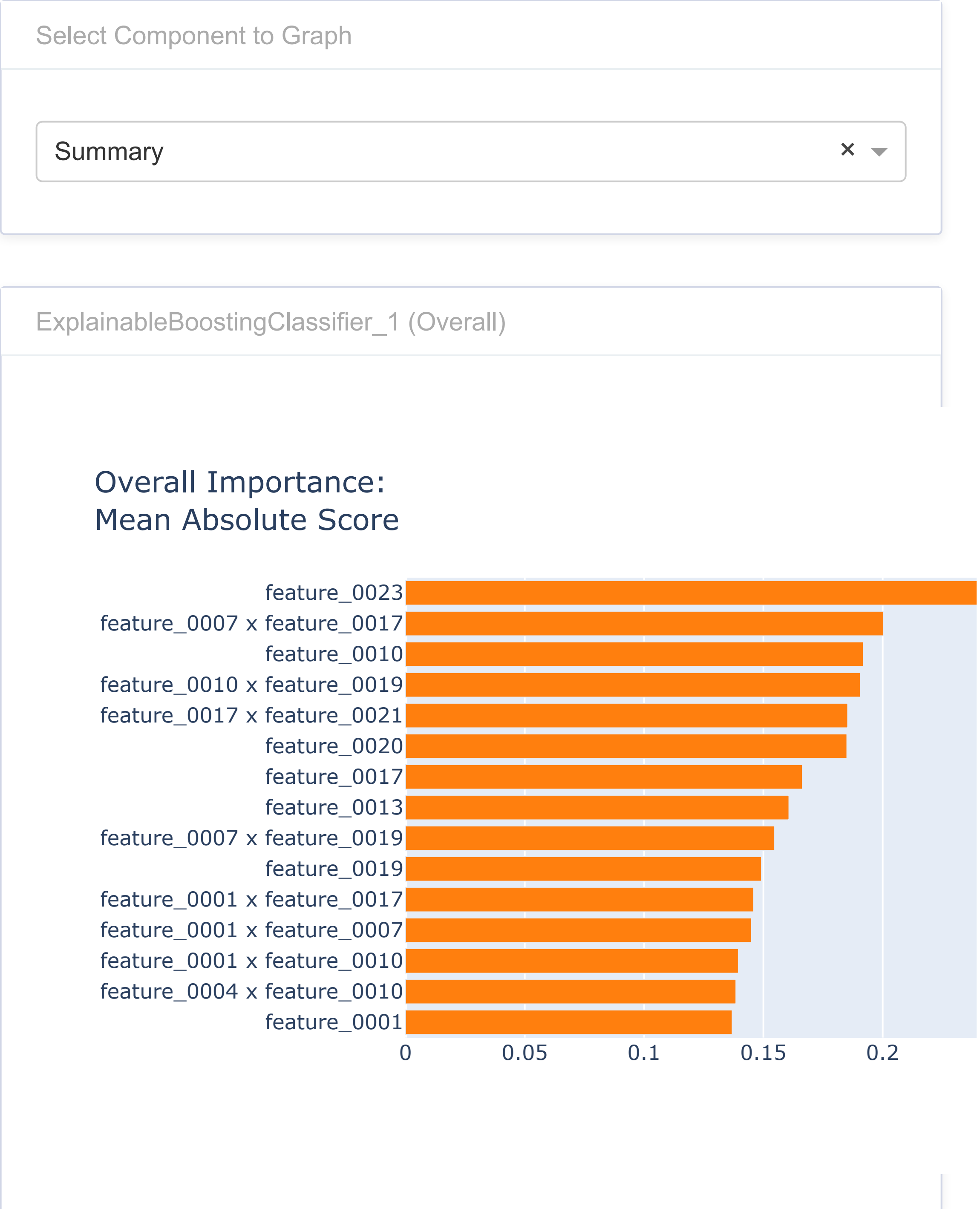
```
c:\Users\ASABUNCUOGLU13\Anaconda3\lib\site-packages\interpret\visual\udash.py:5: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
c:\Users\ASABUNCUOGLU13\Anaconda3\lib\site-packages\interpret\visual\udash.py:6: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
  import dash_core_components as dcc
c:\Users\ASABUNCUOGLU13\Anaconda3\lib\site-packages\interpret\visual\udash.py:7: UserWarning:
The dash_table package is deprecated. Please replace
`import dash_table` with `from dash import dash_table`

Also, if you're using any of the table format helpers (e.g. Group), replace
`from dash_table.Format import Group` with
`from dash.dash_table.Format import Group`
  import dash_table as dt
```



Select Component to Graph

Summary

ExplainableBoostingClassifier_0 (Overall)

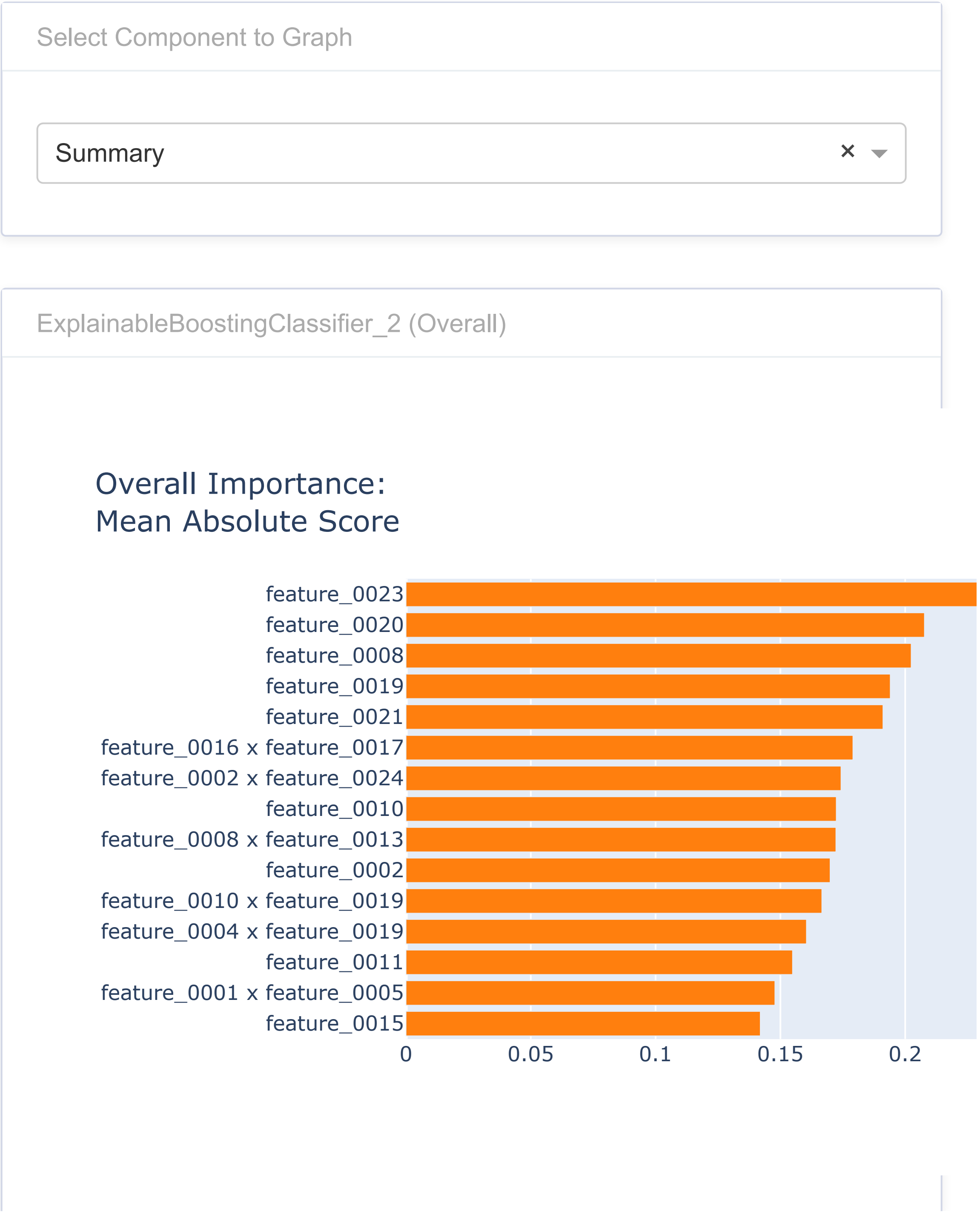


Select Component to Graph

Summary

ExplainableBoostingClassifier_1 (Overall)

Overall Importance:
Mean Absolute Score
feature_0023
feature_0007 x feature_0017
feature_0010
feature_0010 x feature_0019
feature_0017 x feature_0021
feature_0020
feature_0017
feature_0013
feature_0007 x feature_0019
feature_0019
feature_0001 x feature_0017
feature_0001 x feature_0007
feature_0001 x feature_0010
feature_0004 x feature_0010
feature_0001
0
0.05
0.1
0.15
0.2

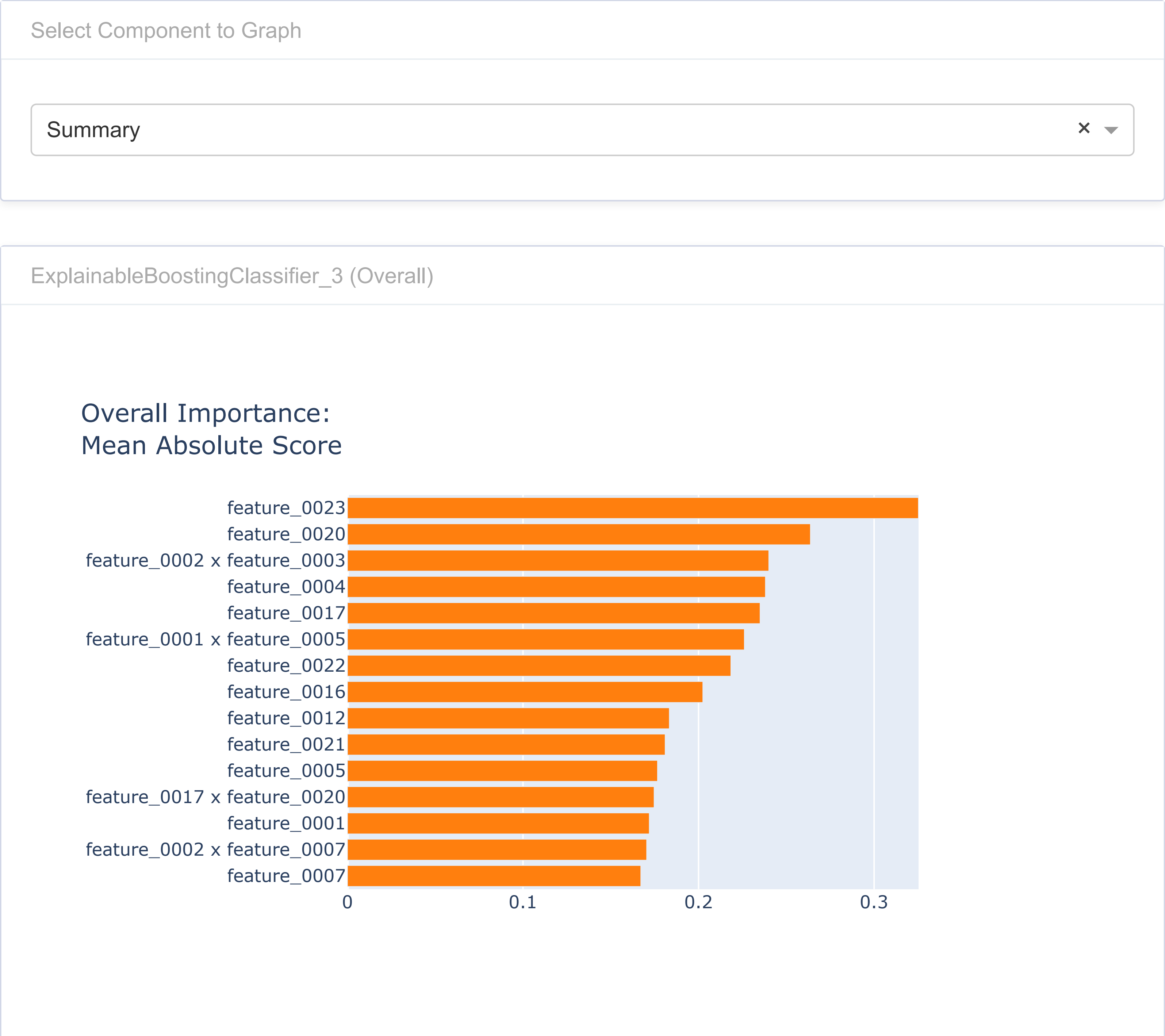


Select Component to Graph

Summary

ExplainableBoostingClassifier_2 (Overall)

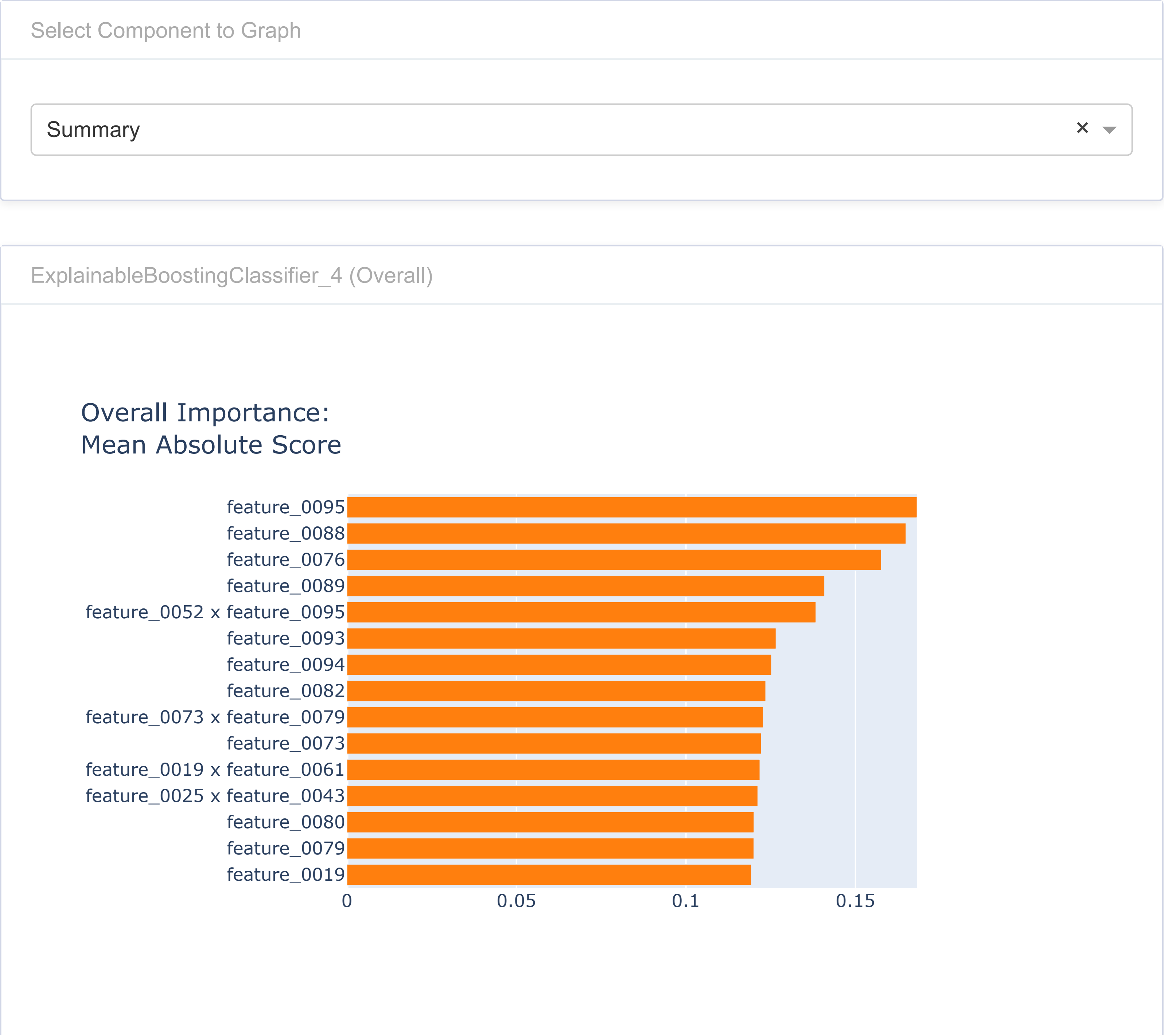
×
Overall Importance:
Mean Absolute Score
feature_0023
feature_0020
feature_0008
feature_0019
feature_0021
feature_0016 x feature_0017
feature_0002 x feature_0024
feature_0010
feature_0008 x feature_0013
feature_0002
feature_0010 x feature_0019
feature_0004 x feature_0019
feature_0011
feature_0001 x feature_0005
feature_0015
0
0.05
0.1
0.15
0.2



Select Component to Graph

Summary

ExplainableBoostingClassifier_3 (Overall)



Select Component to Graph

Summary

ExplainableBoostingClassifier_4 (Overall)

In [ ]:

```
for title in feature_sets:
    dfc = feature_sets[title]
    not_zero_ind = ~(dfc == 0).all(axis=1)

    dfc = dfc.loc[not_zero_ind]
    labels = dfc['label'].loc[not_zero_ind]

    scaler = StandardScaler()
    scaled_samples = scaler.fit_transform(dfc.iloc[:,:-1])

    X_train, X_test, y_train, y_test = train_test_split(scaled_samples, labels, test_size=0.2, random_state=42, stratif

    blackbox_model.fit(X_train, y_train)
    try:
        sensitivity = MorrisSensitivity(predict_fn=blackbox_model.predict_proba, data=X_train)
        sensitivity_global = sensitivity.explain_global(name="Global Sensitivity")

        show(sensitivity_global)

    except ValueError:
        print("zero-size array to reduction operation maximum which has no identity")
```

```
zero-size array to reduction operation maximum which has no identity
zero-size array to reduction operation maximum which has no identity
zero-size array to reduction operation maximum which has no identity
zero-size array to reduction operation maximum which has no identity
zero-size array to reduction operation maximum which has no identity
```

In [ ]: