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 [ ]:
```
person_name = "yagmur"
base_dir = "../face/features/%s" % person_name
labels_path = "../scores/all_levels_binary.csv"
```

In [ ]:
```
labels = pd.read_csv(labels_path)[person_name]
```

In [ ]:
```
# Load data
df = pd.read_csv(os.path.join(base_dir, 'features.csv'))
# Remove empty spaces in column names.
df.columns = [col.replace(" ", "") for col in df.columns]
# Print few values of data.
df.head()
```

Out[ ]:

	frame	face_id	timestamp	confidence	success	gaze_0_x	gaze_0_y	gaze_0_z	gaze_1_x	gaze_1_y	...	AU12_c	AU14_c	AU15_c	AU1
0	1	0	0.0	0.98	1	0.309688	0.217516	-0.925624	-0.078749	0.124520	...	1.0	1.0	0.0	
1	2	0	0.0	0.88	1	-0.744108	-0.003582	-0.668050	-0.731003	0.035917	...	0.0	0.0	0.0	
2	3	0	0.0	0.98	1	-0.730967	-0.056477	-0.680072	-0.739313	-0.004669	...	0.0	1.0	0.0	
3	4	0	0.0	0.98	1	-0.688489	-0.079700	-0.720854	-0.715481	0.028073	...	0.0	1.0	0.0	
4	5	0	0.0	0.98	1	-0.623707	-0.017633	-0.781459	-0.733528	0.004099	...	0.0	1.0	1.0	

5 rows × 714 columns

In [ ]:
```
print(len(df), len(labels))
```

```
6969 6969
```

In [ ]:
```
df.describe()
```

Out[ ]:

	frame	face_id	timestamp	confidence	success	gaze_0_x	gaze_0_y	gaze_0_z	gaze_1_x	gaze_1_y	...	
count	6969.000000	6969.0	6969.0	6969.000000	6969.000000	6969.000000	6969.000000	6969.000000	6969.000000	6969.000000	...	696
mean	215.962548	0.0	0.0	0.766288	0.772421	0.130164	0.126324	-0.715448	-0.048672	0.084677	...	
std	157.337614	0.0	0.0	0.362595	0.419300	0.215496	0.165907	0.392075	0.196111	0.131894	...	
min	1.000000	0.0	0.0	0.000000	0.000000	-0.799789	-0.466460	-1.000000	-0.776111	-0.508339	...	
25%	86.000000	0.0	0.0	0.880000	1.000000	0.000000	0.000000	-0.966496	-0.150220	0.000000	...	
50%	182.000000	0.0	0.0	0.980000	1.000000	0.122265	0.122958	-0.913610	0.000000	0.075680	...	
75%	327.000000	0.0	0.0	0.980000	1.000000	0.291115	0.257790	-0.789262	0.035816	0.183527	...	
max	666.000000	0.0	0.0	0.980000	1.000000	0.847260	0.578685	0.000000	0.776535	0.501084	...	

8 rows × 714 columns

In [ ]:
```
high_conf_ind = ~np.logical_or(df['confidence'] < 0.5, df['success'] == 0)

df = df.loc[high_conf_ind]
labels = labels.loc[high_conf_ind]
```

In [ ]:
```
print(len(df), len(labels))
```

```
5381 5381
```

In [ ]:
```
labels.value_counts()
```

Out[ ]:
```
1    5176
0     205
Name: yagmur, dtype: int64
```

In [ ]:
```
# Define Feature Series Ranges
r_au_intensities = range(df.columns.get_loc("AU01_r"), df.columns.get_loc("AU45_r"))
r_au_class = range(df.columns.get_loc("AU01_c"), df.columns.get_loc("AU45_c"))
r_3d_eye_landmarks = range(df.columns.get_loc("eye_lmk_X_0"), df.columns.get_loc("eye_lmk_Z_55"))
r_gaze_directions = range(df.columns.get_loc("gaze_0_x"), df.columns.get_loc("gaze_angle_y"))
```

```
r_pose = range(df.columns.get_loc("pose_Tx"), df.columns.get_loc("pose_Rz"))
r_3d_face_landmarks = range(df.columns.get_loc("X_0"), df.columns.get_loc("Z_67"))
```

In [ ]:
```
df_au_intensities = df.iloc[:, r_au_intensities]
df_au_class = df.iloc[:, r_au_class]
df_3d_eye_landmarks = df.iloc[:, r_3d_eye_landmarks]
df_gaze_directions = df.iloc[:, r_gaze_directions]
df_pose = df.iloc[:, r_pose]
df_3d_face_landmarks = df.iloc[:, r_3d_face_landmarks]
```

In [ ]:
```
df_au_intensities['label'] = labels.values
df_au_class['label'] = labels.values
df_3d_eye_landmarks['label'] = labels.values
df_gaze_directions['label'] = labels.values
df_pose['label'] = labels.values
df_3d_face_landmarks['label'] = labels.values
```

```
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_18012\819994795.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_au_intensities['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_18012\819994795.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_au_class['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_18012\819994795.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_3d_eye_landmarks['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_18012\819994795.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_gaze_directions['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_18012\819994795.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_pose['label'] = labels.values
C:\Users\ASABUNCUOGLU13\AppData\Local\Temp\ipykernel_18012\819994795.py:6: 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_3d_face_landmarks['label'] = labels.values
```

In [ ]:

```
df_face_and_pose = pd.concat([df_3d_face_landmarks.iloc[:, :-1],
df_pose],axis=1)
```

```
df_all = pd.concat([df_3d_eye_landmarks.iloc[:, :-1],
df_au_intensities.iloc[:, :-1],
df_gaze_directions.iloc[:, :-1],
df_3d_face_landmarks.iloc[:, :-1],
df_pose],axis=1)


feature_sets = {
    "AU Intensity": df_au_intensities,
    "3D Eye Landmark": df_3d_eye_landmarks,
    "3D Face Landmark": df_3d_face_landmarks,
    "Gaze Directions": df_gaze_directions,
    "Head Pose": df_pose,
    "3D Face and Head Pose": df_face_and_pose,
    "All OpenFace Fts": 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/%s/rule-%s.csv" % (person_name, 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=8,
                                                  n_estimators=565,
                                                  random_state=564,
                                                  subsample=0.11469081487094829)) f1 0.9536517177833821
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=555,
                                                  random_state=554,
                                                  subsample=0.11469081487094829)) f1 0.9418720599171825
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=6,
                                                  n_estimators=559,
                                                  random_state=558,
                                                  subsample=0.11469081487094829)) f1 0.9506059829805349
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=548,
                                                  random_state=547,
                                                  subsample=0.11469081487094829)) f1 0.9435081896662996
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=4,
                                                  n_estimators=571,
                                                  random_state=570,
                                                  subsample=0.11469081487094829)) f1 0.9475060643816446
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=4,
                                                  n_estimators=549,
                                                  random_state=548,
                                                  subsample=0.11469081487094829)) f1 0.9469428623909881
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=583,
                                                    random_state=582,
                                                    subsample=0.11469081487094829)) f1 0.9442806902594673
```

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
```

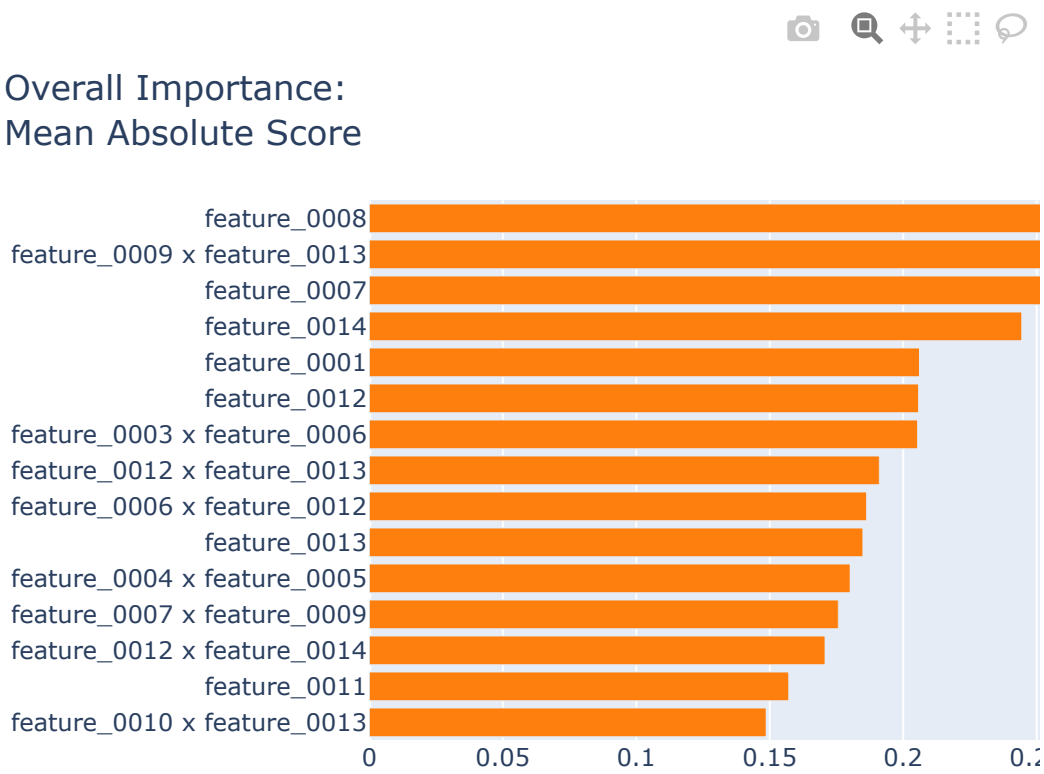
In [ ]:
```
for title in feature_sets:
    ebm = ExplainableBoostingClassifier()
    dfc = feature_sets[title]
    not_zero_ind = ~(dfc == 0).all(axis=1)

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


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

    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

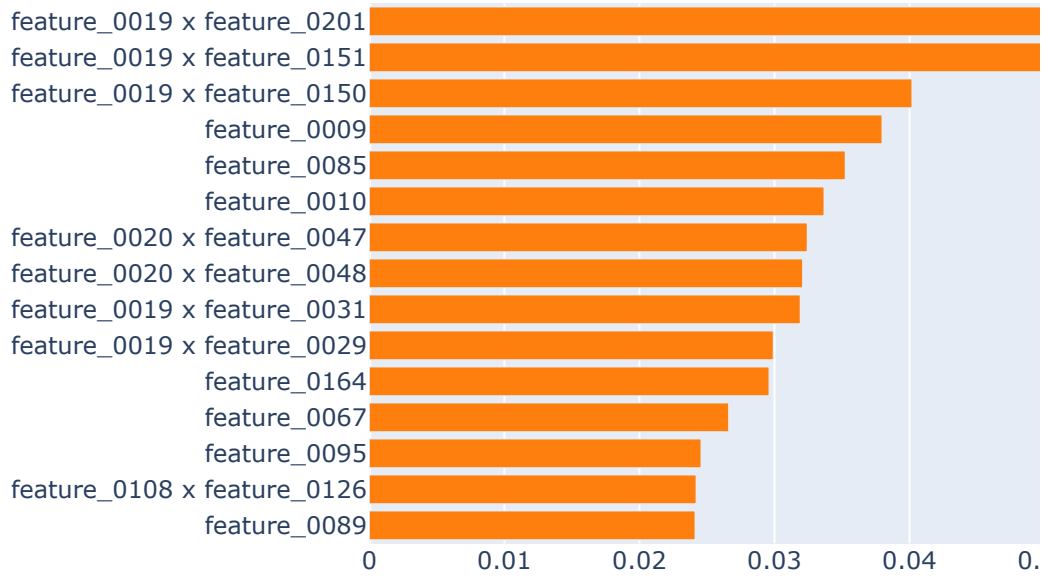
Summary

ExplainableBoostingClassifier_1 (Overall)

Select Component to Graph

Summary

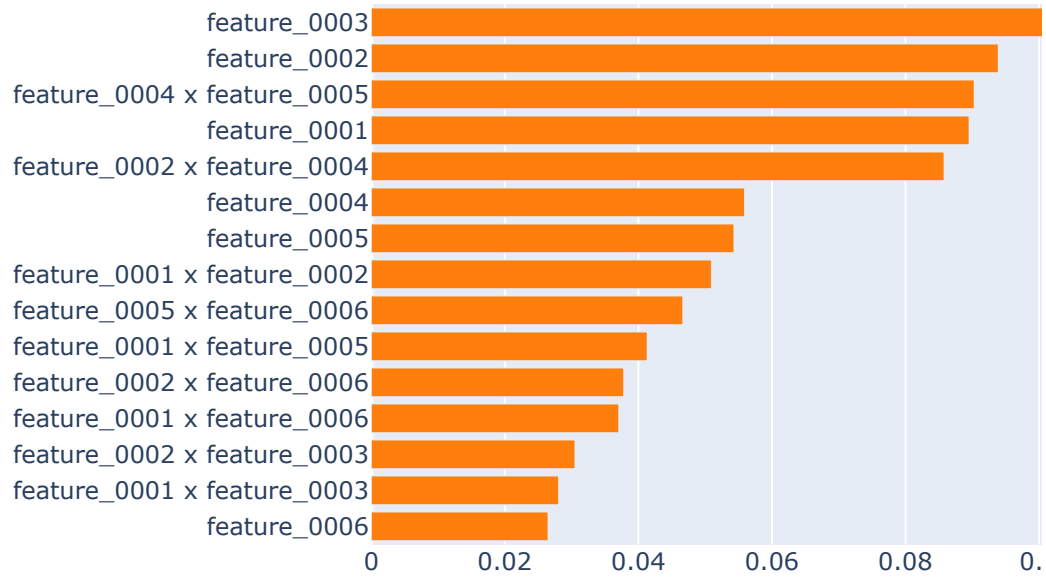
ExplainableBoostingClassifier_2 (Overall)



Select Component to Graph

Summary

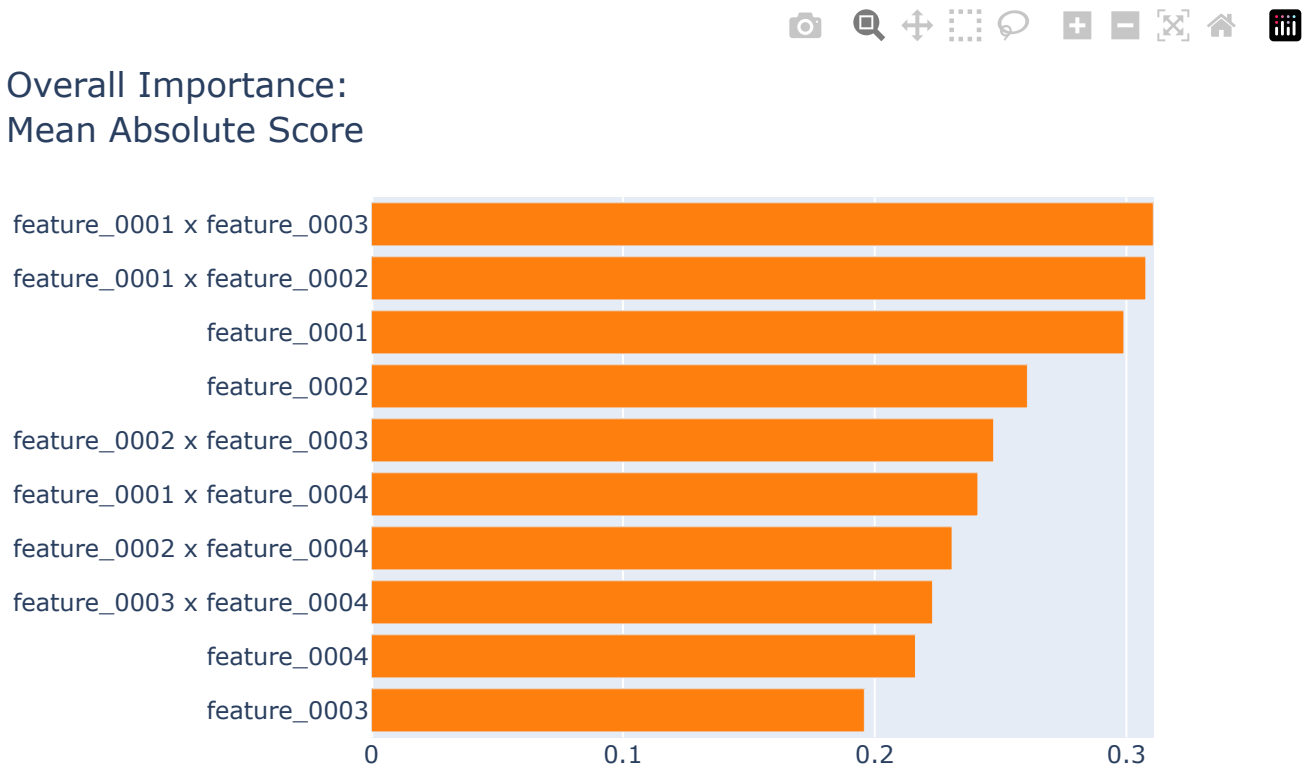
ExplainableBoostingClassifier_3 (Overall)



Select Component to Graph

Summary

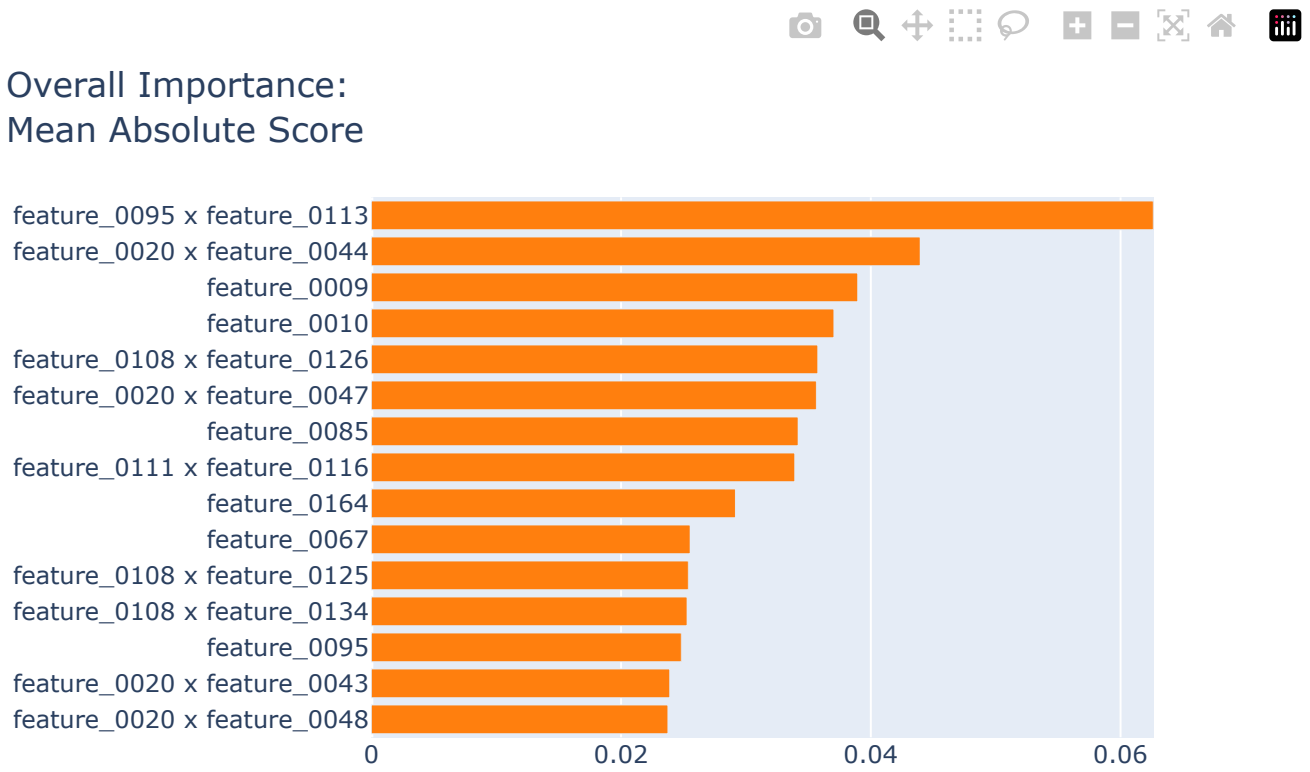
ExplainableBoostingClassifier_4 (Overall)



Select Component to Graph

Summary

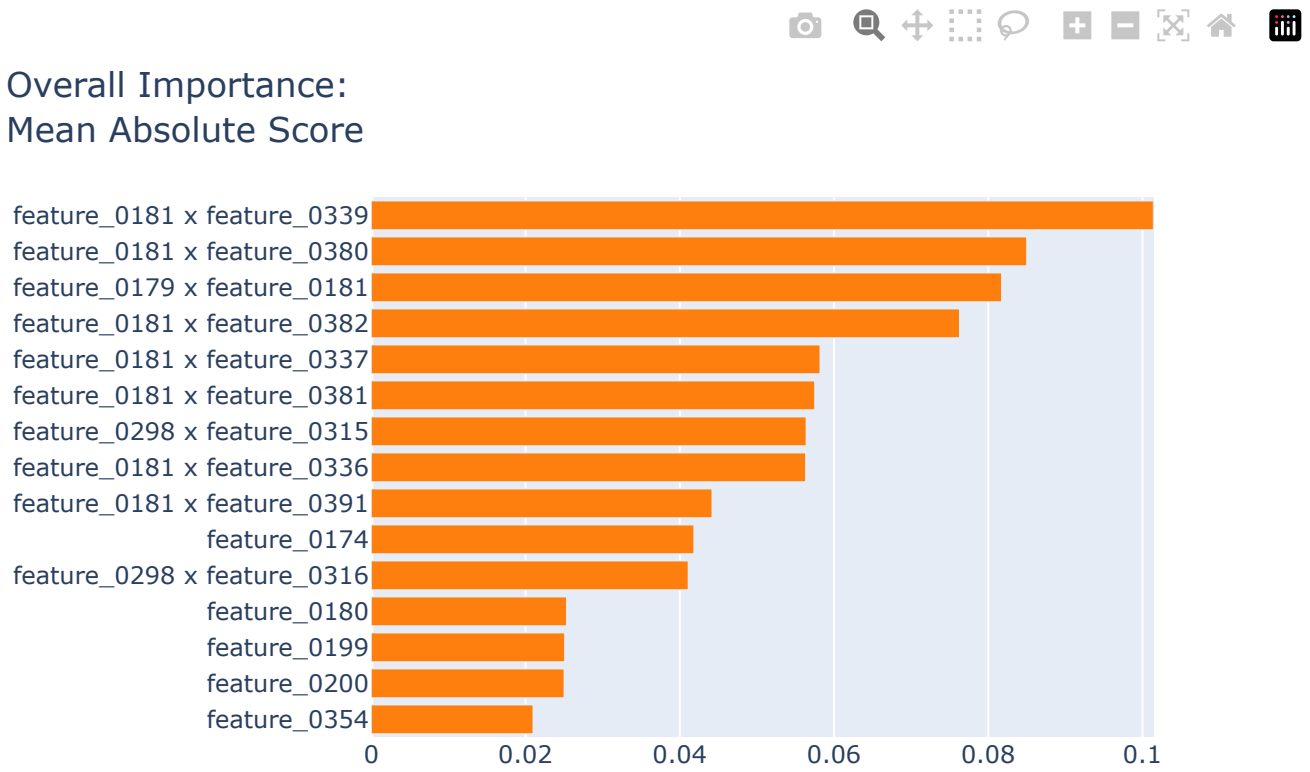
ExplainableBoostingClassifier_5 (Overall)



Select Component to Graph

Summary

ExplainableBoostingClassifier_6 (Overall)



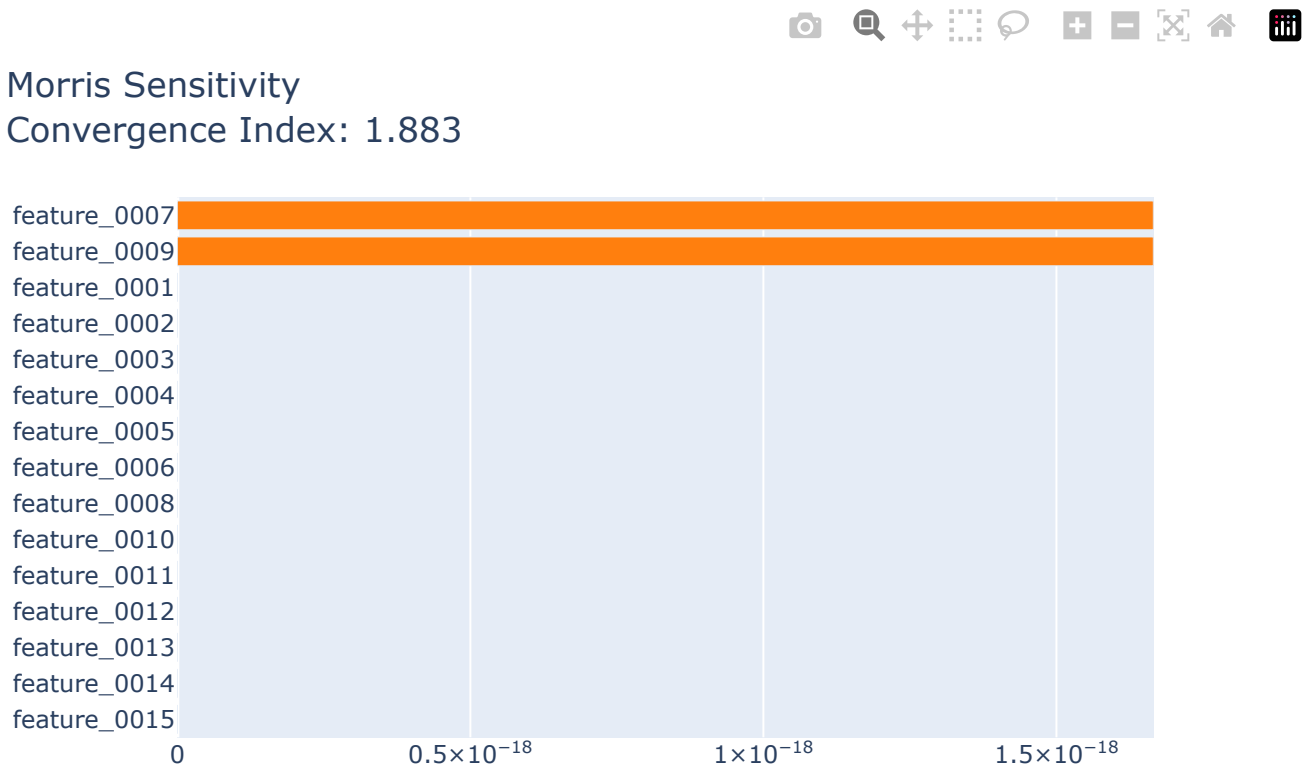
In [ ]:

```
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[:,:-2])

    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")
```

Select Component to Graph

Summary

Global Sensitivity (Overall)

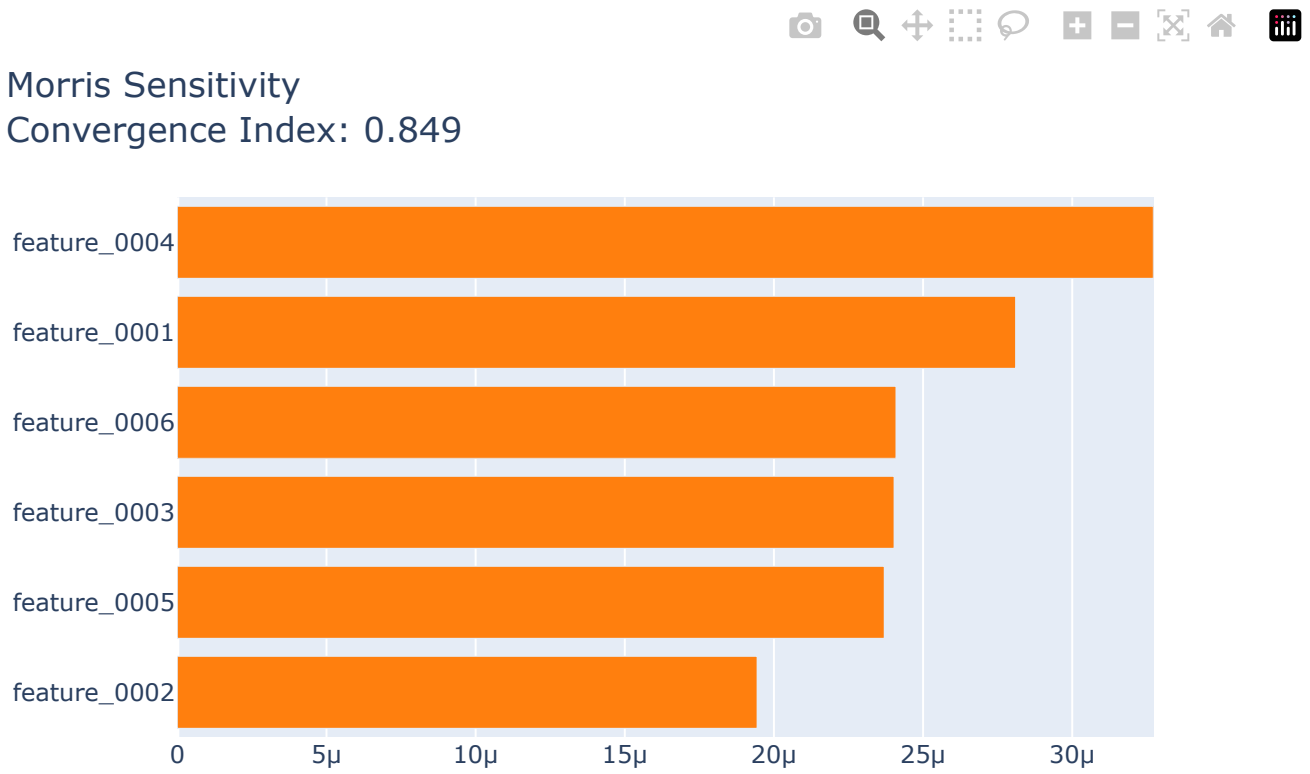


```
zero-size array to reduction operation maximum which has no identity
zero-size array to reduction operation maximum which has no identity
```

Select Component to Graph

Summary

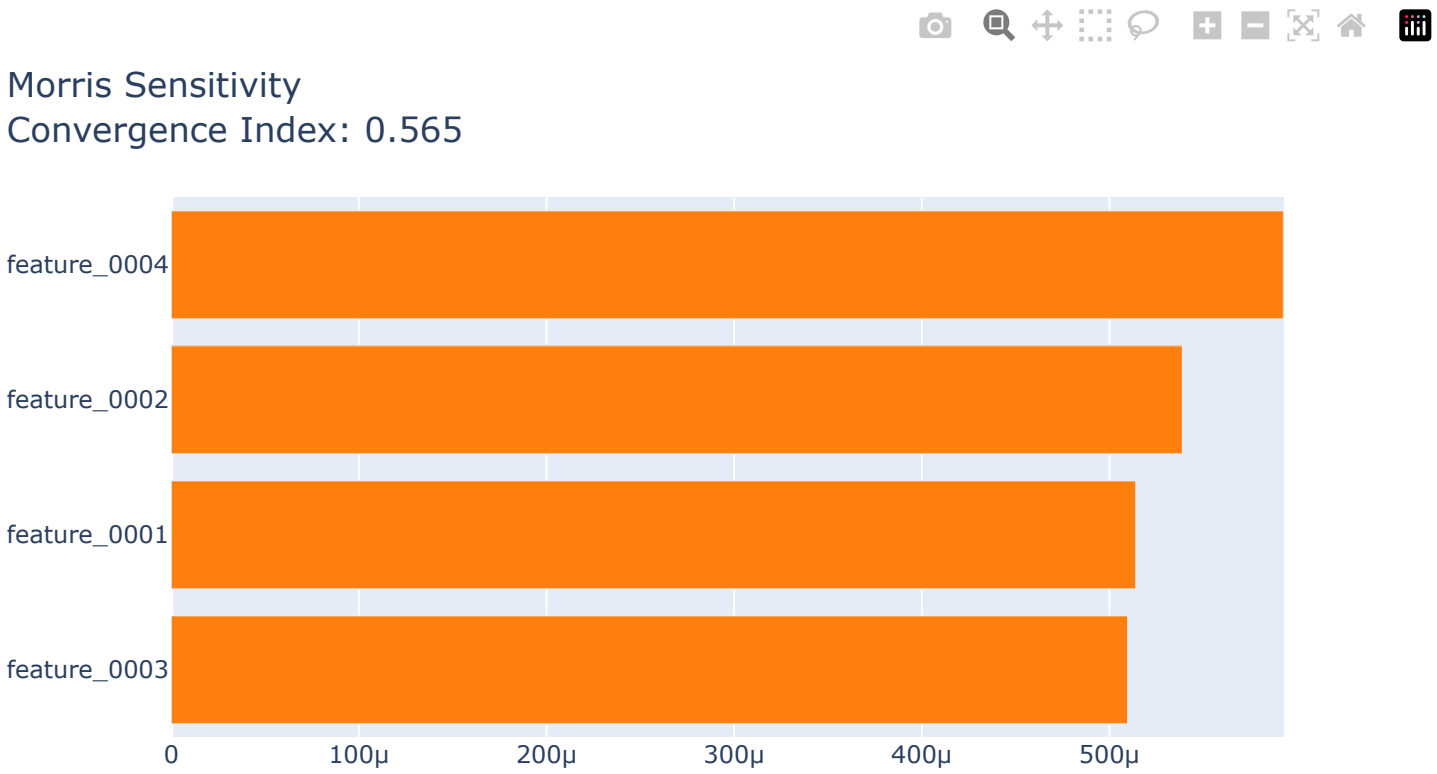
Global Sensitivity (Overall)



Select Component to Graph

Summary

Global Sensitivity (Overall)



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

In [ ]: